20 лет. Юбилейное издание

# UNIX и LINUX
# РУКОВОДСТВО СИСТЕМНОГО АДМИНИСТРАТОРА

## 4-Е ИЗДАНИЕ




**ЭВИ НЕМЕТ, ГАРТ СНАЙДЕР, ТРЕНТ ХЕЙН, БЭН УЭЙЛИ**

*при участии Терри Морреале, Нэда Мак-Клейна, Рона Якима, Дэвида Швайкерта и Тоби Отикера*

# UNIX AND LINUX SYSTEM ADMINISTRATION HANDBOOK

Fourth Edition

Evi Nemeth
Garth Snyder
Trent R. Hein
Ben Whaley

*with Terry Morreale, Ned McClain, Ron Jachim, David Schweikert, and Tobi Oetiker*


Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

# UNIX И LINUX
# РУКОВОДСТВО СИСТЕМНОГО АДМИНИСТРАТОРА

4-е издание

Эви Немет
Гарт Снайдер
Трент Хейн
Бэн Уэйли

*при участии Терри Морреале, Нэда Макклейна, Рона Якима, Дэвида Швайкерта и Тоби Оликера*

Москва • Санкт-Петербург • Киев
2012

ББК 32.973.26-018.2.75
Н50
УДК 681.3.07

Издательский дом "Вильямс"
Зав. редакцией *С.Н. Тригуб*
Перевод с английского докт. физ.-мат. наук *Д.А. Клюшина* и *Н.М. Ручко*
Под редакцией докт. физ.-мат. наук *Д.А. Клюшина*
По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

**Немет**, Эви, **Снайдер**, Гарт, **Хейн**, Трент, **Уэйли**, Бэн.
Н50 Unix и Linux: руководство системного администратора, 4-е изд. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2012. — 1312 с. : ил. — Парал. тит. англ.
ISBN 978-5-8459-1740-9 (рус.)

**ББК 32.973.26-018.2.75**



*Научно-популярное издание*
**Эви Немет, Гарт Снайдер, Трент Хейн, Бэн Уэйли**

# Unix и Linux: руководство системного администратора

**4-е издание**

Литературный редактор *Е.Д. Давидян*
Верстка *М.А. Удалов*
Художественный редактор *В.Г. Павлютин*
Корректор *Л.А. Гордиенко*

Подписано в печать 07.12.2011. Формат 70x100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 105,78. Уч.-изд. л. 96,3.
Тираж 1000 экз. Заказ № 27034.

Отпечатано по технологии CtP в ОАО "Первая Образцовая типография",
обособленное подразделение "Печатный двор".
197110, Санкт-Петербург, Чкаловский пр., 15.

ООО "И. Д. Вильямс", 127055, г. Москва, ул. Лесная, д. 43, стр. 1

SBN 978-5-8459-1740-9 (рус.)
SBN 978-0-13-148005-6 (англ.)

# Оглавление

# Содержание

# Об авторах

**Эви Немет** уволилась с факультета вычислительной техники Университета штата Колорадо и в настоящее время бороздит просторы Тихого океана на своей 40-футовой яхте "Wonderland" ("Страна чудес"). Это — ее последнее издание, невозможно быть в курсе современных "сисадминовских игрушек", стоя на якоре в каком-то райском уголке с пакетом EMail Connection для радиосвязи со скоростью 30 бод.

sailingevi@gmail.com

**Гарт Снайдер** работал в компаниях NeXT и Sun и получил степень бакалавра электротехники в колледже Суортмор, штат Пенсильвания, а также ученую степень в Университете Рочестера, штат Нью-Йорк.

garth@garthsnyder.com

**Трент Р. Хейн** — один из основателей компании Applied Trust Engineering, которая разрабатывает средства защиты и анализа производительности сетей. Трент получил степень бакалавра вычислительной техники в Университете штата Колорадо.

trent@atrust.com

**Бен Уэйли** — руководитель отдела архитектуры предприятий (Enterprise Architecture) в компании Applied Trust Engineering, которая занимается консалтингом в области информационных технологий (г. Боулдер, штат Колорадо). В 2004 году он получил диплом бакалавра по специальности "Вычислительная техника" в Университете штата Колорадо.

ben@atrust.com

# О соавторах

**Терри Морреале** (Terry Morreale) — старший инженер и руководитель отдела по обслуживанию клиентов в Applied Trust. Она получила диплом по специальности "Вычислительная техника" в Университете штата Колорадо, а также имеет следующие сертификаты: CISSP, GIAC Gold Certified Incident Handler и ITILv3 Foundations. Свободное от работы время Терри посвящает заботе о двух своих детях, чтению и бегу.

**Нэд Мак-Клейн** (Ned McClain) (`ned@atrust.com`) — один из основателей и руководитель технического отдела в Applied Trust Engineering. Он помогает клиентам разного уровня в вопросах архитектуры и эксплуатации систем. Работает над решением проблем производительности, бесперебойности и безопасности работы систем. Но особенно его интересует область системного администрирования. Нэд получил диплом по специальности "Вычислительная техника" в техническом колледже при Корнеллском университете и имеет сертификаты CISSP, MCP и ITIL. Нэд регулярно обновляет содержимое своего блога (`arkingseal.com`).

**Рон Яким** (Ron Jachim) получил степень магистра в Университете Уэйна в Детройте (штат Мичиган), где в настоящее время читает лекции в качестве адъюнкт-профессора. В течение 20 лет он набирался опыта по использованию систем UNIX как в процессе преподавания, так и работая в Ford Motor Company. Он сочетает навыки администрирования с пристрастием к поиску решений по созданию эластичных инфраструктур, включающих тысячи серверов, и к повышению производительности глобальных приложений.

**Дэвид Швайкерт** (David Schweikert) работает менеджером по продукции в Open Systems AG (Швейцария) — организации, предоставляющей услуги по обеспечению безопасности систем. Его команда отвечает за управление конфигурированием и мониторинг более чем 1 500 серверов UNIX в более чем 100 странах. Девид является разработчиком проектов с открытым кодом Mailgraph (инструментальное средство для визуализации статистики электронной почты) и Postgrey (реализация Postfix); см. `david.schweikert.ch`.

**Тоби Отикер** (Tobi Oetiker) — инженер-электрик по образованию и системный администратор по призванию. В течение десяти лет он работал в швейцарском федеральном технологическом институте (Swiss Federal Institute of Technology), где создал роскошную UNIX-среду для студентов и персонала. С 2006 года Тоби трудится в собственной компании OETIKER+PARTNER AG, где занимается поддержкой UNIX-серверов для промышленных клиентов, совершенствует свои любимые проекты с открытым кодом (MRTG™, RRDtool и SmokePing) и применяет этот инструментарий для решения проблем своих клиентов. В ноябре 2006 года Тоби получил престижную награду (SAGE Outstanding Achievement Award) за работу над пакетами MRTG и RRDtool; см. `tobi.oetiker.ch`.

# Предисловие

Двадцать семь лет назад, т.е. в 1983 году, я написал, пожалуй, первое руководство по системному администрированию для операционной системы UNIX. Меня пригласили в Массачусетсскую компанию (Massachusetts Computer Company — MASSCOMP), специализирующуюся на UNIX-машинах, для написания документации. Завершив означенную в контракте работу над руководством по графическому программированию, я прикидывал, к чему бы еще мне можно было здесь приложить свои силы. "При возникновении системных проблем мы всегда обращаемся к Тому Тейкшейре, — сказал я. — А что делать нашим клиентам?"

Ответ не заставил себя ждать: "Нам просто позарез нужно практическое руководство". И вскоре я снова работал, но на этот раз должен был извлечь максимум полезной информации из головы Тома Тейкшейры и перенести ее на бумагу.

В результате получилась книга, в которой были описаны такие основные понятия, как суперпользователь, добавление учетной записи, управление разрешениями, резервирование и восстановление, организация сети с помощью протокола UUCP и пр. Книга была ориентирована на System V, одну из двух главных (на то время) разновидностей UNIX (другой была Berkeley UNIX).

В конечном счете я вполне справился с поставленной передо мной задачей — добыл ценную информацию от Тома и других членов немногочисленной (на то время) касты элитных системных администраторов. И когда в 1989 году увидела свет книга *UNIX System Administration Handbook* (USAH), у меня уже не было сомнений в том, что это авторитетное руководство — "библия" в своем роде — получило заслуженное признание, и неудивительно: ведь оно вышло непосредственно из-под пера (то бишь клавиатуры) истинных мастеров своего дела.

К тому времени О'Рейли стал издателем. Узнав, что многие мои клиенты (в сфере написания технической документации) перешли на UNIX, я начал сохранять права на написанные мною руководства, чтобы можно было перепродавать их другим компаниям. В конце 1985 года мы представили наши первые книги, которые уже не лицензировались для компаний, а находились в открытой продаже. Сначала мы выпускали небольшие книги, посвященные таким отдельным темам, как **vi**, **sed** и **awk**, **termcap** и **terminfo**, или протоколу UUCP. Мы называли их "Nutshell Handbooks" ("Справочники в двух словах" — *Примеч. ред.*), поскольку хотели охватить все темы и собрать их под одной "оболочкой".

В действительности мы ничего не знали об издательском деле. Наши книги не имели ни корешков (они сшивались скобками), ни предметных указателей, ни ISBN (международный стандартный номер книги). Мы продавали их по почте, а не через книжные магазины. Но мало-помалу мы осваивали эту сферу деятельности и в конце концов стали конкурировать с известными издателями компьютерных книг.

Любимая для нас тема — общие методы администрирования UNIX, но до недавних пор мы не брались за нее. И вот почему. Я — сторонник удовлетворения возникших потребностей без конкуренции с кем-либо ради этого. И было абсолютно ясно, что по означенной теме уже есть не просто хорошая, а ВЕЛИКОЛЕПНАЯ книга! Я не видел ни необходимости соперничать с такой всеобъемлющей книгой, ни возможности добиться успеха в этом направлении.

Но со временем, когда наш бизнес стал на ноги и мы вышли на рынок компьютерных книг, стало понятно, что конкуренция действительно может способствовать расширению рынка. Люди, как правило, видя одну книгу, воспринимают ее как нечто исключительное. Если же им на глаза попадается несколько книг по одной теме, то, как говорил певец Арло Гатри (Arlo Guthrie), "они могут подумать, что это движение".

Кроме того, в первом издании *USAH* авторы четко взяли курс на BSD-ориентированные системы (BSD UNIX), и мы подумали, что осталась свободной ниша для книги с уклоном в сторону System V.

В 1991 году мы выпустили в свет собственную книгу по системному администрированию UNIX, а именно Æleen Frisch, *Essential System Administration*.

Как автор, редактор и издатель, я никогда не придавал большого значения конкуренции — за исключением нескольких случаев. И это один из таких случаев. *UNIX System Administration Handbook* — это одна из немногих книг, на которые мы равняемся. Могли бы мы сделать так же хорошо? Могли бы мы сделать еще лучше? Подобно дуэли Мэджика Джонсона и Ларри Берда в NBA, соперничество выявляет лучшее, что есть в нас.

Ну вот, опять! Четвертое издание? Элин не мешало бы вернуться к работе! :-)

Тим О'Рейли
июнь 2010 г.

# Введение

Когда мы писали первое издание этой книги в средине 1980-х, нам очень хотелось сравнить нашу рукопись с другими книгами по системному администрированию. К нашему удивлению, мы отыскали только три. Теперь же вы можете выбирать из сотен книг. Назовем преимущества нашего издания.

- Эта книга является практическим руководством, в котором мы делимся с читателями нашим коллективным опытом системного администрирования и предлагаем рекомендации, которые выдержали испытание временем. Она содержит множество практических примеров и советов.
- Здесь не говорится о том, как использовать UNIX или Linux в домашних условиях. Мы описываем применение операционных систем в коммерческих компаниях, правительственных учреждениях, университетах.
- В книге подробно излагаются вопросы, связанные с работой UNIX- и Linux-систем в сетях. Это самый трудный аспект системного администрирования, и именно здесь наша помощь, как нам кажется, будет для читателей наиболее полезной.
- В этом руководстве рассмотрены основные варианты UNIX и Linux.

## Структура книги

Книга разбита на три части — "Основы администрирования", "Работа в сети" и "Разное".

В части I, "Основы администрирования", приводится общий обзор систем UNIX и Linux с точки зрения системного администратора. В главах этой части представлены основные сведения и описаны методики, необходимые для управления автономной системой. В этой части описаны протоколы, используемые в UNIX-системах, а также способы построения, расширения и администрирования сетей и интернет-серверов. Здесь же рассматривается высокоуровневое сетевое программное обеспечение. Среди изучаемых тем можно выделить систему доменных имен (DNS), сетевую файловую систему (NFS), систему электронной почты и инструменты сетевого управления.

В части II, "Работа в сети", описываются протоколы, используемые в системах UNIX, а также способы настройки, расширения и эксплуатации сетей и серверов, с выходом в Интернет. Кроме того, в ней рассматривается высокоуровневое сетевое программное обеспечение, а также система доменных имен, сетевая файловая система, электронная почта и управление сетью.

В части III, "Разное", содержит разнообразную вспомогательную информацию. В некоторых главах обсуждаются дополнительные функциональные возможности, такие как поддержка виртуализации серверов. В других главах даются рекомендации по самым разным темам: от экологического аспекта до принципов функционирования групп системного администрирования.

В конце каждой главы приводится набор практических заданий. Каждому упражнению дана оценка уровня сложности, причем учитывается как сложность самой задачи, так и время, затрачиваемое на ее выполнение.

Упражнения соответствуют четырем уровням сложности.

| | |
|---|---|
| звездочки отсутствуют | Простое задание, не требующее особых усилий |
| ★ | Более сложное или трудоемкое задание, для которого может потребоваться лабораторная работа |
| ★★ | Наиболее сложное или трудоемкое задание, требующее лабораторной работы |

★★★★★ Курсовые проекты (лишь в нескольких главах)

Для выполнения некоторых упражнений требуется доступ к системе с правами суперпользователя. Иногда нужно получить предварительное разрешение у системного администратора локальной сети. Подобные случаи оговариваются в задании.

## Наши помощники

Мы признательны Неду Мак-Клейну (Ned McClain), Девиду Швайкерту (David Schweikert) и Тоби Отикеру (Tobi Oetiker) за их вклад в создание книги. В написании этого издания принимали участие Терри Морреале (Terry Morreale) и Рон Яким (Ron Jachim). Их глубокие знания в самых разных областях существенно обогатили материал книги.

## Информация для контактов

Свои предложения и комментарии присылайте по адресу: `ulsah@book.admin.com`.

Мы действительно отвечаем на большинство писем, но, пожалуйста, будьте терпеливы. Иногда проходит несколько дней, прежде чем у кого-то из нас появляется возможность ответить на письмо. Для того чтобы просмотреть список выявленных на данный момент опечаток, а также другую свежую информацию по книге, посетите наш веб-сайт `www.admin.com`.

Надеемся, что книга вам понравится, и желаем удачи в системном администрировании.

*Эви Немет*
*Гарт Снайдер*
*Трент Р. Хейн*
*Бен Уэйли*
июнь 2010 г.

# Благодарности

Множество людей внесли ценный вклад в работу над книгой. Кто-то из них выполнял научное редактирование или предлагал интересные примеры, кто-то оказывал моральную поддержку. И все они заслуживают особой благодарности.

Рон Айтчисон (Ron Aitchison)
Эрик Олмен (Eric Allman)
Клей Бензигер (Clay Baenziger)
Адам Богз (Adam Boggs)
Том Кристиансен (Tom Christiansen)
Дэн Фостер (Dan Foster)
Стив Гэд (Steve Gaede)
Питер Хааг (Peter Haag)
Брайан Хелви (Bryan Helvey)
Матийс Меккинг (Matthijs Mekking)
Рэндалл Манро (Randall Munroe)
Эрик Остервейл (Eric Osterweil)
Фил Пеннок (Phil Pennock)
Уильям Путнем (William Putnam)
Джереми С. Рид (Jeremy C. Reed)
Энди Рудов (Andy Rudoff)
Михаил Синатра (Michael Sinatra)
Поль Викси (Paul Vixie)
Уотер Вийнгаардс (Wouter Wijngaards)

Наш редактор из Prentice Hall, Марк Тауб (Mark Taub), заслуживает не только благодарности, но и награды за то, что мужественно выдерживал капризы авторов. Он был бесконечно терпелив к нам и сделал все возможное, чтобы заставить нас сосредоточиться на повышении качества материала.

Нам очень повезло с рецензентами. Особенно хотелось бы отметить Джонатана Корбета (Jonathan Corbet) и Пэта Парсигьяна (Pat Parseghian) не только за их деликатные и подробные замечания, но и за их готовность вновь погрузиться в работу над будущими изданиями.

Мэри Лу Нор (Mary Lou Nohr) проделала огромную работу по оформлению настоящего издания. Мы высоко ценим ее усилия.

Потрясающие рисунки и обложка для этого издания придуманы и выполнены Лизой Хани (Lisa Haney). С ее портфолио можно ознакомиться на сайте `lisahaney.com`.

Линда Григолейт (Linda Grigoleit), Терри Хоффман (Terry Hoffman) и Джон Саливан (John Sullivan) оказали нам неоценимую помощь в переговорах с сетевой службой IBM, в результате чего мы получили необходимое оборудование для оценки качества нашей работы.

Благодарим также компанию Applied Trust (`appliedtrust.com`), руководство которой позволило использовать свои лабораторные площади и оказывало всяческую материально-техническую поддержку.

К сожалению, нам не удалось достичь соглашения, которое позволило бы нам публично выразить признание одному из наших выдающихся авторов. Тем не менее его вклад в проект был нами высоко оценен, и мы посылаем ему этот перевертень, который, несомненно, станет украшением его коллекции: "A man, a plan, a canoe, pasta, Hero's rajahs, a coloratura, maps, snipe, percale, macaroni, a gag, a banana bag, a tan, a tag, a banana bag again (or a camel), a crepe, pins, Spam, a rut, a Rolo, cash, a jar, sore hats, a peon, a canal — Panama!"

## От издательства

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: `info@williamspublishing.com`

WWW: `http://www.williamspublishing.com`

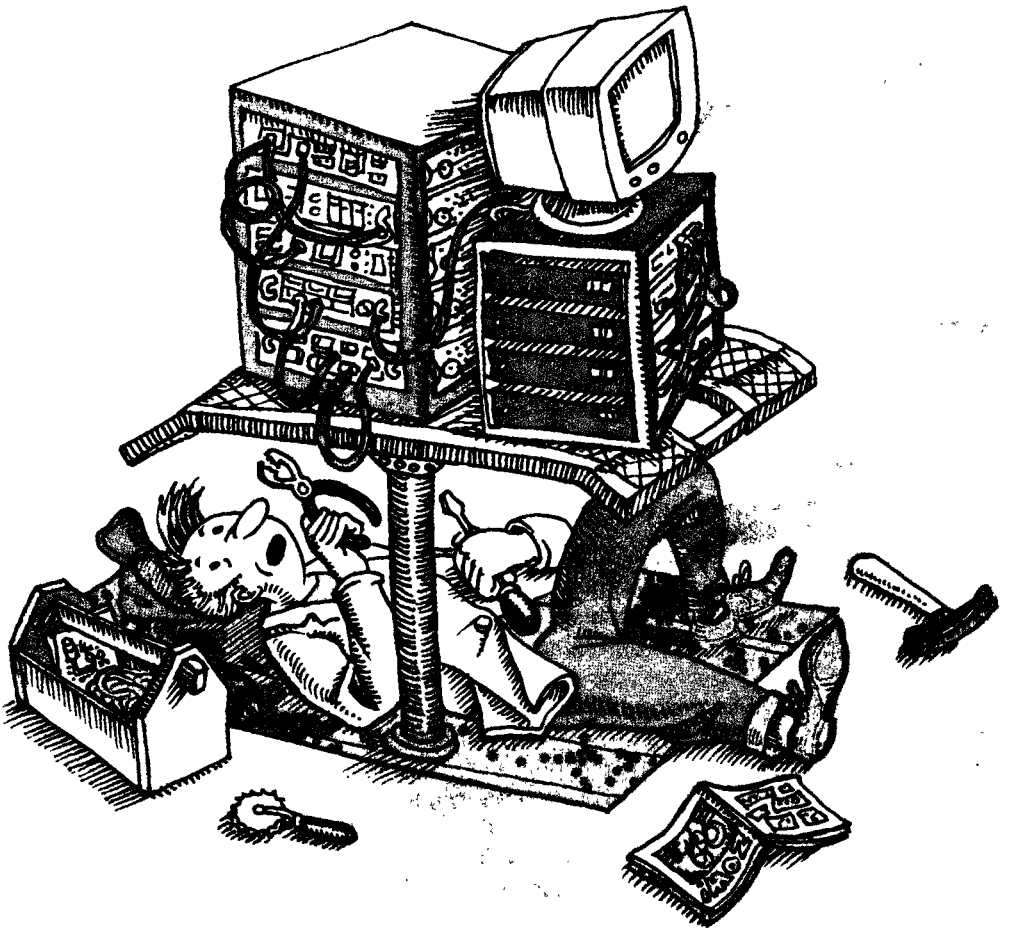
Адреса для писем из:

России: 127055, г. Москва, ул. Лесная, д. 43, стр. 1

Украины: 03150, Киев, а/я 152

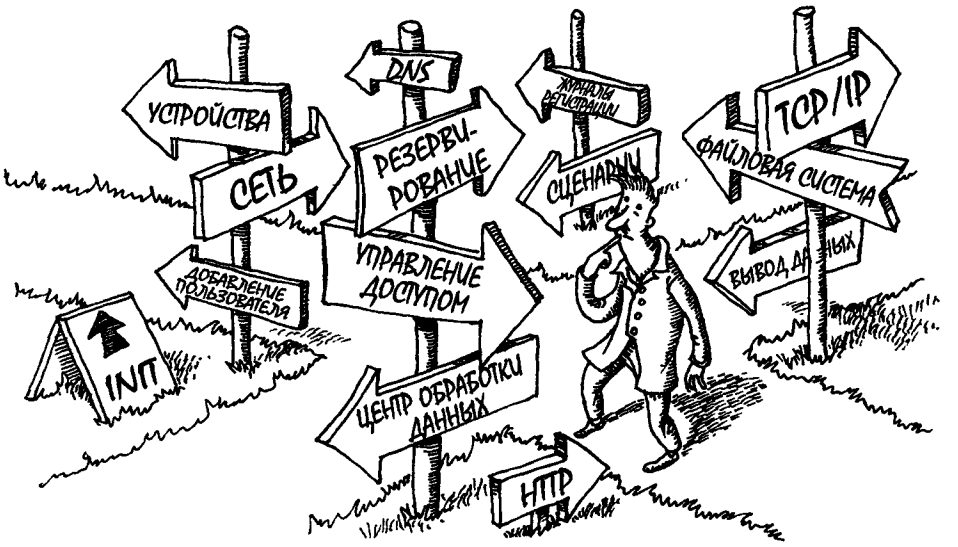
# ЧАСТЬ I

# Основы администрирования



# Глава 1

# С чего начать




В настоящее время в море информации по UNIX и Linux можно утонуть, поэтому мы решили с помощью этой книги проложить свой курс и удовлетворить потребности именно системных администраторов, заняв тем самым вполне конкретную нишу в экосистеме разного рода справочных страниц (man pages), блогов, журналов, книг и прочих справочных материалов.

Во-первых, это — руководство, в котором рассматриваются различные компоненты основных систем администрирования и принципы их совместной работы. Во многих случаях, когда нам приходилось выбирать между разными реализациями некоторой идеи, мы описывали преимущества и недостатки основных вариантов.

Во-вторых, это — краткий справочник, в котором собрано все, что необходимо знать для выполнения задач общего характера в различных версиях UNIX- и Linux-систем. Например, команда **ps**, отображающая статус текущих процессов, поддерживает более 80 ключей (опций) командной строки в системах Linux. Но всего несколько комбинаций ключей удовлетворяют 99% нужд системного администратора (см. раздел 5.7).

Наконец, в этой книге делается акцент на администрировании корпоративных серверов и сетей, т.е. на серьезной теме системного администрирования. Несложно установить операционную систему на отдельном компьютере, гораздо труднее поддерживать устойчивую работу виртуализированной сетевой среды при больших нагрузках, сбоях в работе дисков и умышленных атаках. Поэтому мы описываем методы и практические способы, которые позволят вам "поднимать" "упавшие" сети, а также выбирать решения, которые останутся актуальными при расширении и усложнении вашего сайта и изменении его гетерогенности.

Мы не претендуем на абсолютную степень объективности в решении перечисленных выше задач, поэтому по ходу изложения постарались максимально ясно пояснить свои

субъективные взгляды и предпочтения. Особенность системного администрирования заключается в том, что опытные администраторы могут иметь совершенно разные представления о правилах и процедурах управления системами. Читателям придется самостоятельно решать, какой именно материал и в какой степени соответствует той среде, в которой они работают.

# 1.1. Основные задачи системного администратора

🕮 Подробнее о написании сценариев рассказывается в главе 2.

В Википедии в статье "системный администратор" представлено довольно полное описание функций, которые, как общепринято считать, должен выполнять системный администратор. В английской версии этой статьи проводится четкое разграничение понятий администрирования и разработки программ, но мы знаем не понаслышке, сколько времени профессиональные администраторы отдают написанию сценариев (scripts). Сам по себе этот факт не превращает системных администраторов в разработчиков, но это означает, что им необходимо иметь практически такие же аналитические способности и знания в области архитектуры компьютерных сетей.

В этом разделе перечислены задачи, обычно возлагаемые на системного администратора. Совсем не обязательно, чтобы эти функции выполнял один человек. Во многих организациях работа поручается команде специалистов. В любом случае необходим хотя бы один человек, который понимал бы все поставленные задачи и обеспечивал их корректное выполнение другими людьми.

## Инициализация пользователей

🕮 Подробнее о добавлении новых пользователей рассказывается в главе 7.

В круг обязанностей системного администратора входит создание учетных записей для новых пользователей, удаление учетных записей тех пользователей, которые уже не работают в системе, и решение всех проблем, возникающих во время "системной жизни" своих подопечных (например, "подсказка" забытых паролей). Процесс управления записями можно автоматизировать, но ряд решений, связанных с включением в систему нового пользователя (место размещения его начального каталога, компьютер, на котором должна создаваться учетная запись, и так далее), должен принимать только администратор.

Если необходимо прекратить доступ пользователя к системе, его учетная запись должна быть аннулирована. Все файлы, относящиеся к этому пользователю, необходимо удалить, чтобы они не занимали место на диске.

## Подключение и удаление аппаратных средств

🕮 Дополнительная информация по данной теме приведена в главах 8, 13 и 26.

В случае приобретения новых аппаратных средств или подключения уже имеющихся устройств к другому компьютеру, нужно переконфигурировать систему таким образом, чтобы она распознавала и активизировала эти устройства. Изменение конфигурации может быть как простой задачей (например, подключение принтера), так и более сложной (например, подключение жесткого диска).

Теперь, когда в сферу корпоративной обработки данных прочно вошло понятие виртуализации, конфигурация технических средств может усложниться как никогда ранее. Инсталляция устройств может занимать несколько уровней стека виртуализации, и си-

стемному администратору приходится вырабатывать новые стратегии, которые позволят безопасно и качественно использовать общие аппаратные средства.

## Резервное копирование

🕮 Подробнее о резервном копировании рассказывается в главе 10.

Резервное копирование является, наверное, одной из наиболее важных задач системного администратора, которая, к сожалению, чаще всего игнорируется или выполняется спустя рукава. Процедура резервного копирования довольно утомительна и занимает много времени, но осуществлять ее необходимо. Этот процесс можно автоматизировать или поручить подчиненным, но все равно системный администратор обязан убедиться в том, что резервное копирование выполнено правильно и по графику (а с полученного носителя можно восстановить данные).

## Инсталляция и обновление программ

🕮 Подробнее об утилитах работы с программами рассказывается в главе 12.

После приобретения нового программного обеспечения его нужно инсталлировать и протестировать, часто в нескольких операционных системах и на различном оборудовании. Если программы работают нормально, пользователям необходимо сообщить об их наличии и местонахождении. Выпускаемые пакеты обновлений для исправления ошибок и устранения брешей в системе безопасности должны без проблем устанавливаться в локальных системах.

Локальные программы и административные сценарии следует соответствующим образом упаковать и организовать в виде, обеспечивающем совместимость с процедурами обновления, используемыми в системах вашего сайта. Новые версии ваших программных продуктов, прежде чем использовать их во всем узле сети, необходимо представлять для тестирования.

## Мониторинг системы

Крупные системы требуют неусыпного контроля. Не стоит надеяться, что пользователи всегда сами будут сообщать вам о затруднениях (если, конечно, они не столкнутся по-настоящему с серьезными проблемами). Обычно пользователи идут по пути наименьшего сопротивления, полагая, что на решение проблемы у них уйдет меньше времени, чем на ее описание и сообщение о ее возникновении.

Существует множество обязательных ежедневных операций, а именно: проверка правильности функционирования электронной почты и веб-служб; просмотр журнальных файлов на предмет наличия ранних признаков неисправностей; контроль над подключением локальных сетей; контроль доступности системных ресурсов (в частности, проверка наличия свободного места на диске). Все эти рутинные операции прекрасно поддаются автоматизации, да и множество готовых систем мониторинга могут помочь системным администраторам в решении этой задачи.

## Поиск неисправностей

Сбои систем неизбежны. Задача администратора — диагностировать сбои в системе и в случае необходимости вызвать специалистов. Как правило, найти неисправность бывает намного сложнее, чем устранить ее.

## Ведение локальной документации

Рекомендации, касающиеся ведения документации, даны в разделе 32.5.

При настройке конфигурации системы под конкретные требования очень скоро обнаруживается, что она значительно отличается от базовой конфигурации, которая описана в документации. А поскольку за реализацию этих настроек отвечает системный администратор, он должен документировать все изменения. Администратор должен также вести учет обслуживания всех аппаратных средств, регистрировать состояние резервных копий, документировать разводку сетевых кабелей и локальные правила работы в системе.

## Слежение за безопасностью системы

Вопросы безопасности рассматриваются в главе 22.

Системный администратор отвечает за реализацию стратегии защиты и должен периодически проверять, не нарушена ли безопасность системы. В системах с низким уровнем безопасности эта процедура может быть сведена к нескольким элементарным проверкам на предмет несанкционированного доступа. В системах с высоким уровнем безопасности обычно применяется сложная сеть ловушек и программ контроля.

## Оказание помощи пользователям

О необходимости оказания помощи пользователям в решении различных проблем редко упоминается в должностной инструкции системного администратора, хотя выполнение подобного рода обязанностей "съедает" большую часть его рабочего времени. Системных администраторов терроризируют самыми разными вопросами, начиная от "Вчера моя программа работала, а сегодня нет! Что вы поменяли?" до "Я пролила кофе на клавиатуру! Нужно ли теперь полить ее водой, чтобы смыть кофе?".

В большинстве случаев ваша реакция на все эти "сигналы тревоги" гораздо больше влияет на вашу оценку как администратора, чем реальные технические навыки, которыми, возможно, вы обладаете. Вы можете либо роптать на несправедливость такого положения вещей, либо радоваться тому, что успешным решением одной-единственной проблемы вы можете заслужить такую же благосклонность сотрудников (или начальства), как и пятичасовой отладкой в ночное время. Выбирайте!

# 1.2. Что необходимо знать

Мы предполагаем, что у читателей есть определенный опыт работы с Linux или UNIX. В частности, необходимо иметь общее представление о системе с точки зрения пользователя, поскольку мы не будем на этом останавливаться. Книги, перечисленные в разделе 1.14, помогут заложить необходимый фундамент знаний.

Даже в дни композитных оконных 3D-менеджеров (например, Compiz Fusion) GUI-инструменты для UNIX- и Linux-систем остаются довольно простыми по сравнению с навороченными базовыми программами. На практике большинство задач администрирования по-прежнему решается путем редактирования конфигурационных файлов и написания сценариев, поэтому читатели должны быть знакомы как с оболочкой командной строки, так и с каким-нибудь текстовым редактором.

Ваш редактор может иметь графический интерфейс (как у **gedit**) или работать в среде интерпретации командной строки (как **vi** или **emacs**). Такие текстовые процессоры,

как Microsoft Word и OpenOffice Writer, кардинально отличаются от упомянутых текстовых редакторов и практически бесполезны для решения задач администрирования. Командные редакторы имеют неоспоримое преимущество, поскольку они могут преодолевать простые SSH-соединения и работать на слабых системах, которые отказываются выполнять начальную загрузку; поэтому в оконных системах нет никакой необходимости. Кроме того, они действуют гораздо быстрее маленьких проворных редакторов, которые часто создают администраторы.

Рекомендуем изучить редактор **vi** (или его версию **vim**). Он является стандартным для всех UNIX- и Linux-систем и, хотя выглядит несколько "бледным" на фоне более мощных программ (в частности, **emacs**), абсолютно пригоден для работы. Нам также нравится GNU-редактор **nano**, который прост и прекрасно подходит новичкам, к тому же он оснащен экранными подсказками. Остерегайтесь нестандартных редакторов. Если отдать предпочтение одному из них, очень быстро надоест устанавливать его в каждой новой системе.

Одним из важнейших инструментов системного администратора являются сценарии для автоматизации основных задач. Примеры такого рода сценариев приводятся на протяжении всей книги. Для того чтобы стать профессионалом, необходимо научиться читать и модифицировать сценарии, написанные на языке интерпретатора **sh** (в Linux его эквивалент — интерпретатор **bash**).

🕮 Подробнее о написании сценариев рассказывается в главе 2.

Для новых проектов мы рекомендуем применять язык Perl или Python. Как язык программирования Perl несколько необычен, однако содержит много средств, необходимых администраторам. В качестве учебника по этому языку советуем прочесть книгу Ларри Уолла *Programming Perl*; она также представляет собой образец профессионального технического руководства. Библиографическое описание этой книги приведено в разделе 1.14.

Многие администраторы предпочитают иметь дело с языком Python. В определенном смысле это более элегантный язык, чем Perl, и его сценарии обычно легче читать и сопровождать. (Вот что сказал по этому поводу бывший сотрудник издательства Amazon Стив Йег: "Python уже давно стал пристанищем для тех, кто, наконец, принял красную пилюлю и освободился от Perl-матрицы".) Статьи, посвященные сравнению языка Python с другими языками (включая Perl), можно найти по следующему адресу: `www.python.org/doc/Comparisons.html`.

В большинство дистрибутивов сейчас входит Ruby (в переводе с англ. "рубин") — перспективный язык программирования, в котором сохранена мощность языка Perl и который при этом освобожден от его синтаксических "подводных камней". Но главное то, что он обладает современными объектно-ориентированными свойствами. Ruby еще не стал для системных администраторов традиционным языком написания сценариев, но, по всей вероятности, станет таковым в ближайшие несколько лет.

Мы рекомендуем также изучить **expect**, который представляет собой скорее интерфейс для управления интерактивными программами, а не язык программирования. По сути, это эффективная интегрирующая технология, которая может заменить традиционный процесс написания сценариев. Освоить **expect** можно достаточно быстро.

Все самое важное, что необходимо знать о написании сценариев на языках **bash**, **Perl** и **Python**, можно почерпнуть в главе 2. В ней также рассматриваются регулярные выражения (шаблоны, задающие правило поиска) и некоторые идиомы оболочки, полезные для системных администраторов.

## 1.3. Взаимосвязь систем Linux и UNIX

🕮 Подробнее об истории Linux и UNIX читайте в разделе "Краткая история системного администрирования".

В этой книге описываются как UNIX-, так и Linux-системы ввиду их схожести. Однако употребление названий Linux и UNIX в одном предложении чем-то сродни хождению по политическому минному полю или зыбучим пескам. Тем не менее мы возьмем на себя смелость максимально лаконично и объективно изложить ряд фактов. Однако, поскольку отношения между системами UNIX и Linux, мягко говоря, носят запутанный характер, мы не можем обойти молчанием создавшуюся ситуацию и возьмем на себя смелость максимально лаконично и объективно изложить ряд фактов.

Linux — измененная и усовершенствованная реализация ядра операционной системы UNIX. Она соответствует стандарту POSIX, работает на различных аппаратных платформах и совместима с большинством существующих приложений UNIX. Отличие от множества других (но не всех) вариантов UNIX заключается в том, что Linux — бесплатная операционная система с открытым исходным кодом, разрабатываемая коллективными усилиями тысяч энтузиастов и организаций. Вместе с тем, операционная система Linux содержит ряд технических усовершенствований, которые отсутствуют в UNIX, и поэтому представляет собой нечто большее, чем простой клон UNIX. В то же время традиционные производители UNIX не перестают заниматься усовершенствованием своих продуктов, что говорит о наличии областей, в которых коммерческие UNIX-системы превосходят Linux-системы.

Linux совершенно законно может считаться отдельным программным продуктом, который уже неправильно называть "UNIX" или версией "UNIX". В то же время было бы ошибкой настаивать на том, что Linux — совсем "не UNIX". Если ваше творение ходит как утка и крякает как утка, то, скорее всего, вы изобрели утку.

Раскольники нашлись даже в самом лагере Linux. Часто утверждают, и отчасти это справедливо, что ссылаться на дистрибутивы Linux просто как на "Linux" — значит не признавать труд разработчиков, вложенный в утилиты, выполняющиеся вне ядра (которые, по сути, составляют подавляющее большинство программ в средней системе). К сожалению, альтернативная версия, которую наиболее часто предлагают, — "GNU/Linux" — обременена собственным политическим багажом и официально поддерживается только дистрибутивом Debian. В статье Википедии "Спор об именовании GNU/Linux" приведены аргументы обеих сторон[1]. Интересно отметить факт уже преобладающего использования открытых программных средств в большинстве UNIX-систем, но ни для одной из них, насколько нам известно, пока не предложено использовать название "GNU/UNIX."[2]

Программы, написанные для Linux, являются UNIX-программами. Во многом благодаря проекту GNU наиболее значительные программные компоненты, составляющие основу UNIX-систем, разрабатываются теперь в соответствии с принципом открыто-

[1] Поскольку Википедия содержит информацию о Linux, а значит, она должна часто ссылаться на соответствующую статью, эта дискуссия имеет большое значение для самой Википедии. Рекомендуется также ознакомиться с содержимым вкладки Discussion для упомянутой статьи Википедии.

[2] В конце концов. "GNU — не UNIX"!

сти исходного кода.[3] Один и тот же код может выполняться как в Linux, так и в любой другой однотипной среде. Веб-серверу Apache, к примеру, все равно, где работать: Linux или Solaris. С позиции коммерческих приложений и большинства программ администрирования Linux — это просто одна из наиболее поддерживаемых и самых доступных разновидностей UNIX.

Следует заметить, что Linux — не единственная свободно доступная UNIX-подобная операционная система. OpenSolaris также имеет открытый исходный код, хотя ее строго оговоренные сроки лицензирования кажутся подозрительными для некоторых борцов за чистоту проектов с открытым кодом. Системы FreeBSD, NetBSD и OpenBSD, являющиеся потомками BSD UNIX, имеют своих пылких сторонников. Эти операционные системы сопоставимы с Linux по функциональным возможностям и надежности, хотя и не пользуются такой популярностью у сторонних разработчиков программного обеспечения.

Как UNIX-, так и Linux-системы уже многие годы эффективно функционируют в промышленных условиях.[4] Здесь выбор между ними должен делаться исходя из организации пакетирования, поддержки и корпоративных традиций, а не из реальных различий в качестве или современности.

В этой книге слово "Linux", как правило, относится к дистрибутивам Linux, а не к традиционным версиям UNIX. Но слово "UNIX" здесь употребляется не всегда в одном и том же значении, и иногда мы применяем его к атрибутам, используемым всеми производными от UNIX системами, включая Linux (например, "разрешения UNIX-файла"). Если описываемые детали касаются как UNIX, так и Linux-систем, то во избежание двусмысленности мы обычно так и пишем: "UNIX и Linux".

## 1.4. Дистрибутивы Linux

Все дистрибутивы основаны на едином семействе ядер, однако набор служебных программ, дополняющих ядро, может существенно варьироваться. Дистрибутивы различаются по своему назначению, наличию служб поддержки и степени популярности. В настоящее время существуют сотни независимых дистрибутивов Linux, но мы считаем, что дистрибутивы из семейств Debian, Red Hat и SUSE будут доминировать в производственных средах еще как минимум в течение ближайших пяти лет.

В целом между дистрибутивами Linux нет огромных различий. Остается загадкой: зачем столько? Причем отличительными свойствами каждого из них являются "простота инсталляции" и "внушительная библиотека программных средств". Нет-нет да и проскакивает мысль о том, что кто-то просто испытывает удовлетворение от выпуска новых дистрибутивов Linux.

На удивление, многие другие (более компактные) дистрибутивы так же конкурентоспособны по возможностям настройки. Все значительные дистрибутивы (в том числе

[3] Некоторые наши технические рецензенты не соглашались с тем, что мы, как им казалось, связывали GNU с созданием самого открытого в мире программного обеспечения. Отнюдь нет! Однако участники проекта GNU определенно сделали больше, чем любая другая группа для распространения идеи открытого программного обеспечения как социальной инициативы и систематизации непрекращающегося спора о сроках лицензирования и взаимодействии между открытыми и неоткрытыми программными продуктами.

[4] Под "производственной" средой мы понимаем среду, которую организация использует для выполнения реальных задач (в отличие от использования для тестирования, исследований или разработки).

и второго яруса) включают относительно безболезненную процедуру инсталляции, изящную среду, реализующую концепцию рабочего стола, и форму управления пакетированием. Кроме того, большинство дистрибутивов позволяет выполнять загрузку с DVD-диска, что удобно для отладки и ознакомления с новым продуктом.

Поскольку эта книга сфокусирована на управлении крупномасштабными Linux-системами, мы больше склоняемся к дистрибутивам наподобие Red Hat, которые предусматривают управление сетями. Одни дистрибутивы создавались в расчете на промышленное применение, другие — нет. Какой-нибудь дополнительный программный компонент системы промышленного уровня, на первый взгляд малозначительный, в действительности существенно упрощает администрирование.

Принимая решение в пользу того или иного дистрибутива, обращайте внимание не только на его функциональные возможности, но и на то, как будет протекать сотрудничество вашей организации и поставщика дистрибутива в ближайшие годы.

Задайте себе ряд важных вопросов.

- Будет ли дистрибутив существовать в ближайшие пять лет?
- Будут ли оперативно устраняться бреши в системе защиты?
- Будут ли оперативно выпускаться новые версии программных продуктов?
- Будет ли оказана помощь в решении возникших проблем?

Под этим углом зрения некоторые чрезвычайно интересные, компактные дистрибутивы уже не кажутся столь привлекательными. Но не стоит сбрасывать их со счетов: например такая компания, как E*Trade, работает в системе Gentoo Linux.

Наиболее жизнеспособные дистрибутивы не обязательно имеют корпоративный статус. Например, мы ожидаем, что система Debian Linux (простите, Debian GNU/Linux!) просуществует довольно долго, несмотря на то что Debian — это не коммерческая компания, она ничего не продает и не предлагает сервисного обслуживания. Несмотря на то что Debian не входит в число широко используемых дистрибутивов, эта система извлекает пользу из группы спонсоров и огромной популярности дистрибутива Ubuntu, который основан на системе Debian.

Наиболее популярные дистрибутивы общего назначения перечислены в табл. 1.1.

**Таблица 1.1. Наиболее популярные дистрибутивы Linux общего назначения**

| Дистрибутив | Веб-узел | Комментарии |
|---|---|---|
| CentOS | `centos.org` | Бесплатный аналог Red Hat Enterprise Linux |
| Debian | `debian.org` | Популярный некоммерческий дистрибутив |
| Fedora | `fedoraproject.org` | Вариант Red Hat Linux, предназначенный для использования отдельными пользователями |
| Gentoo | `gentoo.org` | Оптимизированный дистрибутив, предназначенный для самостоятельной компиляции |
| Linux Mint | `linuxmint.com` | Компактный дистрибутив, основанный на Ubuntu |
| Mandriva | `mandriva.com` | Один из наиболее дружественных по отношению к пользователю дистрибутивов |
| openSUSE | `opensuse.org` | Бесплатный аналог SUSE Linux Enterprise |
| Oracle Enterprise Linux | `oracle.com` | Версия RHEL, поддерживаемая системой Oracle |
| PCLinuxOS | `pclinuxos.com` | Ответвление от Mandriva, ориентированное на KDE |
| Red Flag | `redflag-linux.com` | Ведущий дистрибутив в Китае, подобный Red Hat |

*Окончание табл. 1.1*

| Дистрибутив | Веб-узел | Комментарии |
|---|---|---|
| Red Hat Enterprise | `redhat.com` | Коммерческий дистрибутив Red Hat, отличающийся высокой надежностью |
| Slackware | `slackware.com` | Один из старейших дистрибутивов |
| SUSE Linux Enterprise | `novell.com/linux` | Многоязыковой дистрибутив, особенно популярен в Европе |
| Ubuntu | `ubuntu.com` | Доработанная версия Debian |

Обновляемый перечень дистрибутивов Linux (в том числе и не англоязычных) можно найти по следующим адресам: `linux.org/dist`, `lwn.net/Distributions` и `distrowatch.com`.

## 1.5. Примеры систем, используемых в этой книге

В этой книге мы остановились на трех популярных дистрибутивах Linux и трех версиях UNIX: Ubuntu Linux, openSUSE, Red Hat Enterprise Linux, Solaris, HP-UX и AIX. Они представляют собой срез рынка корпоративных систем и охватывают большинство инсталляций в крупных организациях.

Приведенная в книге информация относится, как правило, ко всем упомянутым выше дистрибутивам, если нет специальных уточнений. Подробности, касающиеся конкретного дистрибутива, помечаются значком поставщика.


Ubuntu® 9.10 "Karmic Koala"


openSUSE® 11.2


Red Hat® Enterprise Linux® 5.5


Solaris™ 11 and OpenSolaris™ 2009.06


HP-UX® 11i v3


AIX® 6.1

Эти значки использованы с разрешения их владельцев. Следует заметить, что разработчики приведенных дистрибутивов не рецензировали эту книгу. Подробнее о каждой из упомянутых систем описано в следующих разделах.

### Примеры Linux-дистрибутивов


Информация, относящаяся именно к Linux-системам, но не к какому-то конкретному дистрибутиву, отмечается логотипом с изображением талисмана Linux — пингвина Такса (Tux).


Дистрибутивы Ubuntu сохраняют идеологическую направленность на разработку членами сообщества пользователей и разработчиков и открытый доступ,

поэтому вопрос о том, какие части дистрибутива бесплатны или разрешены для дальнейшего распространения, даже не возникает. Ubuntu в настоящее время финансируется за счет благотворительных пожертвований южноафриканского предпринимателя Мака Шаттлворта (Mark Shuttleworth).

Ubuntu основан на дистрибутиве Debian и использует его систему пакетирования. Он выпускается в двух основных вариантах: Desktop Edition и Server Edition. В сущности, они идентичны, но ядро Server Edition предварительно настроено для использования на сервере и не инсталлирует графический интерфейс (GUI) или такие GUI-приложения, как OpenOffice.

Компания SUSE, которая теперь является одним из подразделений корпорации Novell, пошла по пути Red Hat и начала распространять два связанных дистрибутива: openSUSE, который содержит только бесплатное программное обеспечение, и платный SUSE Linux Enterprise, который включает средства формальной поддержки и предоставляет несколько дополнительных возможностей. Эта книга не содержит никакой информации, характерной для того или иного дистрибутива SUSE, поэтому мы называем их всех просто "SUSE".

В течение более десяти последних лет Red Hat занимает ведущее положение среди вариантов Linux, и его дистрибутивы наиболее популярны в Северной Америке. В 2003 году первоначальный дистрибутив Red Hat Linux был разделен на серию версий, ориентированных на производственные среды, которые получили название Red Hat Enterprise Linux (в этой книге мы иногда употребляем аббревиатуру RHEL), и на версии, разрабатываемые в рамках проекта с привлечением всех членов сообщества пользователей и разработчиков, который получил название Fedora. Это разделение было обусловлено рядом технических, экономических, логических и юридических причин.

Сначала эти дистрибутивы были схожими, но за последние пять лет Fedora претерпела значительные изменения, и теперь эти две системы не синхронизированы. RHEL отличается высоким уровнем поддержки и стабильностью, но ее по существу невозможно использовать, не приобретя лицензию в компании Red Hat.

Проект CentOS (`centos.org`) концентрирует исходный код, который компания Red Hat обязана распространять в соответствии с различными лицензионными соглашениями (наиболее значимой является общедоступная лицензия GNU), и комплектует их в законченный дистрибутив, который во многом подобен дистрибутиву Red Hat Enterprise Linux, но доступен бесплатно. Этот дистрибутив лишен торговой марки компании Red Hat и не содержит некоторых запатентованных программных средств, но в остальном аналогичен платной версии. CentOS стремится к полной совместимости с RHEL, как по двоичному коду, так и по степени исправления ошибок.

Этот дистрибутив — прекрасный выбор для тех сайтов, которые стремятся развернуть крупномасштабную производственную систему, не платя при этом "десятину" компании Red Hat. Возможен также смешанный подход: сетевые серверы могут работать под управлением Red Hat Enterprise Linux, используя при этом преимущества, предоставляемые прекрасным уровнем поддержки со стороны Red Hat, а рабочие станции могут функционировать под управлением CentOS. Такой подход обеспечивает оптимизацию с точки зрения риска и поддержки, в то же время сводя к минимуму затраты и сложность администрирования.

## Примеры UNIX-дистрибутивов

Solaris — операционная система, построенная на основе System V и "обросшая" множеством расширений. Разработана компанией Sun Microsystems, которая ныне принадлежит корпорации Oracle.[5] Изначально Sun UNIX (так эта система называлась в средине 80-х гг.) — потомок Berkeley UNIX, но в результате корпоративного партнерства Sun и AT&T в системе была изменена кодовая основа. Solaris работает на различных аппаратных платформах, в частности на Intel x86 и SPARC.

Под "крышей" компании Sun систему Solaris можно было свободно загружать и использовать. Но корпорация Oracle изменила условия лицензирования, и теперь для загружаемых версий устанавливается бесплатный пробный период (90 дней). Наличие же OpenSolaris (свободно распространяемой версии Solaris с открытым исходным кодом) явно усугубляет сложившуюся ситуацию. На момент написания книги (средина 2010 г.) планы Oracle в отношении Solaris и OpenSolaris были неясны.

Выпуск системы Solaris 11, очень похожей на систему OpenSolaris, ожидался уже в 2010 году. В этой книге, употребляя слово "Solaris", мы имеем в виду смешанную систему, основанную на выпусках Solaris 10 и OpenSolaris (все данные получены от нашей разветвленной сети сильно засекреченных шпионов, действующих в недрах Oracle). В определенных случаях при необходимости мы конкретно указываем названия Solaris 10 или OpenSolaris.

Система HP-UX основана на System V и привязана к аппаратным платформам Hewlett-Packard. По исходному коду она ближе к родительскому генеалогическому древу, чем Solaris илиAIX, но HP сохранила темп ведения разработок в мире операционных систем и расширила круг собственных усовершенствований. Теперь, когда HP начала поддержку и Linux-систем, будущее HP-UX несколько затуманено.

AIX — операционная система компании IBM, изначально построенная на базе BSD 4.2 (Berkeley UNIX), но уже в версии AIX 4 (1994 г.) большинство ее компонентов перешло в System V. В настоящее время систему AIX "отнесло" довольно далеко от обоих источников.

В целом, у нас создалось впечатление, что перекрестное опыление от других систем характерно для AIX в меньшей степени, чем для большинства вариантов UNIX. Вместе с тем мы считаем, что эта система попала под "дьявольское" влияние некоторых мэйнфреймовых и среднего класса (AS/400) операционных систем IBM, от которых она унаследовала такие компоненты, как Object Data Manager (см. раздел 13.6), команды конфигурации (вместо файлов конфигурации) и административный интерфейс SMIT. Со временем эта система, возможно, станет более самобытной.

Интересно отметить, что в последние десять лет IBM демонстрирует ОС-инвариантный (т.е. независимый от операционной системы) подход к маркетингу своих аппаратных средств. IBM, продолжая разработку и продвижение AIX, также заинтересована в сотрудничестве с Red Hat и Novell, чтобы убедиться в устойчивой работе их дистрибутивов Linux на оборудовании IBM. Посмотрим, как этот подход проявит себя в будущем.

[5] Подробнее о системах BSD, System V и истории возникновения UNIX читайте в разделе "Краткая история системного администрирования".

## 1.6. Административные средства конкретных систем

Современные системы включают графические средства и панели управления (такие, как YaST2 в SUSE и SMIT в IBM), упрощающие конфигурирование и администрирование определенных компонентов системы. Эти средства очень удобны, особенно администраторам-новичкам, но они недостаточно полно отражают возможности базового программного обеспечения.

В этой книге мы описываем низкоуровневые механизмы, а не высокоуровневые инструментальные средства. На то есть ряд причин. Во-первых, графические средства зачастую являются платными или по крайней мере зависят от используемой системы. В результате то, что на низком уровне является одинаковым для всех систем, начинает казаться хаотичным и непостоянным. Во-вторых, мы считаем, что для системного администратора важно четко знать, как работает система. Когда в системе происходит сбой, графические утилиты не всегда помогают найти и устранить проблему. Наконец, конфигурировать систему на низком уровне обычно удобнее: так быстрее и надежнее, больше возможностей для манипуляции, легче писать сценарии.

## 1.7. Типографские особенности книги

Имена файлов, команды и аргументы команд, которые следует набирать на клавиатуре без изменений, даны полужирным шрифтом. Аргументы, вместо которых следует подставлять конкретные значения, даны курсивом. Например, в команде

**cp** *файл каталог*

предполагается, что аргумент *файл* следует заменить именем реального файла, а аргумент *каталог* — именем реального каталога.

Результаты работы команд, а также фрагменты сценариев и файлов конфигурации даны моноширинным шрифтом. Комментарии к интерактивным сеансам иногда даются курсивом, например:

```
% grep Bob /pub/phonelist     # Найти номер телефона Боба
Bob Knowles 555-2834
Bob Smith 555-2311
```

При описании синтаксиса команд мы, как правило, используем те же обозначения, что и в интерактивном руководстве:

- текст, заключенный в квадратные скобки ("[" и "]"), является необязательным;
- текст, после которого стоит многоточие ("..."), можно повторять;
- фигурные скобки ("{" и "}") указывают на то, что необходимо выбрать один из элементов, разделенных вертикальной чертой ("|").

Например, спецификации

**bork** [**-x**] {**on**|**off**} *имя_файла*...

соответствует любая из следующих команд:

```
bork on /etc/passwd
bork -x off /etc/passwd /etc/termcap
bork off /usr/lib/tmac
```

В выражениях с шаблонами поиска используются следующие метасимволы:

- звездочка (*) обозначает нуль или более символов;
- знак вопроса (?) обозначает один символ;

- тильда (~) обозначает начальный каталог текущего пользователя[6];
- выражение *~пользователь* обозначает начальный каталог указанного пользователя.

Например, иногда мы обозначаем каталоги, где хранятся сценарии запуска (`etc/rc*.d`, `etc/rc*.d` и т.д.), сокращенным шаблоном `etc/rc*.d`.

# 1.8. Единицы измерения

Такие метрические приставки, как кило-, мега- и гига-, определяются показателями степени числа 10 (например, один мегагерц составляет тысячу килогерц или миллион герц). Эти приставки "позаимствованы" и для компьютерных типов данных, но с использованием степеней числа 2. Например, один "мегабайт" памяти составляет $2^{20}$ или 1 048 576 байт. Ассоциация твердотельных технологий (Solid State Technology Association) Объединенного инженерного совета по электронным устройствам (Joint Electronic Device Engineering Council — JEDEC) даже превратила эти "позаимствованные" единицы измерения в такой официальный стандарт, как Standard 100B.01, который признает их степенями двойки (не без некоторых натяжек).

Пытаясь восстановить справедливость (или внести ясность), Международная электротехническая комиссия (International Electrotechnical Commission — IEC) определила ряд числовых приставок, основанных именно на степенях числа 2: киби- (kibi-), меби- (mebi-), гиби- (gibi-) и так далее с аббревиатурами КиБ (Ki), МиБ (Mi) и ГиБ (Gi) соответственно. С вводом этих единиц измерения ликвидируется двузначность, но их только начинают использовать. Поэтому привычный набор кило-, мега- и гига-префиксов все еще в ходу (в обоих значениях: десятичном и двоичном).

Определить конкретное значение можно по контексту. Объем ОЗУ всегда измеряется в степенях двойки, но пропускная способность сети — в степенях числа 10. Размер дисков их производители обычно указывают в десятичных единицах, а размер блоков и страниц — в двоичных.

В этой книге мы используем единицы измерения МЭК (IEC) для степеней двойки и метрические — для степеней числа 10. Метрические единицы измерения мы применяем также для приблизительных значений и в тех случаях, когда точное основание степени неясно, недокументировано или его невозможно определить. В командах файлов конфигурации `output` и `in` мы оставляем исходные значения и указатели единиц измерений. Для обозначения бита служит буква *b*, а для байта — буква *B*. Некоторые примеры интерпретации единиц измерения приведены в табл. 1.2.

**Таблица 1.2. Примеры интерпретации единиц измерения**

| Пример | Описание возможного варианта |
|---|---|
| 56 кбит/с (kb/s) | Скорость передачи данных по последовательной линии связи, равная 56 000 бит в секунду |
| 1 Кбайт (kB) | Размер файла, равный 1000 байт |
| 4 Кибайт (KiB) | Общий размер страниц полупроводникового диска (SSD) 4 096 байт |
| 8 Кбайт (KB) | Размер памяти — не используется в этой книге (см. описание ниже) |
| 100 Мбайт (MB) | Ограничение по размеру файла (неоднозначность, понимается по контексту) номинально составляет $10^8$ байт |

[6] В дистрибутиве Solaris 10 в качестве стандартной оболочки для пользователя root используется оригинальная оболочка Bourne, которая (что удивительно) не воспринимает метасимвол ~ или *~пользователь*.

*Окончание табл. 1.2*

| Пример | Описание возможного варианта |
|---|---|
| 100 Мбайт (MB) | Размер раздела диска (понимается по контексту) номинально составляет $10^8$ байт, возможно, 99 999 744 байт[a] |
| 1 Гибайт (GiB) | Размер ОЗУ, в точности равный 1 073 741 824 байт[б] |
| 1 Гбит/с (Gb/s) | Скорость передачи информации в сети Ethernet составляет 1 гигабит в секунду (1 000 000 000 бит в секунду) |
| 1 Тбайт (TB) | Объем жесткого диска составляет 1 терабайт (один триллион байт), или 1 000 000 000 000 байт |

[a] Число $10^8$ округлено в меньшую сторону до ближайшего целого, кратного размеру 512-байтового блока диска.

[б] По заявлению компании Microsoft, этого недостаточно для функционирования 64-разрядной версии Windows 7.

Аббревиатура *K*, как в случае "8 Кбайт (KB)", не является частью стандарта. Это компьютерная адаптация метрической аббревиатуры *к* (обозначение приставки *кило-*), которая означает 1 024 в отличие от 1 000. Но поскольку в аббревиатурах для многих метрических приставок уже используется прописная буква, стала возникать путаница при новом и исходном использовании буквы *K* в качестве множителя 1 000.

В дистрибутиве Ubuntu Linux реализуется героическая попытка внести здравый смысл в этот вопрос и ликвидировать противоречивость (подробнее см. `wiki.ubuntu.com/UnitsPolicy`).

# 1.9. Использование справочных страниц и другой оперативной документации

Справочные страницы онлайн-руководства (так называемые man-страницы, поскольку их можно просмотреть с помощью команды `man`) составляют традиционную "интерактивную" документацию (хотя, конечно, в настоящее время практически вся документация в той или иной мере является интерактивной). Эти материалы в основном инсталлируются вместе с системой, а справочной страницы отдельных программ — вместе с соответствующим пакетом.

Интерактивное руководство содержит краткое описание отдельных команд, драйверов, форматов файлов и библиотечных функций. В них не найти ответа на общие вопросы, например, "Как инсталлировать новое устройство?" или "Почему моя система работает так медленно?". Для получения ответов на подобные вопросы обращайтесь к руководствам по администрированию систем соответствующих производителей (см. раздел 1.10) или, если вопрос касается систем Linux, к документам, находящимся в ведении проекта Linux Documentation Project (LDP).

## Организация справочных страниц интерактивного руководства

Во всех системах справочные страницы обычно делятся на разделы, при этом возможны небольшие вариации на тему принципов такого разделения. Основная схема разделения, используемая в наших примерах систем, показана в табл. 1.3.

**Таблица 1.3. Разделы справочных страниц**

| Linux | Solaris | HP-UX | AIX | Содержание |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | Команды пользовательского уровня и приложения |
| 2 | 2 | 2 | 2 | Системные вызовы и коды ошибок ядра |
| 3 | 3 | 3 | 3 | Библиотечные функции |
| 4 | 7 | 7 | 4 | Драйверы устройств и сетевые протоколы |
| 5 | 4 | 4 | 5 | Стандартные форматы файлов |
| 6 | 6 | – | 6 | Игры и демонстрационные программы |
| 7 | 5 | 5 | 7 | Различные файлы и документы |
| 8 | 1m | 1m | 8 | Команды системного администрирования |
| 9 | 9 | – | – | Внутренние интерфейсы и спецификации ядра |
| – | – | 9 | – | Общая информация по HP-UX |

Некоторые разделы делятся на подразделы. Например, подраздел 3c руководства по Solaris содержит страницы с информацией о библиотеке стандартных C-функций системы. В некоторых системах раздел 8 остается пустым, а описание команд, относящихся к системному администрированию, как правило, содержится в разделе 1. Игры и демонстрационные программы во многих системах уже удалены из раздела 6. Раздел руководства под названием "l" (прописная буква от "L") зачастую отводится для локальных справочных страниц.

Точная структура разделов не так уж важна, поскольку команда **man** находит соответствующую страницу в большинстве случаев. При этом в нескольких разделах может быть одно и то же имя, и вот тут важно указать нужный вам раздел. Например, **passwd** — это и команда, и файл конфигурации, поэтому соответствующие статьи есть как в разделе 1, так и разделе 4 или 5.

## Команда man: чтение страниц интерактивного руководства

Команда **man** *заголовок* форматирует конкретную страницу интерактивного руководства и выводит ее на терминал пользователя посредством **more**, **less** или другой программы постраничной разбивки, которая задана в переменной среды PAGER. Аргумент *заголовок* — это, как правило, имя команды, устройства или файла, о которых необходимо получить справочную информацию. Поиск по разделам руководства осуществляется в порядке возрастания номеров, хотя разделы, посвященные описанию команд (1, 8 и 6), обычно просматриваются в первую очередь.

Команда **man** *раздел заголовок* вызывает справочную страницу из указанного раздела. Так, в большинстве систем команда **man** sync вызывает справочную страницу для команды **sync**, а команда **man 2 sync** — для системного вызова **sync**.


В системе Solaris необходимо перед номером раздела использовать флаг **-s**, например **man -s 2 sync**.

Команда **man -k** *ключевое_слово* или **apropos** *ключевое_слово* выводит список справочных страниц, в строке пояснений к которым имеется указанное ключевое слово.

```
% man -k translate
objcopy (1)          - copy and translate object files
dcgettext (3)        - translate message
```

```
tr (1)            - translate or delete characters
snmptranslate (1) - translate SNMP OID values into more useful information
tr (1p)           - translate characters
...
```

База данных ключевых слов может устаревать. При добавлении новых справочных страниц к системе вам, возможно, придется перестроить (перекомпоновать) этот файл с помощью команд **mandb** (Ubuntu, SUSE), **makewhatis** (Red Hat) или **catman -w** (Solaris, HP-UX, AIX).

## Хранение страниц интерактивного руководства

Неформатированная информация для справочных страниц (входные данные команды **nroff**) обычно хранится в подкаталогах каталога **/usr/share/man**. В целях экономии места на диске системы Linux сжимают страницы с помощью утилиты **gzip** (команда **man** может очень быстро разархивировать их). Команда **man** поддерживает кеш отформатированных страниц в каталогах **/var/cache/man** или **/usr/share/man**, если соответствующие каталоги доступны для записи, но эти операции рискованны с точки зрения безопасности. В большинстве систем предварительное форматирование справочных страниц выполняется однократно во время инсталляции (см. команду **catman**) или не выполняется совсем.

В дополнение к традиционной команде **nroff** система Solaris воспринимает справочные страницы, отформатированные с использованием обобщенного языка разметки SGML. SGML-страницы имеют собственные каталоги раздела в каталоге **/usr/share/man**.

Команда **man** ищет страницы в ряде каталогов. В Linux-системах выяснить путь поиска позволяет команда **manpath**. Результат ее работы (в системе Ubuntu) обычно таков.

```
ubuntu$ manpath
/usr/local/man:/usr/local/share/man:/usr/share/man
```

Эта установка хранится в переменной среды MANPATH, и в случае необходимости ее можно изменить.

```
export MANPATH=/home/share/localman:/usr/share/man
```

Некоторые системы позволяют установить общесистемный параметр пути поиска для справочных страниц, который может оказаться полезным в случае, если вам придется поддерживать параллельное дерево справочных страниц, например, сгенерированных кроссплатформенным менеджером пакетов OpenPKG. Но если вы хотите распространять локальную документацию в виде справочных страниц, проще всего использовать стандартный системный механизм пакетирования и выложить справочных страницы в стандартные справочные каталоги. Подробнее об этом написано в главе 12.

## GNU-система Texinfo

Системы Linux включают дополнительный вариант интерактивных справочных страниц, именуемый Texinfo. Система Texinfo была изобретена энтузиастами проекта GNU после того, как выяснилось, что команда **nroff**, применяемая для форматирования справочных страниц, является собственностью AT&T. Теперь то же самое делает GNU-команда **groff**, так что проблема утратила актуальность, хотя Texinfo все еще бродит как зомби в поисках человеческих мозгов.

Несмотря на то что использование системы Texinfo, казалось бы, постепенно сходит на нет, некоторые GNU-пакеты требуют, чтобы документация к ним была представлена именно в формате Texinfo. Чтобы обойти встроенную навигационную систему Texinfo, перенаправьте поток вывода команды **info** посредством команды **less**.

Но есть и приятный момент. Пакеты, документируемые с помощью системы Texinfo, обычно инсталлируют страницы-заглушки, в которых сообщается о том, что информацию о соответствующем пакете можно получить, вызвав команду **info**. Таким образом, можно продолжать пользоваться командой **man**, обращаясь к команде **info** лишь в случае необходимости. Команда **info info** выдает справку по запутанной системе Texinfo.

# 1.10. Другие виды официальной документации

Справочные страницы — это лишь малая часть официальной документации. Остальная, в основном, рассеяна в веб-пространстве.

## Руководства по конкретным системам

Одни производители систем ведут собственные онлайн-проекты по подготовке документации, другие выпускают полезные руководства в виде объемных книг. В настоящее время нужное руководство быстрее найти в Интернете, чем в форме печатного издания. Объем и качество документации бывают разными, но большинство производителей выпускает, по меньшей мере, руководство по администрированию и инсталляции системы. Где найти документацию по каждому из наших примеров систем, показано в табл. 1.4.


Безусловно, нельзя не выделить компанию IBM, которая выпускает множество толстых книг по различным темам администрирования систем. Вы можете купить их или даже бесплатно скачать на свой компьютер. Но, к сожалению, многие документы от IBM отстают на одну или две версии от текущего выпуска AIX.

**Таблица 1.4. Где найти документацию от производителей операционных систем**

| Система | URL | Описание |
|---|---|---|
| Ubuntu | `help.ubuntu.com` | Наиболее дружественная к пользователю; см. раздел "server guide" |
| SUSE | `novell.com/documentation` | Материал по администрированию собран в разделе "reference guide" |
| RHEL | `redhat.com/docs` | В основном, документы по расширениям системы Red Hat |
| Solaris | `docs.sun.com` | Громадный каталог материалов |
| HP-UX | `docs.hp.com` | Книги, технические описания и руководства |
| AIX | `www.redbooks.ibm.com`<br>`ibm.com/support` | Многочисленные книги в формате PDF<br>Доступ к комментариям, разделам FAQs и пр. |

В "соревнованиях" по созданию документации система Red Hat, к сожалению, отстает от лидеров. Большая часть выпускаемых документов касается не общих вопросов администрирования Linux-систем, а собственных систем с дополнительными позитивными характеристиками.

## Документация по конкретным пакетам

Большинство важнейших программных пакетов в мире UNIX и Linux поддерживают отдельные лица или такие сторонние организации, как Internet Systems Consortium и Apache Software Foundation. Сотрудники этих групп пишут собственную документацию, качество которой часто оставляет желать лучшего. При этом существуют и такие прекрасно выполненные документы, как Version Control with Subversion от `svnbook.red-bean.com` (оказывается, надо знать, где искать!).

Производители систем UNIX и дистрибуторы систем Linux всегда включают в свои пакеты соответствующие справочные страницы. К сожалению, многие из них стараются сэкономить на документации другого типа, главным образом, ввиду отсутствия стандартного места для ее представления (см. `/usr/share/doc`). Всегда имеет смысл посетить веб-сайт первоисточника программного продукта и поинтересоваться наличием дополнительных материалов и доступа к ним.

В число дополнительных документов входят: технические отчеты (техническое описание), проектные обоснования и трактовки конкретных тем. Эти дополнительные материалы зачастую не ограничиваются простым описанием команд и могут включать самоучители и другие разделы. Многие программные пакеты, помимо справочных страниц, содержат и статьи по соответствующим темам. Например, после вызова справочной страницы для редактора vi вам будет предложено не только ознакомиться с аргументами командной строки, но и перейти к разделу, из которого вы узнаете, как в действительности отредактировать файл.

## Книги

К самым лучшим источникам информации для системных администраторов в книжном мире (в дополнение к этой книге :-), конечно) можно отнести серию книг Тима О'Рейли (Tim O'Reilly). Начало этой серии было положено более 20 лет назад книгой *UNIX in a Nutshell*. Теперь же практически всем важным подсистемам UNIX и Linux (и даже командам) посвящены ее отдельные тома. Эта серия также включает книги по Интернету, Windows и другим темам, не связанным с UNIX. Все эти книги имеют приемлемую цену, своевременны и ориентированы на конкретную аудиторию.

Тим О'Рейли, заинтересовавшись движением, основанным на принципе открытого исходного кода, регулярно проводит конференции (OSCON) по этой и другим злободневным темам. Конференции OSCON проводятся дважды в год (по одной в США и Европе). Подробнее см. на сайте `oreilly.com`.

## Документы RFC и другие интернет-ресурсы

Документы из серии RFC (Request for Comments — запрос на комментарий) описывают протоколы и процедуры, используемые в Интернете. Большинство из этих документов достаточно детализированы и специализированы, но некоторые написаны в виде обзоров. Информации, которую они содержат, вполне можно доверять, и она во многом оказывается полезной для системных администраторов. Более полное описание этих документов можно найти в разделе 14.1.

## Проект документации по Linux

Для систем Linux существует и такой источник справочной информации, как проект Linux Documentation Project (LDP), доступный на веб-сайте `tldp.org`. Этот сайт пред-

ставляет собой центральное хранилище разного рода информации, посвященной Linux: от FAQ (часто задаваемые вопросы) до объемных руководств. Здесь же концентрируются результаты усилий по переводу документации на различные языки.

К сожалению, многие LDP-документы практически не обновляются, поэтому быстро устаревают. Всегда обращайте внимание на дату публикации документа, так как по ней можно судить об актуальности информации.

# 1.11. Другие источники информации

Источники информации, рассмотренные в предыдущем разделе, можно охарактеризовать как самые надежные, но вряд ли это последнее слово в документации по UNIX и Linux. В Интернете же можно найти многочисленные блоги, дискуссионные форумы и новостные веб-каналы и получить с их помощью самую "свежую" информацию.

Как и следовало ожидать, система Google стала лучшим другом системного администратора. Именно к Google многие администраторы обращаются за консультацией, если вопрос не касается спецификации конкретной команды или формата файла (но и в этом случае Google обязательно поможет). Возьмите себе за правило (если не хотите впустую терять время и репутацию) при возникновении затруднений выносить свои вопросы на онлайн-форумы, ссылающиеся при ответах на Google.[7]

Невозможно перечислить все полезные интернет-источники информации по Linux, поэтому упомянем лишь важнейшие из них (табл. 1.5).

Можем порекомендовать еще один забавный и полезный ресурс. Это страница "Rosetta Stone" (розеттский камень) Брюса Гамильтона (Bruce Hamilton), доступная по адресу `bhami.com/rosetta.html`. Она содержит указатели на команды и утилиты, применяемые для решения основных административных задач в различных операционных системах.

Если вы работаете в Linux-системе, не стесняйтесь обращаться к UNIX-ресурсам общего назначения. Большая часть информации в них напрямую применима к Linux.

**Таблица 1.5. Веб-ресурсы, посвященные системному администрированию**

| Веб-сайт | Описание |
|---|---|
| `blogs.sun.com` | Огромная коллекция технических статей, посвященных, в основном, Solaris |
| `cpan.org` | Заслуживающая доверия коллекция Perl-модулей |
| `freshmeat.net` | Большой каталог программ для Linux и UNIX |
| `kernel.org` | Официальный веб-сайт разработчиков ядра Linux |
| `linux.com` | Linux-форум, прекрасный сайт для новичков |
| `linux.org` | Неофициальное хранилище информации по Linux |
| `linux.slashdot.org` | Огромный архив новейшей справочной технической информации по Linux |
| `linuxhq.com` | Хранилище информации по ядру и "заплат" к нему |
| `lwn.net` | Сборник материалов по Linux и программам с открытым исходным кодом |
| `lxer com` | Агрегатор новинок по Linux |
| `rootvg.net` | Сайт, ориентированный на системы AIX с множеством ссылок и форумов |
| `securityfocus.com` | Хранилище информации по общим вопросам компьютерной безопасности |
| `serverfault.com` | Совместно редактируемая база данных вопросов от системных администраторов |

[7] Или, в худшем случае, ссылающиеся на Google через `lmgtfy.com`.

*Окончание табл. 1.5*

| Веб-сайт | Описание |
|---|---|
| ServerFiles.com | Каталог сетевых программных и аппаратных средств |
| slashdot.org | Хранилище новейшей справочной технической информации по различным темам |
| solariscentral.org | Открытый блог с новостями и статьями по Solaris |
| sun.com/bigadmin | Сайт, содержащий информацию по администрированию Sun-систем |
| sunhelp.org | Прекрасная коллекция Sun-материалов |
| ugu.com | Сайт "Вселенная гуру UNIX" (UNIX Guru Universe) с информацией для системных администраторов |

[a] Сейчас его поддерживает Linux Foundation — некоммерческий консорциум развития Linux.

## 1.12. Как инсталлировать программное обеспечение

Подготовка к работе программного обеспечения подробно рассматривается в главе 12. Но для самых нетерпеливых мы прямо здесь расскажем о том, как выяснить, что уже установлено в вашей системе, и как получить новое программное обеспечение и инсталлировать его.

В современных операционных системах программное обеспечение разделено на пакеты, которые можно инсталлировать независимо друг от друга. При стандартной установке системы используется группа "стартовых" пакетов, которую можно при необходимости расширить.

Добавочные программные продукты зачастую предоставляются также в виде предварительно скомпилированных пакетов, но далеко не во всех системах. Большая часть программного обеспечения создается независимыми группами разработчиков, выпускающих программы в виде исходных кодов. Репозитории пакетов затем берут исходные коды, компилируют их в соответствии с особенностями конкретной системы и включают в пакеты полученные бинарные файлы. Как правило, намного проще инсталлировать бинарную версию пакета, предназначенную для данной системы, чем искать и компилировать исходный код. Однако нужно учитывать, что иногда создатели пакетов на один-два выпуска отстают от текущей версии системы.

Если в двух системах используется один и тот же формат обработки пакетов, то это еще не означает, что пакеты обеих систем будут взаимозаменяемыми. Например, в системах Red Hat и SUSE применяется формат RPM, однако структура их файловых систем неодинакова. Рекомендуется всегда пользоваться пакетами, созданными для конкретной системы.

В известных Linux-дистрибутивах предусмотрены отлаженные системы управления пакетами, которые включают средства доступа к репозиториям программных продуктов и поиска их в Интернете. Дистрибуторы активно поддерживают эти репозитории от имени сообщества, которое они представляют, поэтому у администраторов Linux-систем редко возникает необходимость выходить за рамки стандартного менеджера пакетов. Другими словами, жизнь прекрасна!

При этом UNIX-системы демонстрируют больше неопределенности по части управления пакетами. Solaris, HP-UX и AIX предоставляют соответствующие программные средства работы с пакетами, которые функционируют на уровне отдельных машин. Однако производители этих систем не поддерживают репозитории продуктов с откры-

тым исходным кодом, и поэтому пользователям, в основном, приходится заниматься самообслуживанием.[8] К сожалению, одним из главных звеньев цепочки, которая связывает систему пакетирования в одно целое, является метод ссылки одних пакетов на другие, позволяющий выразить информацию о взаимной зависимости и совместимости. Без центральной координации вся эта экосистема может быстро развалиться на отдельные части.

Действительность демонстрирует совершенно разные результаты. Дистрибутив Solaris предлагает добавочную систему (**pkgutil** от `blastwave.org`), которая предоставляет простую инсталляцию программного обеспечения из интернет-репозитория, во многом подобную системам, содержащимся на Linux-дистрибутивах. Система HP-UX содержит прекрасный интернет-репозиторий в форме HP-UX Porting и Archiving Centre (`hpux.connect.org.uk`), но пакеты должны быть загружены вручную и по отдельности. Что касается AIX, то пока трудно говорить о возможности использования предварительно скомпонованного программного обеспечения для этой системы.

Администраторы без доступа к предварительно скомпонованным бинарным версиям пакетов должны инсталлировать системы по-старому, т.е. загрузить архив исходного кода tar, а затем вручную сконфигурировать, скомпилировать и инсталлировать систему. Иногда этот процесс проходит гладко, а иногда может превратиться в кошмар: все зависит от используемого программного обеспечения и конкретной операционной системы.

В этой книге мы предполагаем, что дополнительное программное обеспечение уже установлено, и не собираемся мучить вас шаблонными инструкциями по инсталляции каждого пакета. При необходимости мы будем указывать точные названия пакетов для выполнения конкретного проекта. Однако большей частью мы не будем повторять инструкции по инсталляции, поскольку они практически идентичны для всех пакетов.

## Определение факта инсталляции программного обеспечения

По ряду причин иногда нелегко определить, какой программный пакет содержит нужный компонент. Для того чтобы выяснить, "прописан" ли важный для вас бинарный файл в строке поиска, лучше использовать команду **which** (а не действовать на пакетном уровне). Например, следующая команда сообщает о том, что компилятор GNU C инсталлирован на данном компьютере.

```
aix$ which gcc
/opt/pware/bin/gcc
```

Если команда **which** не помогла, воспользуйтесь командой **whereis**, которая ведет поиск в системных каталогах без учета строки поиска. Другой вариант — это чрезвычайно удобная команда **locate**, которая просматривает предварительно скомпилированный индекс файловой системы в поиске имен файлов, соответствующих заданному шаблону. Команда **locate** входит в состав GNU-пакета **findutils**, который по умолчанию включен в большинство систем Linux, но для UNIX его необходимо инсталлировать вручную.

С помощью команды **locate** можно искать не только программы или пакеты, но и файлы любых типов. Например, если не известно точно, где искать файл заголовков **signal.h** (в нем содержатся определения сигналов Linux), попробуйте поступить так.

[8] Разработчики OpenSolaris не предлагают систему управления пакетами и интернет-репозиторий. Хотя этих средств нет в Solaris 10, ими предполагается оснастить Solaris 11.

```
ubuntu$ locate signal.h
/usr/include/signal.h
/usr/include/asm/signal.h
/usr/include/asm-generic/signal.h
/usr/include/linux/signal.h
...
```

База данных команды **locate** периодически обновляется командой **updatedb**, которая запускается демоном **cron**. Следовательно, результаты работы команды locate не всегда отражают последние изменения в системе.

🕮 Подробнее об управлении пакетами рассказывается в главе 12.

Если известно имя искомого пакета, воспользуйтесь системными утилитами обработки пакетов, которые непосредственно проверят наличие пакета. Например, в системах Red Hat или SUSE следующая команда определяет наличие интерпретатора языка Python (и его инсталлированной версии):

```
redhat$ rpm -q python
python-2.4.3-21.el5
```

## Добавление новых программ

Если возникнет необходимость инсталлировать дополнительное программное обеспечение, вам, прежде всего, нужно определить каноническое имя соответствующего программного пакета. Например, вам следует преобразовать ваше желание "хочу инсталлировать **locate**" в версию "нужно инсталлировать пакет **findutils**" или перевести намерение "инсталлировать **named**" в вариант "инсталлировать BIND". В этом вам поможет множество веб-индексов, эффективно работающих при указании конкретной системы, особенно при использовании Google. Например, при введении в поле поиска "команда locate" вас направят точно по адресу. Если вы работаете в системе UNIX, укажите также имя операционной системы.

Если вам известно имя нужной утилиты, можете загрузить и инсталлировать ее. В Linux и Solaris полная установка обычно состоит в использовании одной команды, поскольку в этих системах уже инсталлирована утилита **pkgutil**. Для HP-UX и AIX вам придется загрузить либо предварительно собранный бинарный пакет, либо исходный код проекта. В последнем случае попробуйте с помощью Google найти официальную веб-страницу проекта и загрузить оттуда его исходный код.

Следующие примеры демонстрируют инсталляцию утилиты **wget** для каждой системы, взятой в этой книге в качестве примера. Эта GNU-утилита, которая превращает сетевую загрузку файлов (поддерживающую протоколы HTTP и FTP) в элементарную команду, очень полезна для выполнения сценариев. Утилита **wget** инсталлируется по умолчанию в каждом из рассматриваемых здесь примеров систем Linux; при этом отметим, что представленные ниже команды можно использовать как во время начальной инсталляции, так и при последующих обновлениях системы.

В системе Ubuntu используется программа для установки, обновления и удаления программных пакетов в операционных системах Debian (Advanced Package Tool — APT).

```
ubuntu# apt-get install wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
wget is already the newest version.
0 upgraded, 0 newly installed, 0  to remove and 0 not upgraded.
```

Вот как выглядит версия SUSE.

```
suse# yast --install wget
<runs in a terminal-based UI>
```

Вариант инсталляции в системе Red Hat.

```
redhat# yum install wget
Loaded plugins: fastestmirror
…
Parsing package install arguments
Package wget-1.10.2-7.el5.i386 is already installed and latest version
Nothing to do
```

В системе Solaris инсталляция выполняется с помощью уже установленной утилиты `pkutil` (для получения инструкций по данной установке см. сайт `blastwave.org`).

```
solaris#  /opt/csw/bin/pkgutil --install wget
<multiple pages of output as seven packages are installed>
```

Для системы HP-UX мы нашли на сайте `hpux.connect.org.uk` соответствующий бинарный пакет и загрузили его в каталог **/tmp**. Команды распаковки и инсталляции в этом случае выглядят так.

```
hpux# gunzip /tmp/wget-1.11.4-hppa-11.31.depot.gz
hpux# swinstall -s /tmp/wget-1.11.4-hppa-11.31.depot wget
=======  05/27/09 13:01:31 EDT  BEGIN swinstall SESSION
   (non-interactive) (jobid=hpux11-0030)
   * Session started for user "root@hpux11".
   * Beginning Selection
   * Target connection succeeded for "hpux11:/".
   * Source:    /tmp/wget-1.11.4-hppa-11.31.depot
   * Targets:   hpux11:/
   * Software selections:
      wget.wget-RUN,r=1.11.4,a=HP-UX_B./800
   * Selection succeeded.
   * Beginning Analysis and Execution
   …
   * Analysis and Execution succeeded.
   …
```

Пакет **depot** (в строке с командой **swinstall**) должен быть задан как полный путь, начинающийся косой чертой "/"; в противном случае команда **swinstall** попытается найти нужный файл в сети. Утилита **wget** при завершении инсталляции сообщает команде **swinstall**, какой пакет надо загрузить из файла **depot**.

К сожалению, этот процесс инсталляции в действительности не так прост, как может показаться на первый взгляд. Инсталлируемая версия утилиты `wget` на самом деле не будет работать, поскольку некоторые библиотеки, от которых она зависит, не инсталлированы.

```
hpux$ wget http://samba.org/samba/docs/Samba3-HOWTO.pdf
/usr/lib/dld.sl: Can't open shared  library: /usr/local/lib/libcrypto.sl
/usr/lib/dld.sl: No such file or directory
```

```
[HP ARIES32]: Core file for 32 bit PA-RISC application
[HP ARIES32]: /usr/local/bin/wget saved to /tmp/core.wget.
```

Утилита **swinstall** обладает встроенным механизмом управления зависимостями, но его возможности, к сожалению, не распространяются на интернет-репозитории. Прочитайте "паспортные данные" утилиты **swinstall** и инсталлируйте все необходимые для нее пакеты (в данном случае — еще шесть).

## Сборка программного обеспечения из исходного кода

Фактически существует, по крайней мере, один бинарный пакет **wget**, доступный для AIX в формате RPM. Поиск в Google при заданной строке "aix wget rpm" вполне оправдает ваши надежды. После загрузки вам останется выполнить совсем простую команду инсталляции.

```
aix# rpm --install wget-1.11.4-1.aix5.1.ppc.rpm
```

В целях иллюстрации построим AIX-версию пакета **wget** на основе оригинального исходного кода.

Прежде всего, нам нужно найти этот код, но это не проблема: первый же результат, предложенный системой Google для заданной строки поиска "wget", направит нас прямо на страницу GNU-проекта, которая содержит нужные ссылки. После загрузки текущей версии в каталог **/tmp** мы распаковываем ее, задаем конфигурацию, компонуем и инсталлируем.

```
aix# cd /tmp; gunzip wget-1.11.4.tar.gz
aix# tar xfp wget-1.11.4.tar
aix# cd wget-1.11.4
aix# ./configure --disable-ssl --disable-nls # См. комментарии ниже
configure: configuring for GNU Wget 1.11.4
checking build system type... rs6000-ibm-aix
...
config.status: creating src/config.h
config.status: executing default commands
generating po/POTFILES from ./po/POTFILES.in
creating po/Makefile
aix# make
<несколько страниц компиляционных сообщений>
aix# make install
<о странице выходных сообщений>
```

Приведенную последовательность команд инсталляции **configure/make/make install** обычно применяют для большинства UNIX- и Linux-утилит во всех системах, но, безусловно, при наличии инсталлированной среды разработки и программ, составляющих заданный пакет. При этом будет не лишним прочитать файлы **INSTALL** или **README**, сопровождающие данный пакет.

В данном случае использование ключей **--disable-ssl** и **--disable-nls** в команде **configure** позволяет опустить некоторые wget-функции, входящие в состав библиотек, которые не были инсталлированы (при обслуживании реальной системы вам, возможно, захочется включить в инсталляцию опущенные здесь части пакета). Чтобы узнать все опции процесса конфигурации, используйте команду **configure --help**. Обратите внимание на полезную опцию **--prefix**=*каталог* команды **configure**, которая позволит вам поместить программное обеспечение не в каталог **/usr/local**, а в более подходящий для вас *каталог*.

## 1.13. Издержки профессии

Системные администраторы — настоящие "многостаночники". Они часто имеют другую работу, просто их попросили присмотреть за несколькими компьютерами "на стороне". Если вы один из таких людей, подумайте, к чему в конце концов это может привести.

Чем больше вы будете знать о своей системе, тем больше пользователи будут зависеть от вас. Сети неуклонно разрастаются, и вы будете вынуждены тратить все больше и больше времени на выполнение функций администратора. Вскоре окажется, что вы — единственный человек во всей организации, который знает, как решить ряд важнейших проблем.

Если коллеги стали считать вас локальным системным администратором, от этой роли уже трудно отказаться. Мы знаем нескольких людей, которым пришлось поменять место работы, чтобы избавиться от дополнительной нагрузки. Поскольку круг обязанностей системного администратора четко очертить нельзя, от вас, скорее всего, потребуют, чтобы вы были не только штатным администратором, но и штатным инженером, писателем, а также специалистом по психоанализу.

Практика показывает, что некоторые администраторы, ставшие таковыми волей обстоятельств (а не по "велению души"), стараются избежать потока просьб о помощи, демонстрируя раздражение и плохо выполняя свои обязанности.[9]

Не стоит придерживаться такого подхода, так как вы будете плохо выглядеть в глазах окружающих и у вас могут возникнуть дополнительные проблемы. Вместо этого мы предлагаем просто фиксировать время, затрачиваемое на системное администрирование. Ведите работу на должном уровне и собирайте доказательства, которые можно будет использовать, когда вы попросите начальника освободить вас от обязанностей администратора. В большинстве организаций для того, чтобы добиться замены, приходится упрашивать руководство полгода, а то и год, так что учитывайте это в своих планах.

С другой стороны, может оказаться, что системное администрирование вам нравится и вы хотите стать штатным администратором. В этом случае проблем с поиском работы у вас не будет. К сожалению, сама работа от этого не станет легче. О политических аспектах системного администрирования рассказывается в главе 32.

## 1.14. Рекомендуемая литература

- Robbins Arnold. *UNIX in a Nutshell (4th Edition)*. Sebastopol, CA: O'Reilly Media, 2008.
- Siever Ellen, Aaron Weber and Stephen Figgins. *Linux in a Nutshell (5th Edition)*. Sebastopol, CA: O'Reilly Media, 2006.
- Gancarz Mike. *Linux and the Unix Philosophy*. Boston: Digital Press, 2003.
- Salus Peter H. *The Daemon, the GNU & the Penguin: How Free and Open Software is Changing the World*. Reed Media Services, 2008.

  В настоящее время эта захватывающая история развития программного обеспечения с открытым исходным кодом, исследованная наиболее известным историком UNIX, переводится в цифровую форму на сайте `groklaw.com` в соответствии с лицензией Creative Commons (Сообщество представителей творческих профессий).

[9] Обозначенная тенденция с любовью и черным юмором описана в рассказах Синона Травальи (Simon Travaglia *Bastard Operator from Hell* на сайте `bofh.ntk.net` в разделе `BOFH`.).

URL-адрес самой книги достаточно длинный; попытайтесь найти действующую ссылку на сайте `groklaw.com` или проверьте следующий сокращенный вариант: `tinyurl.com/d6u7j`.

- Raymond Eric S. *The Cathedral & The Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly Media, 2001.

## Системное администрирование

- Limoncelli Thomas A., Christina J. Hogan and Strata R. Chalup. *The Practice of System and Network Administration (Second Edition)*. Reading, MA: Addison-Wesley, 2008.

  Это прекрасная книга, описывающая политические и процедурные аспекты системного администрирования. Авторы ведут блог по системному администрированию (`everythingsysadmin.com`).

- Frisch Æleen. *Essential System Administration (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2002.

  Это классическое универсальное руководство по администрированию систем UNIX, которое, к сожалению, уже несколько устарело. Надеемся, что его новая версия уже в работе!

## Основной инструментарий

- Robbins Arnold, Elbert Hannah and Linda Lamb. *Learning the vi and Vim Editors*. Sebastopol, CA: O'Reilly Media, 2008.
- Powers Shelly, Jerry Peek, Tim O'reilly and Mike Loukides. *UNIX Power Tools (3rd Edition)* . Sebastopol, CA: O'Reilly Media, 2003.
- Michael Randal K. *Mastering UNIX Shell Scripting: BASH, Bourne, and Korn Shell Scripting for Programmers, System Administrators, and UNIX Gurus*. Indianapolis, IN: Wiley Publishing, Inc., 2008.
- Robbins Arnold and Nelson H. F. Beebe. *Classic Shell Scripting. Sebastopol*, CA: O'Reilly Media, 2005.
- Wall Larry, Tom Christiansen and Jon Orwant. *Programming Perl (3rd Edition)*. Cambridge, MA: O'Reilly Media, 2000.
- Christiansen Tom and Nathan Torkington. *Perl Cookbook (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.
- Blank-Edelman David N. *Automating System Administration with Perl (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2009.
- Pilgrim Mark. *Dive Into Python*. Berkeley, CA: Apress, 2004.

  К этой книге есть свободный веб-доступ по адресу: `diveintopython.org`.

- Flanagan David and Yukihiro Matsumoto. *The Ruby Programming Language*. Sebastopol, CA: O'Reilly Media, 2008.

  Эта книга, имеющая оптимистический подзаголовок *Everything You Need to Know* (Все, что вам необходимо знать), к сожалению, написана несколько суховато. Но она подкупает полнотой описания языка Ruby (версии 1.9) с богатством деталей, которые могут быть известны лишь создателю языка. Это самая лучшая книга для тех, кто уже владеет (на практическом уровне) языками Perl или Python.

# 1.15. Упражнения

1.1. Какую команду нужно ввести, чтобы получить информацию о драйвере терминала **tty** (а не о команде **tty**)? Как прочитать соответствующую страницу документации, хранящуюся в каталоге **/usr/local/share/man**?

1.2. Определяет ли общий конфигурационный файл поведение команды **man** в вашей системе? Какие строки нужно в него добавить, чтобы локальные материалы хранились в каталоге **/doc/man**? Как организовать каталог **/doc/man**, чтобы его структура соответствовала иерархии справочных страниц?

1.3. Как в настоящее время протекает разработка ядра Linux? Каковы наиболее актуальные вопросы? Какие организации играют в этом процессе ключевую роль? Кто руководит проектом?

1.4. ★Изучите возможности нескольких систем UNIX и Linux и порекомендуйте операционную систему для каждого из перечисленных ниже случаев. Аргументируйте свой выбор.

а) домашний компьютер пользователя

б) университетская компьютерная лаборатория

в) корпоративный веб-сервер

д) серверный кластер, обеспечивающий работу базы данных для судоходной компании

1.5. ★Предположим, вы обнаружили, что определенное свойство демона **httpd** сервера Apache работает не так, как описано в документации к Ubuntu.

а) что нужно сделать перед тем, как сообщать об ошибке?

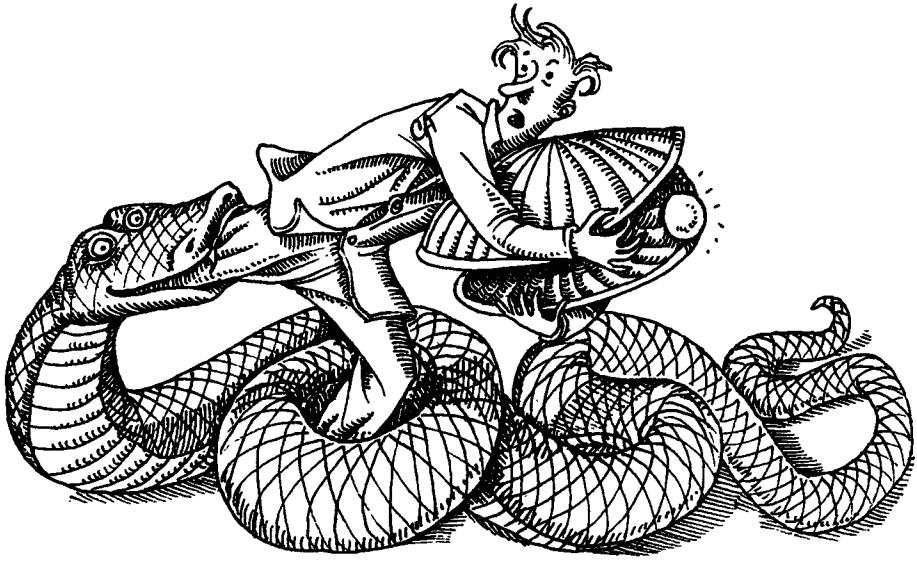
б) если вы решили, что это действительно ошибка, то кого нужно уведомить и каким способом?

в) какую информацию необходимо включить в отчет об ошибке?

1.6. ★★Системы Linux серьезно заявили о себе в производственных средах. Обречены ли UNIX-системы? Почему?

# Глава 2

# Сценарии и командная оболочка



Настоящие системные администраторы пишут сценарии. Сценарии позволяют стандартизировать и автоматизировать выполнение рутинных операций по администрированию систем, тем самым освобождая столь ценное время администраторов для решения более важных и более интересных задач. В известном смысле сценарии — это документирование действий в виде официального перечня команд, необходимых для выполнения конкретной задачи.

Сценарии бывают самыми разными: от очень простых, которые инкапсулируют всего несколько статических команд, до значительных программных проектов, управляющих конфигурациями хостов и административной информацией всего узла. В этой книге нас, главным образом, интересуют небольшие повседневные сценарные проекты, с которыми обычно имеют дело системные администраторы, поэтому мы не будем подробно останавливаться на функциях поддержки (например, отслеживание ошибок и оценка проекта), необходимых для более крупных проектов.

При написании административных сценариев следует делать акцент на эффективности работы программиста и ясности кода, а не на производительности вычислительных средств. Не может служить оправданием для небрежно написанного сценария время его выполнения, т.е. не так уж важно, выполняется он за полсекунды или за две. Удивительно, но оптимизация слабо оправдывает затраты, даже в случае сценариев, которые регулярно выполняются демоном `cron`.

В течение долгого времени стандартный язык написания административных сценариев определялся командной оболочкой. В большинстве систем используется оболочка `bash` (“*B*ourne-*a*gain” *sh*ell — “возрожденная” оболочка Борна), но в ряде систем UNIX

все еще работают **sh** (оригинальная оболочка Борна) и **ksh** (оболочка Корна). Сценарии оболочки обычно применяются для выполнения таких несложных задач, как автоматизация последовательности команд или сборка нескольких фильтров для обработки данных.

Командная оболочка всегда доступна, поэтому ее сценарии отличаются относительной переносимостью и независимостью, в отличие от вызываемых ими команд. Пока вы будете выбирать оболочку, она может выбрать вас: большинство сред включает объемистое дополнение уже существующих **sh**-сценариев, с которыми администраторам часто приходится знакомиться, разбираться и заниматься их настройкой.

При написании более сложных сценариев желательно использовать такие языки программирования, как Perl или Python, поскольку они оба хорошо подходят для выполнения административной работы. Эти языки (по сравнению с командной оболочкой) вобрали в себя такие мощные достижения в области создания языков программирования последних двух десятилетий и оснащены такими возможностями по части обработки текстов (столь бесценные для системных администраторов), что оболочке **sh** остается лишь "рыдать от стыда где-то в уголке".

Основной недостаток языков Perl и Python состоит в отсутствии четкости в установке их сред, особенно в случае, если вы начинаете использовать библиотеки с откомпилированными компонентами от сторонних производителей. Командная оболочка не страдает такой проблемой, поскольку не имеет модульной структуры и не содержит сторонних библиотек.

В этой главе дается краткий обзор языковых инструментов написания сценариев **bash**, Perl и Python, а также регулярных выражений.

# 2.1. Основы работы командной оболочки

Прежде чем обсуждать написание сценариев оболочки, рассмотрим некоторые базовые команды и синтаксис языка командной оболочки. Материал этого раздела актуален для всех основных оболочек **sh**-семейства (включающего оболочки **bash** и **ksh**, но не **csh** или **tcsh**), вне зависимости от используемой вами платформы. Опробуйте формы, с которыми вы пока еще не знакомы, и не бойтесь экспериментировать!

## Редактирование команд

Мы наблюдали, как многие редактируют командные строки с помощью клавиш управления курсором. Вы ведь так не поступаете в своем текстовом редакторе, верно?

Все основные команды редактора **emacs** доступны при редактировании уже выполненного ряда команд (т.е. "предыстории"). Нажав комбинацию клавиш <Ctrl+E>, вы переместитесь в конец строки, а с помощью комбинации клавиш <Ctrl+A> — в начало. Нажатие комбинации клавиш <Ctrl+P> позволяет вернуться к предыдущим командам и вызвать их для редактирования. С помощью комбинации клавиш <Ctrl+R> вы сможете шаг за шагом "прокрутить" свою "предысторию" и найти старые команды.

Если вы предпочитаете использовать редактор **vi**, переключите свою командную оболочку в режим **vi**.

```
$ set -o vi
```

В режиме **vi** редактирование является модальным, но вы начинаете работать в режиме ввода команд. Для того чтобы выйти из режима ввода, нажмите вначале клавишу <Esc>, а затем клавишу <i>, чтобы вернуться в него. В режиме редактирования нажатие клавиши <w> переместит вас вперед на одно слово; использование комбинаций

клавиш <f+X> позволит найти следующий символ "X" в строке и т.д. "Прогуляться" по командам, зафиксированным в "предыстории", можно с помощью клавиш <Esc> и <k>. Решили вернуться в режим **emacs**-редактирования? Выполните эту команду.

```
$ set -o emacs
```

## Каналы и перенаправление потоков

Каждому процессу доступны по меньшей мере три информационных канала: "стандартный ввод" (STDIN), "стандартный вывод" (STDOUT) и "стандартная ошибка" (STDERR). Эти каналы устанавливаются ядром системы "от имени процесса", и поэтому сам процесс не обязательно знает их направленность. Они, например, могут быть связаны с окном терминала, файлом, подключением к сети или с каналом, принадлежащим к другому процессу.

Для систем UNIX используется унифицированная модель ввода-вывода, в которой каждому каналу присваивается целое число, именуемое *файловым дескриптором*. Точное число (номер), назначаемое каналу, обычно не имеет значения, но каналам STDIN, STDOUT и STDERR гарантированно соответствуют файловые дескрипторы 0, 1 и 2 соответственно, чтобы обеспечить безопасное обращение к ним по номерам. В контексте интерактивного окна терминала канал STDIN обычно считывает данные с клавиатуры, а оба канала STDOUT и STDERR записывают свои выходные данные на экран.

Большинство команд принимает входные данные из канала STDIN. Выходная информация записывается ими в канал STDOUT, а сообщения об ошибках — в канал STDERR. Это соглашение позволяет объединять команды подобно строительным блокам для организации конвейерной обработки данных.

Командная оболочка интерпретирует символы "<", ">" и ">>" как инструкции по изменению направления передаваемых командой данных в файл или принимаемых данных из файла. Символ "<" связывает канал STDIN с содержимым некоторого существующего файла. Символы ">" и ">>" перенаправляют поток STDOUT; причем символ ">" используется для замены содержимого файла, а ">>" — для добавления данных в его конец. Например, команда

```
$ echo "Это тестовое сообщение." > /tmp/mymessage
```

сохраняет одну строку в файле `/tmp/mymessage` (при необходимости файл будет создан). Следующая команда отправляет содержимое этого файла по электронной почте пользователю `johndoe`.

```
$ mail -s "Mail test" johndoe < /tmp/mymessage
```

Для того чтобы перенаправить потоки STDOUT и STDERR в одно и то же место, используйте символ ">&". Для того чтобы перенаправить только поток STDERR, используйте вариант "2>".

На примере команды **find** можно показать, почему важно обрабатывать потоки STDOUT и STDERR отдельно. Дело в том, что она формирует выходные данные по обоим каналам, особенно в случае ее выполнения непривилегированным пользователем. Например, при выполнении команды

```
$ find / -name core
```

обычно генерируется так много сообщений об ошибках "permission denied" (отсутствие прав доступа), что настоящие результаты поиска теряются в "шуме". Чтобы отбросить все сообщения об ошибках, можно использовать такой вариант.

```
$ find / -name core 2> /dev/null
```

В этой версии команды **find** только настоящие совпадения (где пользователь имеет разрешение на чтение родительского каталога) выводятся в окно терминала. Чтобы сохранить список совпадающих путей в файле, выполните такую команду.

```
$ find / -name core > /tmp/corefiles 2> /dev/null
```

Эта командная строка отправляет совпадающие пути в файл **/tmp/corefiles**, отбрасывая все ошибки и ничего не посылая в окно терминала.

Для того чтобы связать канал STDOUT одной команды с каналом STDIN другой, используйте символ "|". Вот два примера.

```
$ ps -ef | grep httpd
$ cut -d: -f7 < /etc/passwd | sort -u
```

В первом примере выполняется команда **ps**, генерирующая список процессов, из которого при "проходе" через команду **grep** выбираются строки, содержащие слово "httpd". Результат выполнения команды **grep** никуда больше не перенаправляется, поэтому совпадающие строки попадают в окно терминала.

Во втором примере команда **cut** выбирает из файла **/etc/passwd** пути к оболочке каждого пользователя. Список оболочек затем отправляется через команду **sort -u**, чтобы сформировать отсортированный список уникальных значений.

Для того чтобы последующая команда выполнялась только в случае успешного выполнения предыдущей, можно разделить эти команды символом "**&&**". Например, командная строка

```
$ lpr /tmp/t2 && rm /tmp/t2
```

удалит файл **/tmp/t2** тогда и только тогда, когда она успешно дождется очереди на печать. В данном случае успех выполнения команды **lpr** будет зафиксирован при получении кода завершения, равного нулю. Использование для этой цели символа, означающего операцию "логическое И", может сбить с толку тех, кто в других языках программирования использовал вычисления в сокращенной форме записи. Если кому-то это непонятно, не стоит слишком долго здесь застревать; а просто примите это как идиому командной оболочки.

И наоборот, символ "||" обеспечит выполнение следующей команды только в том случае, если предыдущая команда не выполнится (т.е. сгенерирует ненулевое значение кода завершения).

Для того чтобы разбить команду на несколько строк, для наглядности отделяя тем самым код обработки ошибок от остальной части командного конвейера, можно использовать в сценариях обратную косую черту.

```
cp --preserve --recursive  /etc/* /spare/backup \
|| echo "Did NOT make backup"
```

Для получения обратного эффекта, т.е. объединения нескольких команд в одной строке, можно использовать в качестве разделителя точку с запятой.

## Использование переменных и кавычек

В операциях присваивания имена переменных никак не маркируются, но предваряются знаком доллара при ссылке на их значения.

```
$ etcdir='/etc'
$ echo $etcdir
/etc
```

Не ставьте до и после знака равенства (=) пробелы, в противном случае оболочка ошибочно примет имя вашей переменной за имя команды.

При ссылке на переменную можно заключить ее имя в фигурные скобки, чтобы синтаксический анализатор (да и сам человек) не сомневался в том, где заканчивается имя переменной и начинается другой текст; например, используйте запись **${etcdir}** вместо **$etcdir**. Обычно без фигурных скобок можно обойтись, но они могут оказаться полезными в случае, если вам понадобится раскрывать переменные в строках с двойными кавычками. Часто нужно сделать так, чтобы после значения переменной стояли буквы или знаки препинания.

```
$ echo "Saved ${rev}th version of mdadm.conf."
Saved 8th version of mdadm.conf.
```

Для имен переменных командной оболочки не существует стандартного соглашения, но обычно предполагается, что имена, полностью написанные прописными буквами, принадлежат переменным среды или переменным, считанным из файлов глобальной конфигурации. И чаще всего все буквы в именах локальных переменных — строчные, а отдельные их компоненты разделяются символами подчеркивания. Вообще имена переменных чувствительны к регистру букв.

Переменные среды автоматически импортируются в пространство имен оболочки **bash**, поэтому их можно устанавливать и считывать с использованием стандартного синтаксиса. Для того чтобы перевести переменную оболочки в ранг переменной среды, используйте команду **export** *имя_переменной*. Команды для переменных среды, которые вы хотите установить во время регистрации (при входе в систему), должны быть включены в ваш файл `~/.profile` или `~/.bash_profile`. Другие переменные среды, такие как `PWD` (для хранения имени текущего рабочего каталога), автоматически создаются командной оболочкой.

Командная оболочка интерпретирует строки, заключенные в одинарные и двойные кавычки, почти одинаково. Различие состоит в том, что строки в двойных кавычках служат субъектами для универсализации файловых имен (замены реальных символов в имени и расширении файла универсальными, т.е. такими метасимволами, как "*" и "?") и для раскрытия переменных (замены переменных их значениями).

```
$ mylang="Pennsylvania Dutch"
$ echo "I speak ${mylang}."
I speak Pennsylvania Dutch.
$ echo 'I speak ${mylang}.'
I speak ${mylang}.
```

Обратные апострофы (также известные как обратные галочки, левые кавычки, левые одиночные кавычки или открывающие кавычки) интерпретируются аналогично двойным кавычкам, но несут при этом дополнительную нагрузку. Эта нагрузка состоит в интерпретации содержимого такой строки, как команды оболочки, выполнении этой команды и замене строки результатом выполнения этой команды.

```
$ echo "There are `wc -l /etc/passwd` lines in the passwd file."
There are 28 lines in the passwd file.
```

## Команды общих фильтров

Любые корректные команды, которые считывают входные данные из канала STDIN и записывают результаты в канал STDOUT, могут быть использованы в качестве фильтра (т.е. компонента конвейера) для обработки данных. В этом разделе мы кратко рас-

смотрим некоторые из часто используемых команд фильтрации (хотя их полный список практически бесконечен). Команды фильтрации отличаются настолько сильным "коллективизмом", что порой даже трудно продемонстрировать их идивидуальное использование.

Большая часть команд фильтрации принимает в командной строке одно или несколько имен файлов. И только в том случае, если вы не можете задать соответствующие файлы, указывайте в качестве источника считываемой информации стандартный входной поток.

### Команда cut: разбивка строк на поля

Команда **cut** выводит выбранные части входных строк. Чаще всего эта команда используется для извлечения полей (ограниченных некоторыми разделителями), как в приведенном выше примере, который для удобства читателей повторим еще раз.

```
$ cut -d: -f7 < /etc/passwd | sort -u
```

Команда `cut` также может возвращать сегменты, определенные границами столбцов текста. В качестве стандартного разделителя служит символ <Tab>, но с помощью ключа **-d** разделители можно изменить. Ключ **-f** позволяет указать, какие поля нужно включить в результат.

Еще один пример использования команды **cut** показан в разделе, посвященном команде **uniq** (см. ниже).

### Команда sort: сортировка строк

Команда **sort** сортирует входные строки. Звучит просто, да? Хотя на самом деле не все так безоблачно: существуют потенциальные нюансы, связанные с точно заданными частями сортируемых строк (т.е. с использованием сортировочного ключа) и порядком сортировки. Наиболее распространенные варианты применения этой команды показаны в табл. 2.1, в отношении же остальных случаев обратитесь к соответствующей man-странице.

**Таблица 2.1. Ключи команды `sort`**

| Ключ команды | Значение |
|---|---|
| `-b` | Игнорировать ведущие пробельные символы |
| `-f` | Сортировать независимо от регистра букв |
| `-k` | Указать столбцы, которые формируют сортировочный ключ |
| `-n` | Сравнивать поля как целые числа |
| `-r` | Изменить порядок сортировки на противоположный |
| `-t` | Установить разделитель полей (по умолчанию — пробельный символ) |
| `-u` | Выводить только уникальные записи |

На примерах приведенных ниже команд демонстрируется различие между числовой и лексикографической сортировкой (последняя действует по умолчанию). В обеих командах используются ключи `-t:` и `-k3,3` для сортировки файла `/etc/group` по его третьему полю (в качестве разделителя используется двоеточие), содержащему идентификатор (ID) группы. Первая команда реализует числовую сортировку, а вторая — лексикографическую (в алфавитном порядке).

```
$ sort -t: -k3,3 -n /etc/group[1]
root:x:0:
bin:x:1:daemon
daemon:x:2:
…
$ sort -t: -k3,3 /etc/group
root:x:0:
bin:x:1:daemon
users:x:100:
…
```

### Команда uniq: вывод уникальных строк

Команда **uniq** по существу подобна команде **sort -u**, но у нее есть некоторые полезные ключи, которые команда **sort** не эмулирует. Так, ключ **-c** используется для подсчета количества экземпляров каждой строки, ключ **-d** — для отображения только строк, имеющих дубликаты, а ключ **-u** — для отображения только строк, не имеющих дубликаты. При этом входные данные должны быть предварительно отсортированы, обычно путем "пропускания" через команду **sort**.

Например, по результатам выполнения следующей команды видно, что у 20 пользователей поле login shell (стартовая оболочка) содержит значение **/bin/bash** и у 12 — **/bin/false**. (Последний результат характерен либо для псевдопользователей, либо для пользователей, учетные записи которых были заблокированы.)

```
$ cut -d: -f7 /etc/passwd | sort | uniq -c
   20 /bin/bash
   12 /bin/false
```

### Команда wc: подсчет строк, слов и символов

Подсчет количества строк, слов и символов в файле — еще одна распространенная операция, удобным средством реализации которой и является команда wc (*w*ord *c*ount). Выполненная без каких-либо ключей, она выведет все три результата подсчета.

```
$ wc /etc/passwd
32  77 2003 /etc/passwd
```

В контексте написания сценариев команду **wc** часто применяют только с одним из ключей (**-l**, **-w** или **-c**), чтобы результат ее выполнения состоял из одного числа. Эту форму чаще всего можно встретить внутри обратных апострофов, и тогда результат может быть сохранен или использован по назначению.

### Команда tee: копирование входного потока в два места

Командный конвейер, как правило, действует прямолинейно, но зачастую полезно распараллелить поток данных, т.е. "перехватить" его и отправить копию в файл или окно терминала. Это можно сделать с помощью команды tee, которая отправляет свой стандартный входной поток как в стандартный выходной канал, так и в файл, указанный в командной строке. Представьте действие этой команды в виде тройника в трубопроводе (перев. с англ. *tee* — тройник).

[1] Команда ***sort*** принимает спецификацию ключа ***-k3*** (вместо ***-k3,3***), но в этом случае вы, скорее всего, не получите ожидаемого результата. Без завершающего номера поля сортировочный ключ продолжится до конца строки.

Устройство **/dev/tty** можно считать синонимом для текущего терминала. Например, команда

```
$ find / -name core | tee /dev/tty | wc -l
```

выводит найденные путевые имена файлов `core` и результат подсчета их количества.

Часто работа конвейера с командой **tee**, выводящей результаты и в файл, и в окно терминала (для проверки), занимает много времени. Вы можете понаблюдать за первыми результатами, чтобы убедиться в том, что все работает как надо, а затем смело уходите: ведь результаты же сохранятся в файле.

### *Команды head и tail: чтение файла с начала или с конца*

Просмотр строк с начала или конца файла — обычная административная операция. Команды **head** и **tail** отображают по умолчанию десять строк, но вы можете использовать ключ, задающий желаемое количество строк.

Для интерактивного использования команда **head** уже несколько устарела, и вместо нее (в этом контексте) часто используется команда **less**, которая разбивает файлы для отображения по страницам. Но команду **head** можно успешно применять в сценариях и с другой целью. С командой **tail** используется ключ **-f**, который чрезвычайно полезен для системных администраторов. Вместо немедленного завершения (после вывода требуемого количества строк) команда **tail -f** ожидает "прибытия" новых строк, которые (как только они появятся "на горизонте") будут добавлены в конец файла, а затем выведены по назначению, что очень удобно для мониторинга журналов регистрации. Однако следует иметь в виду, что программа, которая реализует запись данных в файл, может буферизировать свои выходные данные. Даже если строки будут добавляться регулярно (с точки зрения логики), они могут стать видимыми только фрагментами объемом 1 или 4 Кбайт (KiB).[2]

Для остановки процесса мониторинга нажмите комбинацию клавиш <Ctrl+C>.

### *Команда grep: поиск текста*

При выполнении команды **grep** по мере просмотра входного текста выводятся строки, которые совпадают с заданным образцом (шаблоном). Свое название эта команда получила "в наследство" от команды **g/***regular_expression***/p**) из старого (и ныне действующего) редактора **ed**, используемого в самых первых версиях UNIX.

"Регулярные выражения" (regular expressions) — это шаблоны, предназначенные для поиска совпадающего с ними текста и написанные на стандартном языке шаблонов. Они представляют собой универсальный стандарт, используемый большинством программ при поиске по совпадению с шаблоном, хотя и с небольшими различиями в реализациях. Странное имя команды уходит своими корнями в оригинальные изыскания соответствующих разделов теории вычислений. Более детально синтаксис регулярных выражений рассматривается в разделе 2.3.

Подобно большинству фильтров, команда **grep** имеет множество ключей. Например, ключ **-c** позволяет выводить количество совпавших строк, ключ **-i** служит для игнорирования регистра букв при сопоставлении, а ключ **-v** предназначен для вывода отличающихся (а не совпавших) строк. Очень полезным является ключ **-l** (прописная буква "L"), который заставляет команду **grep** выводить только имена файлов, содержащие совпавшие с шаблоном строки, а не все совпавшие строки.

Например, выполнение команды

[2] О единицах измерения см. в разделе 1.8 главы 1.

```
$ sudo grep -l mdadm /var/log/*
/var/log/auth.log
/var/log/syslog.0
```

демонстрирует, что записи с именем утилиты **mdadm** нашлись в двух различных файлах системных журналов.

Команду **grep** можно считать классической реализацией базового механизма использования регулярных выражений, но в некоторых версиях предлагается выбор других диалектов. Например, Linux-команда **grep -p** служит для поиска выражений в стиле языка Perl, хотя в справочных страницах можно найти неопределенное предупреждение о том, что такой вариант носит "экспериментальный характер". Для полной уверенности в своих сценариях просто используйте язык Perl или Python.

## 2.2. Написание bash-сценариев

Командный арсенал оболочки **bash** позволяет успешно писать простые сценарии (без этого средства автоматизации системным администраторам пришлось бы вручную вводить команды в командную строку). Ваше мастерство в использовании командных строк вы должны воплотить в искусстве создания **bash**-сценариев (и наоборот), что поможет вам извлечь максимальную пользу из времени, потраченного на изучение оболочки **bash**. Но когда окажется, что ваш **bash**-сценарий превысил в объеме сотню строк или вам потребовались средства, которыми **bash** не обладает, это будет означать, что настало время переходить к языку Perl или Python.

В среде **bash** комментарии начинаются со знака # и продолжаются до конца строки. Как и в командной строке, вы можете разбить одну логическую строку на несколько физических строк, обозначив переход на новую строку символом обратной косой черты (\). И, наоборот, можно поместить на одной строке несколько операторов, разделив их точкой с запятой.

Сценарий для оболочки **bash** может состоять только из последовательности командных строк. Например, следующий сценарий **helloworld** просто выполняет команду **echo**.

```
#!/bin/bash
echo "Hello, world!"
```

Первая строка содержит "банальный" описательный оператор, который сообщает, что данный текстовый файл является сценарием, предназначенным для интерпретации командной оболочкой **/bin/bash**. При принятии решения о том, как выполнить этот файл, ядро отыщет соответствующий синтаксис. С точки зрения оболочки, "стремящейся" выполнить этот сценарий, первая строка представляет собой просто комментарий. Если бы оболочка находилась в другом каталоге, вам пришлось бы отредактировать эту строку.

Для того чтобы подготовить этот файл к выполнению, достаточно установить его бит, "отвечающий" за выполнение (см. раздел 6.5).

```
$ chmod +x helloworld
$ ./helloworld[3]
Hello, world!
```

[3] Если ваша оболочка понимает команду ***helloworld*** без префикса **./**, это означает, что в вашем пути поиска указан текущий каталог (**.**). И это плохо, поскольку дает другим пользователям возможность устраивать для вас "ловушки" в надежде, что вы будете пытаться выполнить определенные команды при переходе к каталогу, в котором они имеют доступ для записи.

Можно также непосредственно запустить (активизировать) оболочку в качестве интерпретатора сценария.

```
$ bash helloworld
Hello, world!
$ source helloworld
Hello, world!
```

Первая команда выполнит сценарий **helloworld** в новом экземпляре оболочки **bash**, а вторая — заставит вашу начальную оболочку прочитать содержимое файла, а затем выполнить его. Второй вариант полезен в случае, когда сценарий устанавливает переменные среды или выполняет другие операции настройки, применимые только к текущей оболочке. Обычно этот вариант используется при написании сценариев, включающих содержимое файла конфигурации, написанного в виде ряда присваиваний значений **bash**-переменным.[4]

🕮 Подробнее о битах разрешения (доступа) можно прочитать в разделе 6.5.

Если вы пришли сюда из мира Windows, то вам привычно использовать понятие расширения файла, по которому можно судить о типе файла, а также о том, можно ли его выполнить. В мире UNIX и Linux признак того, может ли файл быть выполнен (и если да, то кем), содержится в специальных битах полномочий. При желании вы можете наделить свои **bash**-сценарии суффиксом **.sh**, который бы напоминал вам об их типе, но тогда при выполнении соответствующей команды вам придется вводить суффикс **.sh**, поскольку UNIX не интерпретирует расширения специальным образом.

## От команд к сценариям

Прежде чем переходить к особенностям написания bash-сценариев, остановимся на методике этого процесса. Многие пишут bash-сценарии так же, как Perl- или Python-сценарии, т.е. используя какой-нибудь текстовый редактор. Но все же удобнее рассматривать в качестве интерактивной среды разработки сценариев режим с приглашением на ввод команды.

Предположим, например, что у вас есть журналы регистрации, именованные с суффиксами .log и .LOG и разбросанные по всей иерархической системе каталогов. Допустим также, что вы хотите изменить эти суффиксы, приведя их к "прописному" виду. Прежде всего, попытаемся отыскать все эти файлы.

```
$ find . -name '*log'
.do-not-touch/important.log
admin.com-log/
foo.log
genius/spew.log
leather_flog
...
```

Так, похоже на то, что нам надо включить в шаблон точку и игнорировать при поиске каталоги. Нажмите комбинацию клавиш <Ctrl+P>, чтобы вернуть команду в командную строку, а затем модифицируйте ее.

```
$ find . -type f -name '*.log '
.do-not-touch/important.log
foo.log
```

[4] Синонимом для команды ***source*** служит команда "с точкой" (dot-команда), например ***helloworld***.

```
genius/spew.log
...
```

Ну вот, это уже выглядит лучше. Правда, каталог **.do-not-touch** (т.е. "не трогать") вызывает смутное чувство опасности; но мы можем избавиться от этого неприятного холодка.

```
$ find . -type f -name '*.log ' | grep -v .do-not-touch
foo.log
genius/spew.log
...
```

Теперь все в порядке: мы получили абсолютно точный список файлов, которые должны быть переименованы. Попробуем сгенерировать несколько новых имен.

```
$ find . -type f -name '*.log ' | grep -v .do-not-touch | while read fname
> do
> echo mv $fname ${fname/.log/.LOG/}
> done
mv foo.log foo.LOG
mv genius/spew.log genius/spew.LOG
...
```

Да, это именно те команды, которые позволят переименовать нужные файлы. А как мы это делаем в реальности? Мы могли бы снова вызвать уже выполненную команду и отредактировать команду **echo**, чтобы заставить оболочку **bash** выполнять команды **mv**, а не просто выводить их. Ведь передача команд в отдельную копию оболочки **bash** — более надежный вариант работы, который к тому же требует меньшего объема редактирования.

Нажав комбинацию клавиш <Ctrl+P>, мы обнаруживаем, что оболочка **bash** заботливо свернула наш мини-сценарий в одну-единственную строку. К этой "уплотненной" командной строке мы просто добавляем канал, передающий наши выходные данные команде **bash -x**.

```
$ find . -type f -name '*.log ' | grep -v .do-not-touch | while read fname;
    do echo mv $fname ${fname/.log/.LOG/}; done | bash -x
+ mv foo.log foo.LOG
+ mv genius/spew.log genius/spew.LOG
...
```

Ключ **-x** команды **bash** обеспечивает вывод каждой команды перед ее выполнением.

Теперь мы завершили переименование файлов, но нам хотелось бы сохранить этот сценарий, чтобы можно было использовать его снова. Встроенная в **bash** команда **fc** по своему действию во многом аналогична нажатию комбинации клавиш <Ctrl+P>, но вместо возврата последней команды в командную строку она передает команду в заданный вами редактор. Добавьте в свой файл строку идентификационного комментария, поместите сохраненный файл в приемлемый для вас каталог (например, **~/bin** или **/usr/local/bin**), сделайте файл исполняемым, и вы получите настоящий сценарий.

Итак, подытожим.

- Разрабатывайте сценарий (или его часть) в виде конвейера команд, причем пошагово и в режиме выполнения командных строк.
- Пересылайте результат в стандартный выходной поток, проверяя правильность работы используемых вами команд.
- На каждом этапе используйте буфер ранее выполненных команд для их появления в командной строке и инструменты редактирования — для их модификации.

- Пока вы получаете неправильные результаты, можно считать, что вы, по сути, ничего не сделали, и до тех пор, пока команды не заработают так, как надо, ничего (из уже сделанного) не надо удалять.
- Если результат выглядит правильным, выполните команды на реальном примере, чтобы убедиться, что все получилось так, как ожидалось.
- Используйте команду **fc**, чтобы зафиксировать свою работу в редакторе, оформите ее соответствующим образом и сохраните.

В приведенном выше примере мы вывели командные строки, а затем направили их в подоболочку для выполнения. Этот метод не является универсально применимым, но часто оказывается полезным. В качестве альтернативного варианта можно фиксировать результаты, перенаправляя их в файл. Терпеливо добивайтесь получения нужных результатов, и пока вы не увидите их, не выполняйте никаких потенциально деструктивных действий.

## Организация ввода и вывода данных

Команда **echo** не отличается интеллектуальностью, но зато проста в применении. Для получения большего контроля над выводом данных используйте команду **printf**. Она не очень удобна, поскольку предполагает, что вы должны в явном виде указывать в нужных для вас местах символы перехода на новую строку (\n), но позволяет использовать символ табуляции и другие средства форматирования результата. Сравните результаты выполнения следующих двух команд.

```
$ echo "\taa\tbb\tcc\n"
\taa\tbb\tcc\n
$ printf "\taa\tbb\tcc\n"
    aa    bb    cc
```

В некоторых системах работа команд **echo** и **printf** поддерживается на уровне ОС (обычно соответствующие им исполняемые файлы хранятся в каталогах **/bin** и **/usr/bin** соответственно). Хотя эти команды и встроенные в оболочку утилиты в целом подобны, они могут иметь незначительные отличия, и особенно это касается команды **printf**. Вы можете либо придерживаться **bash**-синтаксиса, либо вызывайте "внешнюю" команду **printf**, указывая ее полный путь.

Для того чтобы сформировать для пользователя приглашение ввести данные, можно использовать команду **read**.

```
#!/bin/bash

echo -n "Введите свое имя: "
read user_name

if [ -n "$user_name" ]; then
     echo "Привет $user_name!"
     exit 0
else
     echo "Вы не назвали свое имя!"
     exit 1
fi
```

Ключ **-n** в команде **echo** подавляет привычный переход на новую строку, но здесь была бы кстати команда **printf**. Мы еще рассмотрим вкратце синтаксис оператора **if**, но его действие здесь, с нашей точки зрения, очевидно. Ключ **-n** в операторе **if** обе-

спечит значение истины, если его строковый аргумент не окажется нулевым. Вот как выглядит результат выполнения этого сценария.

```
$ sh readexample
Введите свое имя: Ron
Привет Ron!
```

## Функции и аргументы командной строки

Аргументы командной строки служат для сценария переменными с числовыми именами: $1 — первый аргумент командной строки, $2 — второй и т.д. Аргумент $0 содержит имя, по которому был вызван сценарий, например **./bin/example.sh**, т.е. это не фиксированное значение.

Переменная $# содержит количество переданных аргументов командной строки, а переменная $* — все эти аргументы. Ни одна из этих переменных не учитывает аргумент $0.

При вызове сценария с некорректными аргументами или вообще без них должно быть выведено короткое сообщение с напоминанием о том, как следует использовать данный сценарий. Приведенный ниже сценарий, принимающий два аргумента, проверяет, являются ли оба эти аргумента каталогами, и, убедившись в этом, отображает их. Если аргументы оказываются недопустимыми, сценарий выводит сообщение о его правильном использовании и завершается с ненулевым значением кода возврата. Если программа, вызвавшая этот сценарий, проверит код возврата, она будет "знать", что этот сценарий не удалось выполнить корректно.

```
#!/bin/bash

function show_usage {
   echo "Использование: $0 source_dir dest_dir"
   exit 1
}
# Основная программа начинается здесь

if [ $# -ne 2 ]; then
     show_usage
else # Существуют два аргумента
    if [ -d $1 ]; then
         source_dir=$1
    else
         echo 'Недопустимый каталог-источник'
         show_usage
    fi
    if [ -d $2 ]; then
         dest_dir=$2
    else
         echo 'Недопустимый каталог-приемник'
         show_usage
    fi
fi

printf "Каталог-источник: ${source_dir}\n"
printf "Каталог-приемник: ${dest_dir}\n"
```

Для вывода сообщения о правильном использовании данного сценария мы создали отдельную функцию show_usage. Если бы позже этот сценарий был модифицирован и стал бы принимать дополнительные аргументы, это сообщение нужно было бы изменить только в одном месте.[5]

```
$ mkdir aaa bbb
$ sh showusage aaa bbb
Каталог-источник: aaa
Каталог-приемник: bbb
$ sh showusage foo bar
Недопустимый каталог-источник
Использование: showusage source_dir dest_dir
```

Аргументы **bash**-функций обрабатываются практически так же, как аргументы командной строки: $1 — первый аргумент командной строки, $2 — второй и т.д. Как видно из приведенного выше примера, аргумент $0 содержит имя сценария.

Для того чтобы сделать предыдущий пример более устойчивым к ошибкам, мы могли бы заставить функцию show_usage принимать в качестве аргумента код ошибки. Это позволило бы конкретизировать код возврата для каждого типа отказа. Реализация этой идеи показана в следующем фрагменте программы.

```
function show_usage {
   echo "Использование: $0 source_dir dest_dir"
   if [ $# -eq 0 ]; then
        exit 99 # Выход с любым ненулевым кодом возврата
   else
        exit $1
   fi
}
```

В этой версии функции добавляется анализ наличия аргументов. Внутри функции переменная $# сообщает, сколько аргументов было ей передано. Сценарий завершится с кодом возврата 99, если при его вызове не было передано никаких аргументов. Но при передаче конкретного значения, например, такого, как

```
show_usage 5
```

сценарий завершится с этим кодом после вывода сообщения об использовании сценария. (Переменная $? содержит статус завершения последней выполненной команды, независимо от того, была ли она использована в сценарии или в командной строке.)

Между функциями и командами в оболочке **bash** существует строгая аналогия. Вы можете определить полезные функции в своем файле **~/.bash_profile**, а затем использовать их в командной строке как команды. Например, если ваш узел стандартизировал в сети порт 7988 для протокола SSH (в форме "безопасность через сокрытие"), вы можете определить в своем файле **~/.bash_profile** функцию **ssh**, чтобы быть уверенным в том, что она всегда будет запускаться (как команда) с ключом **-p 7988**.

```
function ssh {
   /usr/bin/ssh -p 7988 $*
}
```

[5] Обратите внимание на то, что сообщения об ошибках и сообщение об использовании сценария направляются в стандартный поток вывода данных. Разве не логичнее было бы направить их в стандартный канал вывода ошибок? В действительности это было бы корректнее, но поскольку не предполагается использовать этот сценарий в качестве фильтра, канал вывода данных здесь не так уж и важен.

Подобно многим командным оболочкам, **bash** обладает механизмом псевдонимов (**alias**), который может репродуцировать этот ограниченный пример в еще более лаконичном виде, тем не менее функции остаются более универсальным и более мощным средством. Поэтому забудьте о псевдонимах и просто используйте функции.

## Область видимости переменных

Переменные в рамках сценария имеют глобальный статус, но функции могут создавать собственные локальные переменные с помощью объявления `local`. Рассмотрим следующий код.

```
#!/bin/bash

function localizer {
    echo "==> В функции localizer начальное значение a равно '$a' "
    local a
    echo "==> После объявления local значение a стало равным '$a' "
    a="localizer version"
    echo "==> При выходе из функции localizer значение a равно '$a' "
}

a="test"
echo "До вызова функции localizer значение a равно '$a' "
localizer
echo "После вызова функции localizer значение a равно '$a'"
```

По приведенным ниже результатам выполнения сценария **scopetest.sh** видно, что локальная версия переменной `$a` внутри функции `localizer` "забивает" глобальную переменную `$a`. Глобальная переменная `$a` видна в функции `localizer` до тех пор, пока не встретится объявление `local`, т.е. по сути, объявление `local` работает как команда, которая в момент выполнения создает локальную переменную.

```
$ sh scopetest.sh
До вызова функции localizer значение a равно 'test'
==> В функции localizer начальное значение a равно 'test'
==> После объявления local значение a стало равным ' '
==> При выходе из функции localizer значение a равно 'localizer version'
После вызова функции localizer значение a равно 'test'
```

## Поток управления

Выше в этой главе мы уже рассмотрели несколько конструкций `if-then` и `if-then-else` (они работают вполне ожидаемым образом). Терминатором (признаком конца) для оператора `if` служит оператор `fi`. Для образования цепочки `if`-операторов можно использовать ключевое слово `elif`, означающее "`else if`".

```
if [ $base -eq 1 ] && [ $dm -eq 1 ]; then
    installDMBase
elif [ $base -ne 1 ] && [ $dm -eq 1 ]; then
    installBase
elif [ $base -eq 1 ] && [ $dm -ne 1 ]; then
    installDM
else
    echo '==> Installing nothing'
fi
```

Как и специальный `[]`-синтаксис для выполнения операций сравнения, так и "ключеподобные" имена операторов целочисленного сравнения (например **-eq**) уходят "наследственными корнями" в использование утилиты **/bin/test** из ранней командной оболочки Стивена Борна. В действительности квадратные скобки — это не что иное, как условное обозначение вызова утилиты **test**; они не являются частью оператора **if**.[6]

В табл. 2.2 собраны **bash**-операторы сравнения для чисел и строк. В отличие от Perl, в **bash** используются текстовые операторы для чисел и символические операторы для строк.

**Таблица 2.2. Элементарные bash-операторы сравнения**

| Строки | Числа | Истина, если |
|---|---|---|
| `x = y` | `x -eq y` | x равно y |
| `x != y` | `x -ne y` | x не равно y |
| `x < y` | `x -lt y` | x меньше y |
| `x <= y` | `x -le y` | x меньше или равно y |
| `x > y` | `x -gt y` | x больше y |
| `x >= y` | `x -ge y` | x больше или равно y |
| `-n x` | – | x не равно значению `null` |
| `-z x` | – | x равно значению `null` |

В оболочке **bash** предусмотрены возможности оценки свойств файлов (снова-таки как освобождение от наследства **/bin/test**). Некоторые из операторов тестирования и сравнения файлов приведены в табл. 2.3.

**Таблица 2.3. Некоторые bash-операторы оценки свойств файлов**

| Оператор | Истина, если |
|---|---|
| `-d` *файл* | Файл *файл* существует и является каталогом |
| `-e` *файл* | Файл *файл* существует |
| `-f` *файл* | Файл *файл* существует и является обычным |
| `-r` *файл* | У вас есть право доступа для чтения файла |
| `-s` *файл* | Файл *файл* существует и он — не пустой |
| `-w` *файл* | У вас есть право доступа для записи в файл |
| *файл1* `-nt` *файл2* | Файл *файл1* новее, чем *файл2* |
| *файл1* `-ot` *файл2* | Файл *файл1* старше, чем *файл2* |

Несмотря на всю полезность формы **elif**, зачастую лучше (с точки зрения ясности программного кода) использовать **case**-структуру выбора варианта. Ниже показан ее синтаксис на примере функции, которая централизирует процесс регистрации сообщений для сценария. Конкретные варианты описываются закрывающими скобками после каждого условия и двумя точками с запятой, завершающими блок операторов, который должен быть выполнен при реализации заданного условия. Оператор **case** завершается ключевым словом **esac**.

[6] Сейчас эти операции встроены в оболочку и уже не запускают на выполнение утилиту `/bin/test`.

```
# Уровень протоколирования устанавливается в глобальной
# переменной LOG_LEVEL. Возможные варианты перечислены в порядке
# от самого строгого до наименее строгого: Error, Warning, Info и Debug.

function logMsg {
   message_level=$1
   message_itself=$2
   if [ $message_level -le $LOG_LEVEL ]; then
       case $message_level in
          0) message_level_text="Error" ;;
          1) message_level_text="Warning" ;;
          2) message_level_text="Info" ;;
          3) message_level_text="Debug" ;;
          *) message_level_text="Other"
       esac
       echo "${message_level_text}: $message_itself"
   fi
}
```

Эта функция иллюстрирует общепринятую парадигму "уровня регистрации" (log level), используемую многими приложениями административного характера. Код этого сценария позволяет генерировать сообщения на различных уровнях детализации, но действительно регистрируются или отрабатываются только те из них, которые "проходят" глобально устанавливаемый порог `$LOG_LEVEL`. Чтобы прояснить важность каждого сообщения, его текст предваряется меткой, описывающей соответствующий уровень регистрации.

## Циклы

Конструкция `for...in` предназначена для упрощения выполнения некоторых действий для группы значений или файлов, особенно при универсализации файловых имен, т.е. замене реальных символов в имени и расширении универсальными (например "*" и "?") с целью формирования целых списков имен файлов. Шаблон `*.sh` в приведенном ниже цикле `for` позволяет обработать целый список совпадающих с ним (шаблоном) имен файлов из текущего каталога. Оператор `for`, проходя по этому списку, по очереди присваивает имя каждого файла переменной `$file`.

```
#!/bin/bash

suffix=BACKUP--`date +%Y%m%d-%H%M`

for script in *.sh; do
    newname="$script.$suffix"
    echo "Copying $script to $newname..."
    cp $script $newname
done
```

Результат выполнения этого сценария таков.

```
$ sh forexample
Copying rhel.sh to rhel.sh.BACKUP--20091210-1708...
Copying sles.sh to sles.sh.BACKUP--20091210-1708...
...
```

В раскрытии имени файла здесь нет ничего магического; все работает в точном соответствии с тем, как написано в командной строке. Другими словами, сначала имя файла

раскрывается (т.е. шаблон заменяется существующим именем), а затем уж обрабатывается интерпретатором в развернутом виде.[7]

С таким же успехом вы могли бы ввести имена файла статически, как показано в этой командной строке.

```
for script in rhel.sh sles.sh; do
```

В действительности любой список имен, содержащих пробельные символы (включая содержимое переменной), обрабатывается как объект циклической конструкции `for...in`.

Оболочке **bash** также присуща близость к циклу `for` из традиционных языков программирования, в которых задается стартовое выражение, инкрементация и условие окончания цикла.

```
for (( i=0 ; i < $CPU_COUNT ; i++ )); do
   CPU_LIST="$CPU_LIST $i"
done
```

На примере следующего сценария иллюстрируется **bash**-цикл `while`, который часто применяется для обработки аргументов командной строки и чтения строк файла.

```
#!/bin/bash

exec 0<$1
counter=1
while read line; do
  echo "$counter: $line"
  $((counter++))
done
```

Вот как выглядит результат выполнения этого сценария.

```
ubuntu$ sh whileexample /etc/passwd
1: root:x:0:0:Superuser:/root:/bin/bash
2: bin:x:1:1:bin:/bin:/bin/bash
3: daemon:x:2:2:Daemon:/sbin:/bin/bash
...
```

В этом сценарии есть интересные особенности. Оператор `exec` переопределяет стандартный выходной поток сценария так, чтобы входные данные считывались из файлов, имена которых должны определяться в соответствии с первым аргументом командной строки.[8] Если окажется, что такой файл не существует, данный сценарий сгенерирует ошибку.

Оператор `read` в конструкции `while` на самом деле является встроенным в оболочку, но действует подобно внешней команде. Внешние команды можно также помещать в конструкцию `while`. Цикл `while` в такой форме завершится тогда, когда внешняя команда возвратит ненулевое значение кода завершения.

Выражение `$((counter++))` выглядит несколько странно. Обозначение `$((...))` говорит о вычислении выражения. Кроме того, оно делает необязательным использование символа `$` для обозначения имен переменных. Удвоенный знак "плюс" (++) — это оператор постинкремента, знакомый, например, по языку C. Он возвращает значение

[7] Точнее, небольшая доля магии в раскрытии имен файлов здесь все же имеет место и состоит в поддержке понятия атомарности каждого имени файла. Имена файлов, которые включают пробелы, пройдут этот цикл `for` в один проход.

[8] В зависимости от вызова, оператор `exec` также может иметь и такое значение: "остановить этот сценарий и передать управление другому сценарию или выражению". В этом состоит еще одна странность оболочки: доступ к обеим упомянутым здесь функциям реализуется через один и тот же оператор.

переменной, с которой он связан, но также имеет побочный эффект, который состоит в приращении значения этой переменной.

Выражения `$((...))` работают в контексте двойных кавычек, поэтому тело рассмотренного выше цикла можно свернуть до одной строки.

```
while read line; do
  echo "$((counter++)): $line"
done
```

## Массивы и арифметика

Сложные структуры данных и вычисления нельзя отнести к сильной стороне оболочки `bash`. Но, по крайней мере, вы можете надеяться на использование массивов и арифметических операций.

Все `bash`-переменные представляют собой строковые значения, поэтому оболочка `bash` не делает различия в присваиваниях между числом 1 и символьной строкой "1". Различие лежит в использовании переменных. Следующий код иллюстрирует это различие.

```
#!/bin/bash

a=1
b=$((2))

c=$a+$b
d=$(($a+$b))

echo "$a + $b = $c \t(знак плюс как строковый литерал)"
echo "$a + $b = $d \t(знак плюс как арифметическое сложение)"
```

При выполнении этого сценария получим такой результат.

```
1 + 2 = 1+2 (знак плюс как строковый литерал)
1 + 2 = 3 (знак плюс как арифметическое сложение)
```

Обратите внимание на то, что знак "плюс" в присваивании переменной `$c` не работает как оператор конкатенации для строк. Он остается всего лишь литеральным символом, и посему эта строка эквивалентна следующей.

```
c="$a+$b"
```

Для того чтобы добиться вычисления, необходимо заключить выражение в двойные скобки: `$((...))`, как показано выше в присваивании переменной `$d`. Но даже эта мера предосторожности не позволяет получить в переменной `$d` числового значения; это значение по-прежнему хранится в виде строки "3".

В оболочке **bash** реализован обычный ассортимент операторов: арифметических, логических и отношений (подробнее см. соответствующие man-страницы).

Массивы в командной оболочке **bash** могут показаться немного странными объектами (да они и используются не очень часто). Тем не менее при необходимости их можно применять. Литеральные массивы ограничиваются круглыми скобками, а отдельные элементы разделяются пробельными символами. Для включения литеральных пробелов в элемент можно использовать кавычки.

```
example=(aa 'bb cc' dd)
```

Для доступа к отдельным элементам массива используйте выражение `${`*имя_массива*`[`*индекс*`]}`. Индексация начинается с нуля. Такие индексы, как "*" и "@",

относятся к массиву в целом, а специальные конструкции `${#имя_массива[*]}` и `${#имя_массива[@]}` возвращают количество элементов в массиве. Не спутайте эти выражения с конструкцией `${#имя_массива}` — и хотя эта форма кажется более логичной, но в действительности она содержит указатель на первый элемент массива (эквивалент для `${#имя_массива[0]}`).

Можно подумать, что выражение `$example[1]` должно служить однозначной ссылкой на второй элемент массива, но оболочка `bash` интерпретирует эту строку так: `$example` (обозначение ссылки на `$example[0]`) плюс литеральная строка `[1]`. Отсюда вывод: ссылаясь на переменные массива, всегда используйте фигурные скобки (без каких-либо исключений).

Рассмотрим сценарий, который иллюстрирует некоторые особенности `bash`-массивов и подводные камни, на которые можно наткнуться при управлении ими.

```
#!/bin/bash

example=(aa 'bb cc' dd)
example[3]=ee

echo "example[@] = ${example[@]}"
echo "Массив example содержит ${#example[@]} элементов"

for elt in "${example[@]}"; do
    echo "  Элемент = $elt"
done
```

Вот как выглядит результат выполнения этого сценария.

```
$ sh arrays
example[@] = aa bb cc dd ee
Массив example содержит 4 элемента
  Элемент = aa
  Элемент = bb cc
  Элемент = dd
  Элемент = ee
```

Этот пример кажется простым, но лишь потому, что мы организовали его "хорошее поведение". Но если потерять бдительность, вы можете попасть в ловушку. Например, сценарий с заменой `for`-строки вариантом

```
for elt in ${example[@]}; do
```

(т.е. без кавычек, в которых заключено выражение массива) также работает, но вместо четырех элементов он выведет пять: `aa`, `bb`, `cc`, `dd` и `ee`.

Важно помнить, что все **`bash`**-переменные все равно остаются строками, поэтому работа с массивами — в некотором роде иллюзия. В тонкостях, связанных с тем, когда и как строки разбиваются на элементы, можно утонуть. Для того чтобы не рисковать, лучше используйте язык Perl или Python. Настойчивые читатели, желающие разобраться в нюансах, могут с помощью Google обратиться к руководству Менделя Купера (Mendel Cooper) *Advanced Bash-Scripting Guide*.

## 2.3. Регулярные выражения

Регулярные выражения поддерживаются большинством современных языков, хотя одни языки "принимают их ближе к сердцу", чем другие. Они также используются в таких UNIX-командах, как **`grep`** и **`vi`**. Регулярные выражения настолько распространены,

что их название в оригинальной литературе часто сокращали до слова "regex" (*reg*ular *ex*pressions). О том, как использовать их немалые возможности, написаны уже целые книги, и неудивительно, что они стали объектом исследования многочисленных докторских диссертаций.

Сопоставление имен файлов и их раскрытие, выполненное оболочкой в процессе интерпретации таких командных строк, как `wc -l *.pl`, *не является* формой сопоставления с регулярным выражением. Это совсем другая система, именуемая "универсализацией файловых имен с помощью оболочки" (shell globbing), в которой используется другой, причем более простой, синтаксис.

Регулярные выражения — очень мощный инструмент, но они не могут распознавать все возможные грамматики. Их самый большой недостаток состоит в том, что они не в состоянии различать вложенные ограничители. Например, невозможно написать регулярное выражение, которое распознает допустимые арифметические выражения для ситуации, когда для группирования членов выражения разрешено использовать круглые скобки.

Регулярные выражения в мощности и совершенстве достигли кульминации в языке Perl. Функции сопоставления шаблонов в Perl настолько тщательно разработаны, что в действительности не совсем правильно называть их реализацией регулярных выражений. Шаблоны Perl могут включать вложенные разделители, они могут распознавать палиндромы (перевертни или перевертыши) и сопоставляться, например, с произвольной строкой букв "А", за которыми последует такое же количество букв "Б", — словом, пойдут в ход всевозможные вариации на тему регулярных выражений. При этом в Perl можно обрабатывать и довольно примитивные регулярные выражения.

Язык сопоставления Perl-шаблонов остается промышленным мерилом качества, и он принят на вооружение во многих других языках и инструментах. Библиотека Филиппа Гейзела (Philip Hazel) PCRE (*P*erl-*c*ompatible *r*egular *e*xpression) несколько упрощает для разработчиков включение этого языка в проекты.

Сами по себе регулярные выражения не являются частью языка написания сценариев, но они настолько полезны, что заслуживают подробного рассмотрения в любом обсуждении "сценарной" темы, а уж тем более в этом разделе.[9] Здесь мы рассматриваем их в базовой форме с учетом некоторых Perl-нюансов.

## Процесс сопоставления

Код, который использует регулярное выражение, пытается сопоставить одну заданную текстовую строку с одним заданным шаблоном. Сопоставляемая текстовая строка может иметь очень большой размер и содержать встроенные символы перехода на новую строку. Зачастую регулярное выражение удобно использовать для сопоставления с содержимым целого файла или HTML-документа.

Для того чтобы обнаружитель совпадений мог доложить о благоприятном исходе, шаблон поиска должен целиком совпасть с непрерывной областью исследуемого текста. При этом шаблон может совпасть в любом месте исследуемого текста. После обнаружения успешного совпадения вычислитель возвращает текст совпадения вместе со

[9] Известный знаток Perl, Том Кристиансен (Tom Christiansen) сказал: "Я не знаю, что такое язык написания сценариев, но я согласен с тем, что регулярные выражения нельзя отнести ни к процедурным, ни к функциональным языкам. Скорее, это язык, основанный на логике, или декларативный язык, относящийся к группе языков, в которую также входят Prolog и Makefile. И формы Бэкуса-Наура (BFS). Его также можно назвать языком, основанным на системе правил. Лично я предпочитаю называть подобные языки декларативными".

списком совпадений для любых частей (подразделов) шаблона, ограниченных специальным образом.

## Литеральные символы

В общем случае символы регулярного выражения при сопоставлении должны соответствовать самим себе. Так, шаблон

```
I am the walrus
```

совпадает со строкой "I am the walrus" и только с этой строкой. Поскольку совпадение может быть обнаружено в любом месте исследуемого текста, этот шаблон может дать успешный результат совпадения со строкой "I am the egg man. I am the walrus. Koo koo ka-choo!" При этом реальное совпадение ограничено фрагментом "I am the walrus". Процесс сопоставления чувствителен к регистру букв.

## Специальные символы

В табл. 2.4 собраны значения некоторых распространенных специальных символов, которые могут использоваться в регулярных выражениях (вообще, этих специальных символов гораздо больше).

**Таблица 2.4. Распространенные специальные символы регулярных выражений**

| Символ | С чем совпадает или что делает |
|---|---|
| `.` | Совпадает с любым символом |
| `[chars]` | Совпадает с любым символом из заданного набора |
| `[^chars]` | Совпадает с любым символом не из заданного набора |
| `^` | Совпадает с началом строки |
| `$` | Совпадает с концом строки |
| `\w` | Совпадает с любым алфавитно-цифровым символом (аналогично набору `[A-Za-z0-9_]`) |
| `\s` | Совпадает с любым пробельным символом (аналогично набору `[ \f\t\n\r]`)[a] |
| `\d` | Совпадает с любой цифрой (аналогично набору `[0-9]`) |
| `\|` | Совпадает с элементом либо слева, либо справа |
| `(expr)` | Ограничивает область действия, группирует элементы, позволяя формировать группы совпадений с "захватом" и без "захвата" |
| `?` | Позволяет одно повторение (или ни одного) предшествующего элемента |
| `*` | Позволяет множество повторений (или ни одного) предшествующего элемента |
| `+` | Позволяет одно или больше повторений предшествующего элемента |
| `{n}` | Ровно `n` повторений предшествующего элемента |
| `{min, }` | Не менее `min` повторений (обратите внимание на запятую) |
| `{min,max}` | Любое число (от `min` до `max`) повторений |

[a] пробел, символ прогона страницы, табуляции, новой строки или возврата каретки.

Многие такие специальные конструкции, как `+` и `|`, оказывают влияние на сопоставление подстрок слева или справа. В общем случае подстроку может составлять один символ, или "подшаблон", заключенный в круглые скобки, или класс символов, заключенный в квадратные скобки. Однако для символа "`|`" подстрока распространяется нео-

граниченно как влево, так и вправо. Если вы хотите ограничить область действия вертикальной черты, заключите ее и обе подстроки в их собственный набор круглых скобок. Например, шаблон

```
I am the (walrus|egg man)\.
```

даст совпадение либо со строкой "I am the walrus.", либо со строкой "I am the egg man.". Этот пример демонстрирует также способ "экранирования" специальных символов (в данном случае точки) с помощью обратной косой черты (\). Шаблон

```
(I am the (walrus|egg man)\. ?){1,2}
```

совпадет с любой из следующих строк.

- `I am the walrus.`
- `I am the egg man.`
- `I am the walrus. I am the egg man.`
- `I am the egg man. I am the walrus.`

К сожалению, он также даст совпадение со строкой "I am the egg man. I am the egg man.". (И какой в этом смысл?) Важнее то, что приведенный выше шаблон также совпадет со строкой "I am the walrus. I am the egg man. I am the walrus.", хотя количество повторений явно ограничено числом два. Дело в том, что шаблон не обязательно должен совпадать полностью с исследуемым текстом. В данном случае после обнаружения совпадений этого регулярного выражения с двумя предложениями дальнейшее его сопоставление прекратилось с объявлением об успешном завершении. После выполнения условия, заданного в регулярном выражении, уже не важно, что в данном тексте есть еще одно совпадение с шаблоном.

Часто метасимвол регулярного выражения "*" (квантификатор, равный нулю или больше нуля) путают с символом "*", используемым командной оболочкой для универсализации файловых имен. В системе регулярных выражений используйте метасимвол "*" в случае, если для совпадения приемлема любая последовательность символов (включая их отсутствие).

## Примеры использования регулярных выражений

В Соединенных Штатах почтовые индексы (zip-коды) имеют либо пять цифр, либо пять цифр с последующим тире и еще четырьмя цифрами. При создании регулярного выражения для zip-кода необходимо обеспечить сопоставление пятизначного номера. Следующее регулярное выражение отвечает всем этим требованиям.

```
^\d{5}$
```

Символы "^" и "$" сопоставляются с началом и концом обрабатываемого текста, не соответствуя при этом реальным символам в тексте; они представляют собой "утверждения нулевой длины" (т.е. это не символы, а лишь "позиции" в тексте). Эти символы гарантируют, что только те текстовые строки, которые состоят ровно из пяти цифр, должны совпадать с регулярным выражением (строки большей длины не должны давать совпадения). Сочетание символов `\d` обеспечивает совпадение с цифрой, а квантификатор `{5}` выражает требование совпадения ровно пяти цифр.

Для того чтобы можно было выявить либо пятизначный почтовый индекс, либо расширенный zip-код (zip+4), добавим в качестве необязательной части тире и четыре дополнительные цифры.

```
^\d{5}(-\d{4})?$
```

Круглые скобки объединяют в группу тире и дополнительные цифры, чтобы они считались одной необязательной единицей. Например, это регулярное выражение не даст совпадения с пятизначным почтовым индексом, за которым следует "голое" тире (без последующих цифр). Если тире существует, также должно существовать четырехзначное расширение, в противном случае совпадение зафиксировано не будет.

Классический пример использования регулярного выражения демонстрирует следующее выражение.

```
M[ou]'?am+[ae]r ([AEae]l[- ])?[GKQ]h?[aeu]+([dtz][dhz]?)+af[iy]
```

Оно дает совпадения со многими вариантами произношения имени ныне покойного лидера Ливии Муаммара Каддафи (Moammar Gadhafi), например, с такими, как:

- Muammar al-Kaddafi (BBC);
- Moammar Gadhafi (Associated Press);
- Muammar al-Qadhafi (Al-Jazeera);
- Mu'ammar Al-Qadhafi (U.S. Department of State).

Вы понимаете, почему каждый из приведенных вариантов успешно совпал с шаблоном?

Приведенное выше регулярное выражение также иллюстрирует, насколько быстро могут быть достигнуты пределы удобочитаемости. Многие системы регулярных выражений (включая используемую в Perl) поддерживают опцию x, которая игнорирует литеральные пробельные символы в шаблоне и легитимизирует комментарии, разрешая "растягивать" шаблон и располагать его на нескольких строках. Затем вы можете использовать пробельные символы для выделения логических групп и "прояснения отношений" между ними подобно тому, как это делается в процедурных языках.

```
M [ou] '? a m+ [ae] r    # Имя: Mu'ammar, Moamar и т.д.
\s                       # Пробельный символ; здесь нельзя
                         # использовать литеральный пробел
(                        # Группа для необязательного префикса-фамилии
  [AEae] l               # Al, El, al или el
  [-\s]                  # Тире или пробел
)?
[GKQ] h? [aeu]+          # Начальный ñëîã фамилии: Kha, Qua и т.д.
(                        # Группа для согласных в начале 2-го слога
  [dtz] [dhz]?           # dd, dh и т.д.
)+
af [iy]
```

Такой способ описания регулярного выражения, вероятно, слегка облегчит понимание, но все же и он в состоянии замучить потенциальных читателей. Поэтому лучше использовать иерархический подход и строить множество небольших сопоставлений вместо попытки описать все возможные ситуации в одном огромном регулярном выражении.

## Захваты

В случае, если совпадение происходит, каждый набор круглых скобок становится "группой захвата", которая фиксирует реально совпавший текст. То, как именно эти элементы становятся доступными для вас, зависит от конкретной реализации и контекста. В Perl доступ к результатам можно получить в виде списка или последовательности пронумерованных переменных.

Поскольку круглые скобки могут быть вложенными, как узнать, чему соответствует каждое совпадение? Ответ простой: совпадения выстраиваются в результате в том же порядке, в котором располагаются открывающие скобки. Количество "захватов" равно количеству открывающих скобок, независимо от роли (или ее отсутствия), которую играла каждая взятая в скобки группа в реальном совпадении. Если взятая в скобки группа не используется (например, при сопоставлении регулярного выражения `Mu(')?ammar` со строкой "Muammar"), соответствующий "захват" будет пустым.

Если обнаружится несколько совпадений группы, в качестве результата возвращается содержимое только последнего совпадения. Например, при шаблоне

```
(I am the (walrus|egg man)\. ?){1,2}
```

и таком варианте обрабатываемого текста

```
I am the egg man. I am the walrus.
```

существует два результата (по одному для каждого набора круглых скобок).

```
I am the walrus.
Walrus
```

Обратите внимание на то, что обе захваченные группы в действительности совпали дважды. Но был захвачен только последний текст, совпавший с каждым набором круглых скобок.

## Жадность, леность и катастрофический поиск с возвратом

Регулярные выражения сопоставляются слева направо. Если каждый компонент шаблона перед переходом к следующему компоненту стремится "захватить" максимально возможную строку, то такая характеристика поведения называется *жадностью* (greediness).

Если обработчик регулярных выражений (анализатор) достигает состояния, из которого сопоставление выполнить уже невозможно, он немного "откатывается" назад от кандидата на совпадение и заставляет одного из "жадных" компонентов отказаться от части текста. Например, рассмотрим регулярное выражение `a*aa` применительно к входному тексту "аааааа".

Вначале анализатор присваивает весь входной текст компоненту `a*`, поскольку он (т.е. `a*`) является "жадным". Когда окажется, что больше букв "а" не осталось, анализатор перейдет к поиску совпадений со следующим компонентом регулярного выражения. Но поскольку им является `a` и во входном тексте больше нет ничего, что можно было бы сопоставить с `a`, пора делать "откат". Компонент `a*` вынужден отказаться от одной из букв "а", с которыми уже было зафиксировано совпадение.

Теперь анализатор может сопоставлять компонент `a*a`, но он по-прежнему не может это сделать для последней буквы "а" в шаблоне. Поэтому он снова откатывается и "отнимает" вторую букву "а" из "добычи" компонента `a*`. На этот раз вторая и третья буквы "а" в шаблоне имеют буквы "а" для фиксации совпадения, и на этом обработка текста завершена. Этот простой пример иллюстрирует некоторые важные общие моменты. Прежде всего, "жадное" сопоставление с "откатами" делает использование таких простых шаблонов, как `<img.*></tr>`, дорогим удовольствием при обработке целых

файлов.[10] Действие порции `.*` начинается с сопоставления всего содержимого от первой найденной подстроки `<img` до конца входного текста, и только благодаря повторным откатам область поиска сузится, чтобы "подобраться" к локальным тегам.

Более того, порция `></tr>`, с которой связан этот шаблон, — это *последнее возможное* допустимое вхождение искомого элемента во входном тексте, что, наверняка, совсем не отвечает вашим намерениям. Вероятно, вы хотели отыскать совпадение с подстрокой `<img>`, за которой следует тег `</tr>`. Тогда лучше переписать этот шаблон в виде `<img[^>]*></tr>`, что позволит распространить совпадение с начальными групповыми символами только до конца текущего тега благодаря явно заданной невозможности пересечь границу, обозначенную правой угловой скобкой.

Можно также использовать "ленивые" (в противоположность "жадным") операторные выражения: `*?` вместо `*` и `+?` вместо `+`. Эти варианты квантификаторов ограничивают количество вхождений искомых символов во входном тексте до минимально возможного. Во многих ситуациях эти операторы работают эффективнее и дают результаты, более близкие к желаемым, чем "жадные" варианты.

Однако обратите внимание на то, что "ленивые" квантификаторы (в отличие от "жадных") могут давать различные совпадения; разница состоит не просто в используемой реализации. В нашем HTML-примере "ленивый" шаблон выглядел бы так: `<img.*?></tr>`. Но даже в этом случае элемент `.*?` мог бы стать причиной внесения в результат ненужных нам символов ">", поскольку следующим после тега `<img>` может быть не тег `</tr>`. А такой результат вас, скорее всего, не устроит.

Шаблоны с несколькими секциями групповых символов могут "спровоцировать" анализатор регулярных выражений на экспоненциальное поведение, особенно в случае, если порции текста могут совпадать с несколькими шаблонными выражениями, и тем более в случае, если искомый текст в действительности не соответствует шаблону. Эта ситуация не является такой уж необычной, как может показаться, главным образом тогда, когда шаблон сопоставляется с HTML-кодом. Очень часто вам придется искать конкретные теги, за которыми следуют другие теги, возможно, разделенные третьими тегами, и так далее, одним словом, вам придется создавать задание, которое потребует, чтобы анализатор проверил массу возможных комбинаций.

Большой специалист в области регулярных выражений Ян Гойвертс (Jan Goyvaerts) называет этот эффект *катастрофическим откатом* (catastrophic backtracking) и описывает его в своем блоге (за подробностями и некоторыми удачными решениями обращайтесь по адресу: `regular-expressions.info/catastrophic.html`).

Из всего сказанного выше можно сделать такие выводы.

- Если вы можете организовать построчное сопоставление шаблона, а не пофайловое (т.е. с просмотром сразу всего файла), значит, вы значительно уменьшаете риск получения, мягко говоря, низкой производительности.
- Несмотря на то что регулярные выражения являются "жадными" по умолчанию, вам, скорее всего, такие не нужны. Используйте "ленивые" операторы.

[10] Несмотря на то что в этом разделе в качестве примеров обрабатываемого текста показаны фрагменты HTML-кода, регулярные выражения — не самый подходящий инструмент в данном случае (что не преминули отметить и наши рецензенты). Языки Perl и Python имеют прекрасные расширения, которые анализируют HTML-документы надлежащим образом. Вы можете получить доступ к интересующим вас порциям с помощью Xpath-селекторов. Подробнее о модульных репозиториях соответствующих языков можно узнать, обратившись к странице Википедии, посвященной языку запросов Xpath (XML Path Language).

- Все экземпляры специальных символов ".*" по своему существу подозрительны и должны быть тщательно исследованы.

## 2.4. Программирование на языке Perl

Язык Perl, созданный Ларри Уоллом (Larry Wall), был первым из самых популярных языков написания сценариев. Он предлагает чрезвычайно больше возможностей, чем `bash`, а написанные на нем программы (если они написаны хорошо) довольно просты для понимания. Поскольку в языке Perl не предусмотрена обязательность соблюдения разработчиками стилистических правил создания кода, читабельность Perl-кода зависит от их дисциплинированности. Не без основания Perl называют языком "только для записи".

Здесь описывается версия Perl 5, которая была стандартом в течение последнего десятилетия. Версия Perl 6 все еще находится в стадии разработки. Подробнее можно узнать на сайте `perl6.org`. Языки Perl и Python (см. раздел 2.5) больше подходят для работы в области системного администрирования, чем такие традиционные языки программирования, как C, C++, C# и Java. Их отличает большая эффективность: при лаконичности кода и менее мучительном процессе отладки, а также без компиляционной "волокиты" они могут продемонстрировать большие возможности.

Выбор языка обычно зависит от личных предпочтений или стандартов, используемых вашим работодателем. Как Perl, так и Python предлагают библиотеки модулей и языковые расширения. По популярности язык Perl "старше", чем Python, поэтому его предложения "вылились" в довольно длинный список возможностей. Но при решении каждодневных задач системного администрирования библиотеки поддержки обоих языков практически эквивалентны.

Для языка Perl характерен следующий девиз: "Достичь цель можно несколькими путями". Поэтому имейте в виду, что в большинстве примеров, описанных в этом разделе, существуют и другие (помимо приведенного здесь) способы выполнения поставленной задачи.

Операторы Perl разделяются точкой с запятой.[11] Комментарии начинаются знаком # и продолжаются до конца строки. Блоки операторов заключаются в фигурные скобки. Вот пример простой программы "Привет, мир!".

```
#!/usr/bin/perl
print "Привет, мир!\n";
```

Как и для **bash**-программ, вы должны либо изменить права доступа (**chmod +x**) для исполняемого файла, либо непосредственно вызвать Perl-интерпретатор.

```
$ chmod +x helloworld
$ ./helloworld
Привет, мир!
```

Строки в Perl-сценарии не являются командами оболочки (они представляют собой Perl-код). В отличие от командной оболочки **bash**, которая позволяет объединить некоторую последовательность команд в сценарий, Perl не "видит дальше своего носа", если ему не дать соответствующего указания. Тем не менее Perl обеспечивает действие многих таких же соглашений, которые действуют в оболочке **bash**, например использование обратных апострофов, заключающих в себя некоторую команду, для перехвата результата выполнения этой команды.

[11] Поскольку знаки "точка с запятой" являются разделителями, а не ограничителями (терминаторами), последнюю точку с запятой в блоке ставить необязательно.

## Переменные и массивы

В Perl используются три базовых типа данных: скаляры (т.е. такие унитарные значения, как числа и строки), массивы и хеши (известные также как ассоциативные массивы). Тип переменной всегда очевиден, поскольку он "встроен" в имя переменной: скалярные переменные начинаются знаком "$", переменные массивов — символом "@" и хеш-переменные — знаком "%".

В Perl понятия "список" и "массив" часто взаимозаменяемы, но, пожалуй, корректнее сказать, что список — это некоторая последовательность значений, а массив — это переменная, которая может содержать такой список. Отдельные элементы массива являются скалярами, поэтому, подобно обычным скалярным переменным, их имена начинаются со знака "$". Индексация массивов начинается с нуля, а индекс самого последнего элемента в массиве `@a` определяется значением `$#a`. Чтобы получить размер массива, достаточно добавить к этому значению единицу.

Массив `@ARGV` содержит аргументы командной строки сценария. Его можно использовать подобно любому другому массиву.

Пример использования массивов демонстрирует следующий сценарий.

```
#!/usr/bin/perl

@items = ("носки", "туфли", "шорты");
printf "Есть %d вида одежды.\n", $#items + 1;
print "Наденьте ${items[2]} первыми, затем  ", join(" и ", @items[0,1]), ".\n";
```

Результат выполнения этого сценария таков.

```
$ perl clothes
Есть 3 вида одежды.
Наденьте шорты первыми, затем носки и туфли.
```

В этих нескольких строках сценария есть много интересного. Рискуя отвлечь ваше внимание, мы все же включаем в каждый наш Perl-пример несколько распространенных идиом, разъясняя сложные моменты в тексте, следующем за примером. Если вы внимательно прочитаете все пояснения к примерам (не пугайтесь, они короткие!), то к концу этой главы приобретете практические знания в использовании большинства распространенных Perl-конструкций.

## Массив и строковые литералы

В этом примере обратите внимание, прежде всего, на то, что набор круглых скобок `(...)` создает список литералов. Отдельными элементами этого списка являются строки (которые разделяются запятыми). Созданный список присваивается переменной `@items`.

В языке Perl нет жесткого требования заключать все строки в кавычки. В конкретном случае начальное присваивание значения переменной `@items` с тем же успехом работает и без кавычек.

```
@items = (носки, туфли, шорты);
```

Такие строки без кавычек называются в Perl "голыми словами" (barewords), и они интерпретируются таковыми в качестве "последнего средства". Это значит, что, если для некоторого объекта невозможно найти смысловую нагрузку, Perl пытается интерпретировать его как строку. В некоторых случаях это имеет смысл, если сохранится читабельность кода. Однако наш пример не относится к таким случаям. Даже если вы сами пред-

почитаете всегда заключать строки в кавычки, будьте готовы к тому, что вам придется "разгадывать" намерения других программистов, разбираясь в их "голословном" коде.

В Perl существует еще один способ инициализировать этот массив. Он состоит в использовании оператора qw (*q*uote *w*ords — т.е. брать слова в кавычки). Это, по сути, форма заключения строки в кавычки, которая предоставляет вам право самим выбирать разделители (как и во многих других случаях использования кавычек в Perl). Форма

```
@items = qw(носки туфли шорты);
```

является самой традиционной, но при этом может ввести в заблуждение, поскольку часть после оператора qw — уже не список. В действительности это строка, разделенная пробельными символами для формирования списка. Версия

```
@items = qw[носки туфли шорты];
```

тоже работает и, пожалуй, точнее передает смысл происходящего. Обратите внимание на то, что запятых здесь нет, поскольку их нагрузку "взял на себя" оператор qw.

## Вызовы функций

Оба варианта — **print** и **printf** — принимают произвольное количество аргументов, которые разделяются запятыми. Но вот вы видите join(...). Не напоминает ли вам это вызов функции? Чем же "этот join" отличается от **print** и **printf**?

По сути, ничем. **print**, **printf** и **join** — это обычные функции. Perl позволяет опускать круглые скобки в вызовах функций, если эта "экономия" не послужит причиной неоднозначности, поэтому обе формы приемлемы. В приведенной выше **print**-строке форма записи со скобками (join(...)) используется для явного отделения аргументов, передаваемых функции **join**, от аргументов, передаваемых функции **print**.

Выражение @items[0,1] должно дать "списочный" результат, поскольку оно начинается с символа "@". В действительности это выражение означает "кусочек массива" (или подмассив), а список индексов 0,1 перечисляет индексы элементов исходного массива, которые должны быть включены в упомянутый "кусочек". Здесь Perl принимает диапазон значений, как и в эквивалентном выражении @items[0..1]. Здесь также было бы допустимо использовать одиночный числовой индекс: например, выражение @items[0] означало бы список, содержащий один скаляр, т.е. строку "носки", что в данном случае было бы эквивалентно литералу ("носки").

Массивы в вызовах функций автоматически раскрываются, поэтому в выражении

```
join(" и ", @items[0,1])
```

функция join получает три строковых аргумента: " и ", "носки" и "туфли". Она конкатенирует свой второй и последующие аргументы, вставляя между каждой парой копию первого аргумента. В результате получаем "носки и туфли".

## Преобразования типов в выражениях

В **printf**-строке при вычислении выражения $#items + 1 получим число 3. По определению выражение $#items содержит числовое значение, но это не дает основания считать, что выражение $#items + 1 предполагает выполнение арифметического действия; смешанный вариант "2" + 1 работает так же. "Магия" кроется в операторе "+", который всегда подразумевает арифметику, т.е. он преобразует свои аргументы в числа и генерирует числовой результат. Аналогично ему оператор "точка" (.), назначение которого — конкатенировать строки, преобразует свои операнды "по контексту", т.е. при вычислении выражения "2" . (12 ** 2) получим в результате "2144".

## Раскрытие строк и устранение неоднозначности при ссылках на переменные

Как и в командной оболочке **bash**, строки, заключенные в двойные кавычки, являются объектами для раскрытия переменных. Так же, как и в **bash**, вы можете заключить имена переменных в фигурные скобки, чтобы при необходимости устранить неоднозначность, как в выражении `${items[2]}`. (В данном случае фигурные скобки используются только для иллюстрации; на самом деле они не нужны.) Символ "`$`" информирует о том, что результат вычисления выражения должен быть скалярным. Переменная `@items` определяет массив, но его любой элемент в отдельности — это скаляр, и сей факт отражен в соглашениях о присваивании имен переменным.

## Хеши

Хеш (также именуемый ассоциативным массивом или хеш-массивом) представляет набор пар "ключ/значение". Хеш можно представить себе как массив, индексы которого (ключи) являются произвольными скалярными значениями, т.е. они необязательно должны быть числами. Но на практике в качестве ключей используются именно числа и строки.

Имена хеш-переменных начинаются с символа "`%`" (например, `%myhash`), но их отдельными значениями являются скаляры, поэтому для работы с хеш-массивами нужно, как и в обычных массивах, использовать префикс разыменования "`$`". Индексация обозначается фигурными скобками, а не квадратными, например `$myhash{'ron'}`.

Для системных администраторов хеши — это очень важный инструмент, который вы будете использовать практически в каждом сценарии. В приведенном ниже коде мы считываем содержимое файла, анализируем его в соответствии с правилами, заданными структурой файла **/etc/passwd**, и формируем хеш с именем `%names_by_uid`. Значением каждого элемента в хеше становится имя пользователя, связанное с идентификатором UID.

```
#!/usr/bin/perl

while ($_ = <>) {
   ($name, $pw, $uid, $gid, $gecos, $path, $sh) = split /:/;
   $names_by_uid{$uid} = $name;
}
%uids_by_name = reverse %names_by_uid;

print "\$names_by_uid{0}  is $names_by_uid{0}\n";
print "\$uids_by_name{'root'} is $uids_by_name{'root'}\n";
```

Как и в предыдущем примере сценария, мы "начинили" эти программные строки некоторыми новыми понятиями. Прежде чем рассматривать анонсированные нюансы, ознакомьтесь с результатом работы этого сценария.

```
$ perl hashexample /etc/passwd
$names_by_uid{0} is root
$uids_by_name{'root'} is 0
```

В каждом проходе цикла `while ($_ = <>)` считывается одна входная строка, которая присваивается переменной `$_` (напомним: значение всего оператора присваивания, как в языке C, определяется значением его правой части). При достижении конца вход-

ного потока, оператор <> возвратит значение "Ложь", что послужит причиной завершения цикла.

Интерпретируя оператор <>, Perl проверяет командную строку, чтобы узнать, указаны ли там какие-нибудь имена файлов. Если вы не забыли это сделать, Perl открывает каждый файл по очереди и "прогоняет" его содержимое через цикл. Если вы не назвали ни одного файла в командной строке, Perl возьмет входные данные для циклической обработки из стандартного входного канала.

Внутри цикла указанные в скобках переменные получают значения, возвращаемые функцией `split`, которая "разрезает" входную строку, используя регулярное выражение, переданное ей в качестве разделителя полей. Здесь наш шаблон ограничивается слешами (символами "/"), действие которых подобно двойным кавычкам. Мы могли бы с таким же успехом написать `split ':'` или `split ":"`.

Строка, которую функция `split` должна разбить на подстроки в позициях сопоставления с двоеточиями, нигде явно не задана. Если второй аргумент функции `split`, т.е. подлежащая разбиению строка, опущен, Perl предполагает, что вы хотите разбить на подстроки значение `$_`. Яснее не бывает! По правде говоря, иногда можно опустить даже шаблон — в этом случае разбиение строки производится по пробельному символу, при этом начальные пробелы в каждой подстроке игнорируются.

Но и это еще не все. Даже исходное присваивание переменной `$_` в первой строке цикла необязательно. Если просто написать так:

```
while (<>) {,
```

то Perl автоматически сохранит каждую входную строку в переменной `$_`. Можно обрабатывать строки без создания явной ссылки на переменную, в которой они будут сохраняться. Использование переменной `$_` в качестве операнда, действующего по умолчанию, — распространенная практика, и Perl позволяет такие "вольности" везде, где это имеет смысл.

Во множественном присваивании, которое "захватывает" содержимое каждого поля `passwd`, наличие списка в левой части

```
($name, $pw, $uid, $gid, $gecos, $path, $sh) = split /:/;
```

создает для функции `split` "контекст списка", что предписывает ей возвращать в результате список всех полей. Если бы в присваивании участвовала скалярная переменная, например

```
$n_fields = split /:/;,
```

функция `split` "переключилась" бы на "скалярный контекст" и вернула бы только количество обнаруженных полей. Пользовательские функции также могут отличать скаляры от списочного контекста с помощью функции `wantarray`. Она возвращает значение "Истина" в контексте списка, "Ложь" — в контексте скаляра и неопределенное значение — в `void`-контексте (для пустых типов данных).

Строка

```
%uids_by_name = reverse %names_by_uid;
```

также заслуживает внимания. Хеш в контексте списка (здесь в качестве аргумента функции `reverse`) приводится к списку в форме (*`key1, value1, key2, value2, ...`*). Функция `reverse` меняет порядок следования элементов в списке на обратный: (*`valueN, keyN, ..., value1, key1`*). Наконец, присваивание хеш-переменной `%uids_by_name` преобразует этот список в вариант (*`key1, value1, ...`*), создавая тем самым перестановочный индекс.

## Ссылки и их самооживление

Обозначенная в заголовке тема относится к более сложным, но мы посчитали себя не вправе обойти ее молчанием. По крайней мере, мы должны изложить здесь основные положения. Массивы и хеши могут хранить скалярные значения, но не исключено, что вам понадобится сохранить в них другие массивы и хеши. Например, возвращаясь к нашему предыдущему примеру анализа файла **/etc/passwd**, вы могли бы сохранить все поля каждой строки **passwd** в хеше, индексируемом значением идентификатора UID.

Если нельзя хранить массивы и хеши, то можно хранить ссылки (указатели) на массивы и хеши, которые сами по себе являются скалярами. Чтобы создать ссылку на массив или хеш, необходимо предварить имя соответствующей переменной обратной косой чертой (например, `\@array`) или использовать литеральный синтаксис ссылок на массив или на хеш. Например, наш цикл passwd-анализа мог выглядеть так.

```
while (<>) {
    $array_ref = [ split /:/ ];
    $passwd_by_uid{$array_ref->[2]} = $array_ref;
}
```

Квадратные скобки возвращают ссылку на массив, содеращий результаты работы функции `split`. Выражение `$array_ref->[2]` ссылается на поле UID (это третий член массива, адресуемого переменной `$array_ref`).

Выражение `$array_ref[2]` здесь не работает, поскольку мы не определили массив `@array_ref`; `$array_ref` и `@array_ref` — это разные переменные. Хуже того, вы не получите сообщение об ошибке, если ошибочно использовали здесь выражение `$array_ref[2]`, поскольку `@array_ref` — это абсолютно легитимное имя для массива; вы ведь не присваивали ему никаких значений.

Отсутствие предупреждающих сообщений может показаться проблемой, но это, возможно, одно из лучших качеств Perl, именуемое *самооживлением* (autovivification). Поскольку имена переменных и синтаксис использования ссылок всегда делают довольно ясной структуру данных, к которой вы пытаетесь получить доступ, вам не придется создавать промежуточные структуры данных вручную. Достаточно организовать присваивание на самом низком уровне, и промежуточные структуры возникают автоматически. Например, используя только один оператор присваивания, можно создать хеш ссылок на массивы, в качестве содержимого которых будут выступать ссылки на хеши.

## Регулярные выражения в Perl

Регулярные выражения в Perl используются путем "привязки" строк к операциям над регулярными выражениями, а именно с помощью оператора связывания `=~`. Например, при выполнении кода

```
if ($text =~ m/ab+c/) {
```

проверяется, совпадает ли строка, хранимая в переменной `$text`, с регулярным выражением `ab+c`. Для выполнения действий по умолчанию над строкой, в которой хранится результат предыдущей операции, т.е. `$_`, можно просто опустить имя переменной и оператор связывания. На самом деле можно опустить и оператор `m`, поскольку эта операция по умолчанию приводит к совпадению.

```
if (/ab+c/) {
```

Подстановки работают аналогично.

```
$text =~ s/etc\./and so on/g;  # Замена текста в строке $text, ИЛИ
s/etc\./and so on/g;           # Применительно к переменной $_
```

Мы вставили здесь модификатор `g`, чтобы заменить не просто первый экземпляр подстроки "etc." вместе с подстрокой "and so on", а все найденные экземпляры. Помимо модификатора `g` (который применяется для поиска множественных совпадений), часто используются и другие: модификатор `i`, чтобы игнорировать регистр букв; модификатор `s`, чтобы "нацелить" точку (`.`) на совпадение с символами новой строки; и модификатор `m`, чтобы с помощью маркеров "^" и "$" искать совпадения в начале и конце отдельных строк, а не только в начале и конце обрабатываемого текста.

Для того чтобы обратить ваше внимание на другие важные нюансы, рассмотрим следующий сценарий.

```
#!/usr/bin/perl

$names = "huey dewey louie";
$regex = '(\w+)\s+(\w+)\s+(\w+)';

if ($names =~ m/$regex/) {
   print "1-ое имя $1.\n2-ое имя $2.\n3-е имя $3.\n";
   $names =~ s/$regex/\2 \1/;
   print "Новые имена \"${names}\".\n";
} else {
   print qq{"$names" не совпадает с "$regex".\n};
}
```

Вот как выглядит результат выполнения этого сценария.

```
$ perl testregex
1-ое имя huey.
2-ое имя dewey.
3-е имя louie.
Новые имена "dewey huey".
```

Этот пример показывает, что символы `//` позволяют раскрыть переменные, т.е. регулярное выражение необязательно должно быть фиксированной строкой. Обратите внимание на оператор `qq` — это еще одно имя для оператора заключения в двойные кавычки.

После реализации совпадения или подстановки переменные `$1`, `$2` и так далее соотносятся с содержимым круглых скобок в регулярном выражении. Содержимое этих переменных также доступно в самом процессе замены, в контексте которого они "именуются" как `\1`, `\2` и т.д.

## Ввод и вывод данных

Открывая файл для чтения или записи, вы определяете так называемый *дескриптор файла*, чтобы идентифицировать соответствующий канал. В приведенном ниже примере переменная `INFILE` служит дескриптором для файла **`/etc/passwd`**, а переменная `OUTFILE` — дескриптором, связанным с файлом **`/tmp/passwd`**. Используемое в цикле `while` условие `<INFILE>` аналогично рассмотренному выше выражению `<>`, но имеет свою специфику, связанную с данным конкретным дескриптором файла. Здесь считываются строки из файла с дескриптором `INFILE` до тех пор, пока не обнаружится конец файла, что и остановит работу цикла `while`. Каждая строка размещается в переменной `$_`.

```
#!/usr/bin/perl

open(INFILE, "</etc/passwd") or die  "Не удается открыть /etc/passwd";
open(OUTFILE, ">/tmp/passwd") or die "Не удается открыть /tmp/passwd";

while (<INFILE>) {
    ($name, $pw, $uid, $gid, $gecos, $path, $sh) = split /:/;
     print OUTFILE "$uid\t$name\n";
}
```

Функция `open` возвратит значение "Истина", если файл успешно откроется, тем самым "замкнув" (т.е. сделав необязательным) вычисление частей `die`, поскольку результат вычисления всего выражения уже не изменится. Perl-оператор `or` подобен оператору `||` (который также есть в Perl), но с более низким приоритетом. Оператор `or` в общем случае предпочтительнее, если вы хотите подчеркнуть, что прежде, чем Perl переключит "свое внимание" на последствия неудачного исхода операции, должны быть выполнены все действия, указанные в левой части выражения.

Синтаксис Perl для задания способа использования каждого файла (для чтения, записи или добавления в конец) отражает возможности, заложенные в командной оболочке. Кроме того, использование таких имен файлов, как `"/bin/df|"`, позволит открывать входные или выходные каналы связи с командами оболочки.

## Поток управления

В приведенном ниже примере представлена Perl-версия уже знакомого вам **bash**-сценария, в котором проверялась достоверность аргументов командной строки (для сравнения можете вернуться к разделу "Функции и аргументы командной строки" этой главы). Обратите внимание на то, что Perl-конструкция `if` не оснащена ни ключевым словом `then`, ни каким-либо признаком завершения — блок операторов здесь просто заключен в фигурные скобки.

Для того чтобы обеспечить выполнение любого отдельного оператора при соблюдении некоторого условия, вы можете добавлять постфиксный вариант оператора `if` (или его обратную версию `unless`).

```
#!/usr/bin/perl

sub show_usage {
   print shift, "\n" if scalar(@_);
   print "Применение: $0 source_dir dest_dir\n";
   exit scalar(@_) ? shift : 1;
}
if (@ARGV != 2) {
   show_usage;
} else { # Существует два аргумента
  ($source_dir, $dest_dir) = @ARGV;
  show_usage "Недопустимый каталог-источник" unless -d $source_dir;
  -d $dest_dir or show_usage "Недопустимый каталог-приемник";
}
```

Здесь две строки, которые используют унарный Perl-оператор `-d` для проверки существования каталогов `$source_dir` и `$dest_dir`, эквивалентны. Вторая форма (с оператором `-d` в начале строки) предпочтительнее, поскольку утверждение используется в начале строки, где оно заметнее всего. Однако применение оператора `or`, обозначающего "в противном случае", несколько сомнительно; и многих, кому приходится разбираться в чужом коде, это сбивает с толку.

Переменная массива в скалярном контексте (заданном оператором `scalar` в данном примере) возвращает количество элементов в массиве. Это значение на единицу превышает значение `$#array` (как всегда, в Perl предусмотрено несколько способов получения одного и того же результата).

Функции в Perl принимают аргументы в массив с именем `@_`. Обычной практикой считается в Perl получать доступ к ним с помощью оператора `shift`, который удаляет первый элемент массива аргументов и возвращает его значение.

Эта версия функции `show_usage` принимает необязательное сообщение об ошибке для вывода на печать. Если вы предоставляете сообщение об ошибке, то можете также предоставить специфический код выхода. Тернарный оператор `?:` оценивает свой первый аргумент; если оценка даст значение "Истина", то результатом всего выражения станет значение второго аргумента, в противном случае — третьего.

Как и в среде `bash`, Perl имеет специальное условие "else if ", но его ключевым словом является `elsif`, а не `elif`. (Для тех, кто использует оба языка, эти забавные пустяковые различия либо заострят их ум, либо — наоборот.)

Как показано в табл. 2.5, Perl-операторы сравнения не совпадают с аналогичными операторами в оболочке **bash**: строки используют текстуальные операторы, а числа — традиционную алгебраическую запись. Сравните эту таблицу с табл. 2.2.

**Таблица 2.5. Элементарные Perl-операторы сравнения**

| Строки | Числа | Истина, если |
|---|---|---|
| `x eq y` | `x = y` | x равно y |
| `x ne y` | `x != y` | x не равно y |
| `x lt y` | `x < y` | x меньше y |
| `x le y` | `x <= y` | x меньше или равно y |
| `x gt y` | `x > y` | x больше y |
| `x ge y` | `x >= y` | x больше или равно y |

В Perl у вас есть все операторы тестирования файлов, показанные в табл. 2.3, за исключением операторов `-nt` и `-ot`, которые доступны только в среде **bash**.

Подобно оболочку **bash**, язык Perl предоставляет два типа цикла `for`. При использовании более распространенной формы цикла `for` обеспечивается выполнение итераций с проходом по всем элементам явно заданного списка аргументов. Например, в приведенном ниже коде перебираются по очереди элементы списка названий животных (`animals`) и выводятся по одному на строке.

```
@animals = qw(lions tigers bears);
foreach $animal (@animals) {
   print "$animal \n" ;
}
```

Также вы вправе использовать цикл `for` в стиле языка C.

```
for ($counter=1; $counter <= 10; $counter++) {
    printf "$counter ";
}
```

Хотя мы продемонстрировали применение циклов с разными ключевыми словами — `for` и `foreach`, на самом деле в обоих случаях работает одно и то же ключевое слово, т.е. в Perl вы можете использовать любую форму, которая кажется вам предпочтительнее.

В версиях языка Perl до 5.10 (2007 г.) не было явно определенных операторов case или switch, но, как уже подчеркивалось не раз, один результат можно получить несколькими путями. В дополнение к очевидному, но громоздкому варианту каскадирования операторов if, можно прибегнуть к еще одному способу использования оператора for, в котором главное — установить переменную $_, сравнить ее с нужными значениями и после выполнения соответствующих действий обеспечить выход из контекста (с помощью оператора last).

```
for ($ARGV[0]) {

    m/^websphere/ && do { print "Инсталляция для websphere\n"; last; };
    m/^tomcat/    && do { print "Инсталляция для tomcat\n" ; last; };
    m/^geronimo/  && do { print "Инсталляция для geronimo\n"; last; };

    print "Предоставлен недопустимый вариант.\n"; exit 1;
}
```

Здесь регулярные выражения по очереди сравниваются с аргументом, сохраненным в переменной $_. При неуспешном результате сравнения в левой части операции && правая часть уже не вычисляется и управление передается следующему оператору. Если же происходит совпадение с некоторым регулярным выражением, выполняется соответствующий ему блок do, после чего остается лишь "красиво уйти". Немедленный выход из блока for обеспечивают операторы last.

## Прием входных данных и проверка их достоверности

В приведенном ниже сценарии объединены многие уже рассмотренные нами конструкции Perl, включая использование подпрограммы, нескольких постфиксных операторов if и цикла for. Сама программа — это просто "обертка", в которую "завернута" основная функция get_string, обеспечивающая проверку достоверности входных данных. Эта функция приглашает пользователя ввести строку, удаляет замыкающий символ новой строки и проверяет, не является ли введенная строка нулевой. После приема нулевой строки пользователю предлагается повторить ввод, но после трех неудачных попыток сценарий "умоет руки".

```
#!/usr/bin/perl

$maxatt = 3; # Максимальное число попыток ввести корректные данные

sub get_string {
  my ($prompt, $response) = shift;
  # Пытаемся прочитать входную строку до $maxatt раз
  for (my $attempts = 0; $attempts < $maxatt; $attempts++) {
      print "Пожалуйста, попытайтесь еще раз.\n" if $attempts;
      print "$prompt: ";
      $response = readline(*STDIN);
      chomp($response);
      return $response if $response;
  }
      die "Слишком много неудачных попыток ввести данные";
}
# Прием имен с помощью функции get_string и перевод в прописные буквы
$fname = uc get_string "Имя";
```

```
$lname = uc get_string "Фамилия";
printf "Полное имя: $fname $lname\n";
```

Результат выполнения этого сценария таков.

```
$ perl validate
Имя: John Ball
Фамилия: Park
Полное имя: JOHN BALL PARK
```

Функция `get_string` и цикл `for` иллюстрируют использование оператора `my` в создании переменных локальной области видимости. По умолчанию все переменные в Perl являются глобальными. Список локальных переменных для функции `get_string` инициализируется с помощью одного скаляра, получаемого из массива аргументов функции. Переменные в списке инициализации, которые не имеют надлежащего значения (в данном случае это переменная `$response`), остаются неопределенными.

Значение `*STDIN`, переданное функции чтения строк, имеет "всеядный" тип `typeglob`, который считается слабым местом в языковом проектировании. Из соображений самосохранения, лучше не ломайте голову над тем, что он (этот тип) в действительности означает. Возможно, вас удовлетворит такое короткое пояснение: в Perl дескрипторы файлов не относятся к типам данных "первого класса", поэтому в общем случае для передачи их функциям в качестве аргументов необходимо перед их именами ставить символ "звездочка".

В присваиваниях для переменных `$fname` и `$lname` обе функции `uc` (*c*onvert to *u*ppercase — означает *преобразовать в прописные буквы*) и `get_string` вызываются без использования круглых скобок. Поскольку в данном случае функциям передается один-единственный аргумент, здесь нет никакой неоднозначности, а потому "экономия" никак не навредила.

## Использование языка Perl в качестве фильтра

Язык Perl можно использовать без сценария — просто введя отдельные выражения в командную строку. Это позволяет быстро выполнить преобразования текста и "объявить" устаревшими такие программы фильтрации, как `sed`, `awk` и `tr`.

Ключ командной строки `-pe` дает возможность циклически обработать поток `STDIN`, вычислить простое выражение в каждой строке и вывести результат. Например, команда

```
ubuntu$ perl -pe 's#/bin/sh$#/bin/bash#' /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/bash
...
```

заменяет текст **/bin/sh** в конце строк файла **/etc/passwd** текстом **/bin/bash**, выводя трансформированный файл **passwd** в поток `STDOUT`. Возможно, вы привыкли к тому, что оператор подстановки текста использует в качестве разделителей слеши (например, **s/foo/bar/**), но Perl позволяет применять любой символ. Здесь как искомый текст, так и текст замены содержат слеши, поэтому в данном случае в качестве разделителя проще использовать символ "#". Предпочитая парные разделители, вы должны применять их в количестве четырех вместо "обычных" трех, например `s(foo)(bar)`.

Perl-ключ **-a** устанавливает режим автоматического разбиения, в котором входные строки разбиваются на поля, сохраняемые в массиве `@F`. Разделителем полей по умолчанию служит пробел, но с помощью ключа **-F** вы можете установить другой шаблон разделителя.

Режим авторазбиения удобно использовать вместе с ключом **-p** или его вариантом запрещения автовывода **-n**. Например, в приведенных ниже строках используется вариант команды **perl -ane**, чтобы отформатировать результат работы двух вариантов команды **df**. В третьей строке выполняется команда **join**, которая объединяет два набора полей в поле `Filesystem`, создавая составную таблицу, включающую поля, "вытащенные" из обеих версий результата команды **df**.

```
suse$ df -h | perl -ane 'print join("\t", @F[0..4]), "\n"' > tmp1
suse$ df -i | perl -ane 'print join("\t", @F[0,1,4]), "\n"' > tmp2
suse$  join tmp1 tmp2
Filesystem      Size    Used    Avail    Use%    Inodes    IUse%
/dev/hda3       3.0G    1.9G    931M     68%     393216    27%
udev            126M    172K    126M     1%      32086     2%
/dev/hda1       92M     26M     61M      30%     24096     1%
/dev/hda6       479M    8.1M    446M     2%      126976    1%
...
```

Вариант сценария без временных файлов выглядит следующим образом.

```
#!/usr/bin/perl

for (split(/\n/, 'df -h')) {
    @F = split;
    $h_part{$F[0]} = [ @F[0..4] ];
}
for (split(/\n/, 'df -i') {
    @F = split;
    print join("\t", @{$h_part{$F[0]}}, $F[1], $F[4]), "\n";
}
```

Истинно бесстрашные программисты могут использовать ключ **-i** вместе с ключом **-pe**, чтобы редактировать файлы "прямо по месту": Perl считывает содержимое файлов, передает их строки на редактирование и сохраняет результаты в исходных файлах. Вместе с ключом **-i** можно предоставить шаблон, который укажет, как именно резервировать оригинальную версию каждого файла. Например, вариант **-i.bak** позволит сохранить резервную копию файла **passwd** в виде файла **passwd.bak**. Но будьте осторожны — если вы не предоставите образец резервирования, резервная копия не будет создана совсем. Обратите внимание на то, что ключ **-i** и предоставляемый вами суффикс не разделяются пробелом.

## Модули расширения для Perl

Всеобъемлющая сеть архивов Perl (Comprehensive Perl Archive Network — CPAN) — это обширное хранилище пользовательских Perl-библиотек (`cpan.org`). Инсталляция новых модулей значительно облегчается командой **cpan**, которая действует во многом подобно открытому консольному менеджеру **yum** или менеджеру управления пакетами APT, "специализирующимся" на модулях Perl. Если вы работаете в системе Linux, поинтересуйтесь, есть ли в вашем дистрибутиве модуль, который вы собираетесь использовать в качестве стандартного средства, — намного проще инсталлировать однажды пакет системного уровня, а затем позволить системе самой периодически заботиться о собственных обновлениях.

Если в системе нет команды **cpan**, можно "пойти другим путем" и попытаться выполнить команду **perl -MCPAN -e shell**.

```
$ sudo perl -MCPAN -e shell

cpan shell -- CPAN exploration and modules installation (v1.9205)
ReadLine support available (maybe install Bundle::CPAN or Bundle::CPANxxl?)

cpan[1]> install Class::Date
CPAN: Storable loaded ok (v2.18)
CPAN: LWP::UserAgent loaded ok (v5.819)
CPAN: Time::HiRes loaded ok (v1.9711)
... several more pages of status updates ...
```

Можно посоветовать пользователям инсталлировать модули Perl в их домашние каталоги для личного применения, но этот процесс не совсем простой. В масштабе всей системы рекомендуем инсталлировать модули (с открытым кодом) сторонних производителей из архива CPAN. Эти модули Perl (их создателей можно идентифицировать по имени) не опаснее других программных продуктов с открытым кодом.

Многие модули Perl используют компоненты, написанные на языке C (что способствует повышению производительности). Инсталляция включает компиляцию этих сегментов, поэтому для работы вам необходимо иметь полную среду разработки, содержащую C-компилятор и полный набор библиотек.

Как и во многих других языках, самой распространенной ошибкой в Perl-программах считается повторная реализация средств, которые уже реализованы в модулях расширения.[12] При решении любой задачи на языке Perl возьмите себе за правило первым делом обращаться к архиву CPAN. Так вы сэкономите на времени разработки и отладки.

## 2.5. Создание сценариев на языке Python

По мере усложнения и увеличения в объеме программных проектов становятся очевидными преимущества объектно-ориентированного проектирования и реализации. Языку Perl явно не хватало объектно-ориентированного "буксира" в течение почти пяти лет, и хотя разработчики изо всех сил "налегали на весла", чтобы как-то удержаться на плаву, Perl-версия объектно-ориентированного программирования все еще оставляет желать лучшего.

Этот раздел посвящен языку Python (версии 2). Версия Python 3 пока находится в работе и, по всей вероятности, будет выпущена после выхода этой книги. Но, в отличие от Perl 6, она будет значительно усовершенствована.

Разработчикам с сильной подготовкой в области объектно-ориентированного программирования обычно нравятся Python и Ruby — языки написания сценариев с резко выраженным объектно-ориентированным уклоном. Создается такое впечатление, что язык Python в данный момент находится на нисходящей части кривой восприятия, поэтому его распространение — относительно простое направление деятельности для

[12] Вот что по этому поводу думает разработчик UNIX Том Кристиансен (Tom Christiansen): "В принципе, это правильно, но я бы выбрал другого кандидата. Претендентом на звание "самой распространенной ошибки в программах" я бы назвал тенденцию, состоящую в том, что программы, как правило, никогда не переписываются. Если вы когда-то писали сочинение, то знаете, насколько сильно могут отличаться его начальный и конечный варианты. Так и в программировании. Вероятно, вы слышали такое высказывание: 'Никогда не останавливайся на опытном образце'. Другими словами, очень печально, что люди, затратив силы на создание какой-то программы, почему-то никогда не переделывают ее, чтобы улучшить ее ясность и повысить эффективность".

менеджмента. Некоторые операционные системы, включая OpenSolaris, инвестируют значительные средства в развитие Python-возможностей по написанию сценариев. Язык Ruby, наоборот, по-прежнему ассоциируется, главным образом, с веб-разработкой и редко используется для сценариев общего назначения.

Язык Python был создан Гуидо ван Россумом (Guido van Rossum). По сравнению с Perl, здесь проще процесс кодирования, а сам код читабельнее. Python отличается простым для понимания синтаксисом даже для тех, кто сам не разрабатывал код. Если вам трудно запомнить детали использования операторов сравнения, вы оцените унифицированный подход, реализованнный в языке Python. Что касается системных администраторов, то они считают весьма полезными дополнительные типы данных, предлагаемые языком Python.

Если Python еще не инсталлирован в вашей системе, просмотрите список доступных пакетов, предлагаемых вашим поставщиком. Если с этим у вас возникнут проблемы, вы можете получить исходный код Python с сайта `python.org`. Здесь также вы найдете модули расширений, разработанные сторонними производителями.

Для того чтобы получить более подробное введение в язык Python, чем представленное здесь, начните с книги Марка Пилгрима (Mark Pilgrim) *Dive Into Python*. Ее можно прочитать и бесплатно загрузить на сайте `diveintopython.org` (или купить в печатном виде от издательства Apress). Полный список предлагаемой литературы по обсуждаемой теме приведен в разделе 2.7.

## Быстрое погружение в языковую среду Python

Как обычно, начнем с вывода сообщения "Привет, мир!". Как видите, его Python-версия практически идентична Perl-варианту.

```
#!/usr/bin/python
print "Привет, мир!"
```

Для выполнения этой программы установите "исполнительный" бит или вызовите напрямую интерпретатор языка Python.

```
$ chmod +x helloworld
$ ./helloworld
Привет, мир!
```

В этой программе не иллюстрируется самое "возмутительное" отступление Python от традиций, к которым мы так привыкли в программировании. В языке Python не используются ни скобки (фигурные и квадратные), ни специальные ключевые слова (`begin` и `end`), ограничивающие блоки. Операторы на одном и том же уровне отступа автоматически формируют блоки. Конкретный стиль отступов (пробелы или табуляция, глубина отступа) не имеет значения. Формирование блоков в Python прекрасно показано на следующем примере, в котором используется оператор `if-then-else`.

```
#!/usr/bin/python

import sys

a = sys.argv[1]

if a == "1":
    print 'a равно одному'
    print 'Это все еще ветвь then оператора if.'
else:
```

```
        print 'a равно', a
        print 'Это все еще ветвь else оператора if.'

    print 'А это уже за границей оператора if.'
```

В третьей строке программы импортируется модуль `sys`, который содержит массив `argv`. Обе ветви `then` и `else` оператора `if` имеют по две строки, и все они "отмечены" отступом одинакового уровня. Последний оператор `print` "выпадает" из контекста оператора `if`. Как и в языке Perl, Python-оператор `print` принимает произвольное количество аргументов. Но, в отличие от Perl, Python-оператор `print` вставляет пробел между каждой парой аргументов и автоматически "приправляет свое блюдо" символом новой строки. Вы можете подавить разделитель строк, включив дополнительную запятую в конец строки `print`; кроме того, `null`-аргумент предписывает оператору `print` не выводить символ новой строки.

Двоеточие в конце строки обычно служит признаком того, что за ней (этой строкой) следует блок, состоящий из строк с отступами.

```
$ python blockexample 1
a равно одному
Это все еще ветвь then оператора if.
А это уже за границей оператора if.

$ python blockexample 2
a равно 2
Это все еще ветвь else оператора if.
А это уже за границей оператора if.
```

Соглашение о Python-отступах ограничивает наши возможности в форматировании кода, но зато дает выигрыш в том, что программы, написанные разными программистами, выглядят практически одинаково. Это соглашение также означает, что для программистов отпадает необходимость "сдабривать" код знаками "точка с запятой" для завершения операторов.

Комментарии начинаются с символа # и продолжаются до конца строки (точно так же, как в оболочке **bash** и языке Perl).

Длинную строку можно разбить на части с помощью символа "\", помещенного в конце. В этом случае имеет значение только отступ первой строки. При желании можете использовать отступы и в остальных строках-продолжениях (частях разбитой длинной строки). Строки с несбалансированными скобками (круглыми, квадратными или фигурными) автоматически сигнализируют о продолжении строки даже при отсутствии символа "\", но вы можете вставить этот символ, чтобы прояснить структуру вашего кода.

При выполнении некоторых операций вырезки и вставки символы табуляции преобразуются в пробелы, что может иметь для вас неожиданные последствия. Поэтому следуйте золотому правилу: никогда не смешивайте символы табуляции и пробелы; для создания отступов используйте либо табуляцию, либо пробелы. Работа многих программных продуктов основана на традиционном предположении, что символ табуляции соответствует интервалу, состоящему из восьми пробелов, что на практике "выливается" в слишком большие отступы для получения читабельного кода. Большая часть Python-сообщества предпочитает использовать пробелы и 4-символьные отступы.

Если вы захотите кардинально решить проблему отступов, то, "идя навстречу вашим пожеланиям", многие текстовые редакторы включают средства, которые помогают сохранить в здравии вашу психику, либо запрещая символы табуляции (в пользу пробелов), либо отображая эти "конкурирующие" символы по-разному. В качестве послед-

него средства можете переводить символы табуляции в пробелы с помощью команды **expand** или использовать команду **perl -pe** для замены табуляции строкой символов, которая лучше видна.

## Объекты, строки, числа, списки, словари, кортежи и файлы

Все типы данных в языке Python являются объектами, и это придает им больше силы и гибкости, чем в языке Perl.

В языке Python списки заключаются в квадратные скобки (а не в круглые). Индексация массивов начинается с нуля (это один из немногочисленных принципов, который не меняется для всех трех "сценарных" языков, рассматриваемых в этой главе).

"Новым словом" в языке Python прозвучали "кортежи," которые, по сути, представляют собой неизменяемые списки. Кортежи работают быстрее, чем массивы, и оказываются весьма полезными для представления данных, которые должны быть немодифицированными. Синтаксис для кортежей такой же, как для списков, за исключением того, что разделителями у них служат круглые скобки (а не квадратные). Поскольку запись `(thing)` похожа на простое алгебраическое выражение, кортежи, содержащие только один элемент, должны иметь дополнительную запятую, которая не оставит у вас сомнений, с чем именно вы имеете дело, — `(thing, )`.

Рассмотрите пример использования некоторых основных типов данных в языке Python.

```
#!/usr/bin/python

name = 'Gwen'
rating = 10
characters = [ 'ГубкаБоб', 'Патрик', 'Сквидвард' ]
elements = ( 'литий', 'углерод', 'бор' )

print "Имя:\t%s\nРейтинг:\t%d" % (name, rating)
print "Герои:\t%s" % characters
print "Элементы:\t%s" % (elements, )
```

Вот как выглядит результат выполнения этого примера.

```
$ python objects
Имя:        Gwen
Рейтинг:    10
Герои:      ['ГубкаБоб', 'Патрик', 'Сквидвард']
Элементы:   ('литий', 'углерод', 'бор')
```

Переменные в языке Python не отмечаются синтаксически и не объявляются с указанием типа, но объекты, на которые они ссылаются, имеют базовый тип. В большинстве случаев язык Python автоматически не преобразует типы за вас, но это могут сделать отдельные функции или операторы. Например, вы не можете сцепить строку и число (с помощью оператора "+") без явного преобразования числа в его строковое представление. Однако операторы форматирования приводят объекты любого типа к строковому виду. Каждый объект имеет строковое представление.

Действие оператора строкового форматирования "%" во многом подобно действию функции **sprintf** из языков C или Perl, но его можно использовать везде, где может находиться строка. Это бинарный оператор, в левой части которого стоит строка, а в

правой — вставляемые значения. Если вставляемых значений несколько, они должны быть представлены в виде кортежа.

Словарь в языке Python — это то же самое, что хеш в языке Perl, т.е. список пар "ключ/значение". Словарные литералы заключаются в фигурные скобки, а элементы, составляющие пару "ключ/значение", отделяются друг от друга двоеточием.

```
#!/usr/bin/python

ordinal = { 1 : 'первый', 2 : 'второй', 3 : 'третий' }
print "Массив ordinal содержит", ordinal
print "Значением ordinal для ключа 1 является", ordinal[1]
```

На практике Python-словари во многом подобны массивам, за исключением того, что "словарные" индексы (ключи) могут быть объектами, отличными от целых чисел.

```
$ python dictionary
Массив ordinal содержит {1: 'первый', 2: 'второй', 3: 'третий'}
Значением ordinal для ключа 1 является первый
```

В языке Python файлы открываются как объекты с помощью соответствующих методов. Метод `readline`, вполне оправдывая свое имя, считывает одну строку. При выполнении приведенного ниже примера считываются (и выводятся) две строки из файла **/etc/passwd**.

```
#!/usr/bin/python

f = open('/etc/passwd', 'r')
print f.readline(),
print f.readline(),
f.close()
$ python fileio
at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/true
bin:x:1:1:bin:/bin:/bin/true
```

Замыкающие запятые в операторах `print` используются для подавления символов новой строки, поскольку каждая строка и так уже включает этот символ, считанный из исходного файла.

## Пример контроля входных данных

Приведенный ниже сценарий — это Python-версия уже знакомой нам программы проверки допустимости входных данных. В этом сценарии демонстрируется использование подпрограмм и аргументов командной строки, а также некоторых других "питонизмов" (от англ. *Python* — питон).

```
#!/usr/bin/python

import sys
import os

def show_usage(message, code = 1):
   print message
   print "%s: source_dir dest_dir" % sys.argv[0]
   sys.exit(code)
if len(sys.argv) != 3:
   show_usage("Нужно передать 2 аргумента; вы передали %d" % (len(sys.argv) - 1))
elif not os.path.isdir(sys.argv[1]):
```

```
    show_usage("Недопустимый каталог-источник")
elif not os.path.isdir(sys.argv[2]):
    show_usage("Недопустимый каталог-приемник")

source, dest = sys.argv[1:3]

print "Каталогом-источником является", source
print "Каталогом-приемником является", dest
```

В дополнение к импортированию модуля `sys`, мы также импортируем модуль `os` (чтобы получить доступ к утилите `os.path.isdir`). Обратите внимание на то, что команда `import` не позволяет вам "фамильярничать" с содержимым импортируемых модулей при доступе к любым определенным в них символам, т.е. вы *не* освобождаетесь от необходимости использовать полные имена, которые начинаются с имени модуля.

Определение подпрограммы `show_usage` предполагает использование значения по умолчанию для кода завершения на случай, если вызывающий оператор не передаст этот аргумент в явном виде. Поскольку все типы данных являются объектами, аргументы функции передаются по ссылке.

Первый элемент массива `sys.argv` (в позиции 0) содержит имя сценария, поэтому длина массива на единицу больше, чем количество реально переданных аргументов командной строки. Запись `sys.argv[1:3]` означает только часть массива. Удивительно, но часть (подмассив) не включает элемент, расположенный на дальнем конце заданного диапазона, поэтому рассматриваемая часть массива содержит только элементы `sys.argv[1]` и `sys.argv[2]`. Чтобы включить в подмассив второй и последующие аргументы командной строки, можно использовать выражение `say sys.argv[1:]`.

Подобно оболочке `bash` и языку Perl, в языке Python предусмотрено специальное условие "else if ", которому соответствует ключевое слово `elif`. В языке Python нет оператора-переключателя (`switch`) для передачи управления на различные ветви (`case`) в зависимости от результата сравнения аргумента с заданными значениями.

Параллельное присваивание значений переменным `source` и `dest` несколько отличается от Perl-версии тем, что самих этих переменных нет в списке. Язык Python позволяет параллельные присваивания в любой форме.

В языке Python используются одинаковые операторы сравнения для числовых и строковых значений. Оператор сравнения "`!=`" означает "не равно". Обратите внимание на то, что в языке Python унарного оператора "`!`" нет; используйте вместо него оператор `not`. Булевы операторы `and` и `or` выполняют булевы операции (что вполне ожидаемо), но они не возвращают булевы значения: результатом всегда является значение одного из операндов.

## Циклы

В приведенном ниже фрагменте используется конструкция `for...in`, обеспечивающая выполнение итераций в диапазоне от 1 до 10.

```
for counter in range(1, 10):
    print counter,
```

Как и в подмассиве из предыдущего примера, правый конец указанного диапазона на самом деле не включается, поэтому результат содержит только числа от 1 до 9.

```
1 2 3 4 5 6 7 8 9
```

Цикл `for` в этой форме существует только в языке Python, причем он обладает такими сильными свойствами, которые существенно отличают его от циклов `for` в других языках.

- В синтаксисе цикла нет никаких специальных требований в отношении числовых диапазонов. Итерационную Python-модель может поддерживать практически любой объект. Так, вы можете организовать выполнение итераций по строке (посимвольно), списку, файлу (по символам, строкам или блокам), подмассиву и т.д.
- Итераторы могут возвращать несколько значений, и вы можете использовать несколько переменных цикла. Присваивание в начале каждой итерации работает подобно обычным множественным Python-присваиваниям.
- Оба цикла `for` и `while` могут иметь в конце оператор `else`, который выполняется только в том случае, если цикл завершается нормально (в противоположность выходу из цикла посредством оператора `break`). Это свойство на первый взгляд может показаться нелогичным, но оно позволяет реализовать довольно элегантные решения для выполнения определенных действий при заданных условиях.

В приведенном ниже примере сценарий принимает в командной строке регулярное выражение и сопоставляет его со списком имен карликов (из сказки "Белоснежка и семь гномов") и цветов их костюмов. Первое совпадение выводится в виде сообщения о том, какого цвета костюм у такого-то карлика, причем те части текста, которые совпали с регулярным выражением, содержат знаки подчеркивания.

```
#!/usr/bin/python

import sys
import re

suits = { 'Bashful':'red', 'Sneezy':'green', 'Doc':'blue', 'Dopey':'orange',
    'Grumpy':'yellow', 'Happy':'taupe', 'Sleepy':'puce' }
pattern = re.compile("(%s)" % sys.argv[1])

for dwarf, color in suits.items():
   if pattern.search(dwarf) or pattern.search(color):
       print "%s's dwarf suit is %s." % \
          (pattern.sub(r"_\1_", dwarf), pattern.sub(r"_\1_", color))
       break
else:
           print "No dwarves or dwarf suits matched the pattern."
```

Вот как выглядят результаты выполнения этого сценария.

```
$ python dwarfsearch '[aeiou]{2}'
Sn_ee_zy's dwarf suit is gr_ee_n.

$ python dwarfsearch go
No dwarves or dwarf suits matched the pattern.
```

Присваивание переменной `suits` демонстрирует Python-синтаксис для кодирования литеральных словарей. Метод `suits.items()` работает как итератор для пар "ключ/значение" — обратите внимание на то, что на каждой итерации мы извлекаем как имя карлика, так и цвет костюма. Если хотите, чтобы итерация происходила только по ключам, можете использовать такой вариант организации цикла: `for dwarf in suits`.

Python реализует обработку регулярного выражения посредством его модуля `re`. Никакие правила работы с регулярными выражениями не встроены в сам язык, поэтому их обработка в Python более громоздкая, чем в Perl. Здесь сначала выбирается регулярное выражение `pattern` из первого аргумента командной строки, заключенного в круглые скобки, чтобы сформировать "группу захвата".

Затем проверяются и модифицируются строки с помощью методов `search` и `sub` объекта регулярного выражения. Можно также использовать вызов соответствующих функций (например, `re.search`), передав регулярное выражение в качестве первого аргумента. Выражение `\1` в строке подстановки представляет собой обратную ссылку на содержимое первой "группы захвата".

## 2.6. Передовой опыт создания сценариев

Несмотря на то что фрагменты программ в этой главе снабжены некоторыми комментариями и сообщениями о правильном применении сценариев (поскольку мы хотели обратить ваше внимание на кое-какие нюансы), хотелось бы, чтобы реальные сценарии были еще лучше. Существует множество книг о написании качественных программ, но все же будет не лишним изложить здесь некоторые основные рекомендации по написанию сценариев.

- При запуске с неадекватными аргументами сценарий должен вывести сообщение о его корректном применении и завершиться. Можно также реализовать возможность получения справочной информации (**`--help`**).
- Проверяйте входные данные и выходные значения на корректность. Прежде чем выполнять команду **`rm -rf`** (удаление указанных имен файлов из каталога), хорошо было бы, чтобы сценарий сначала удостоверился, что полученный в результате путь будет соответствовать ожидаемому образцу. Такие двойные предосторожности (и средства языка, которые позволяют их реализовать) иногда оказываются весьма полезными.
- Сценарий должен возвращать надлежащий код завершения: нуль — в случае успешного выполнения и ненулевое значение — при неудачном исходе. Необязательно сопровождать каждый вид отказа уникальным кодом завершения, однако следует поинтересоваться, какая информация была бы полезной для вызывающих процедур.
- Используйте соответствующие соглашения о присвоении имен для переменных, сценариев и подпрограмм. Приведите в соответствие соглашения, действующие в языке, в остальной части кодовой основы вашего узла и, что самое важное, других переменных и функциях, определенных в текущем проекте. Используйте принцип смешанного регистра букв или знаки подчеркивания, чтобы сделать читабельными длинные имена.[13]
- Для переменных используйте имена, которые отражают хранимые в них значения, но старайтесь все же давать имена покороче. Например, переменную для хранения количества входных строк можно, конечно, назвать `number_of_lines_of_input`, но уж очень оно длинное, и его лучше заменить так: `n_lines`.

[13] Имена самих сценариев также имеют большое значение. В этом контексте "в роли" пробелов дефисы предпочтительнее, чем знаки подчеркивания (например, **system-config-printer**).

- Разработайте руководство по стилю, чтобы вы и ваши коллеги могли писать код, используя одни и те же соглашения. Такое руководство облегчит вам понимание кода, написанного другими программистами, а им — написанного вами.
- Начинайте каждый сценарий с блока комментария, в котором будет сказано, что делает данный сценарий и какие параметры он принимает. Не забудьте указать свое имя и текущую дату. Если сценарий требует, чтобы в системе были инсталлированы нестандартные инструменты, библиотеки или модули, обязательно перечислите их.
- Комментируйте свой код, используя такую степень детализации, какую посчитаете полезной, с учетом временного "отхода" от этого кода и с последующим возвратом к нему через месяц или два. Имеет смысл также комментировать выбор алгоритма, причины неиспользования более очевидного способа кодирования, необычные пути в коде, все "трудные места", возникшие в период разработки. Не загромождайте код бесполезными комментариями; отнеситесь к написанию комментариев с точки зрения потенциального читателя: пишите лаконично, грамотно и "по делу".
- Комментарии кода лучше всего применять к целым блокам или функциям. Комментарии, которые описывают назначение переменной, следует вставлять вместе с объявлением этой переменной или при ее первом использовании.
- Лучше запускать сценарии от имени суперпользователя `root`. Не устанавливайте для них атрибут **`setuid`**, поскольку такие сценарии довольно трудно сделать совершенно безопасными. Для того чтобы реализовать надлежащие стратегии управления контролем доступа, используйте команду **`sudo`**.
- В оболочке **`bash`** используйте ключ **`-x`**, чтобы отобразить команды, прежде чем они будут выполнены, и ключ **`-n`**, чтобы проверить синтаксис команд без их выполнения.
- Используйте Perl-ключ **`-w`**, который предупредит вас о подозрительном поведении, например об использовании переменных до установки их значений. Вы можете использовать этот ключ в "описательной" строке или "внедрить" его в текст программы с помощью директивы `use warnings`.
- Выполняя Python-сценарий, вы находитесь в режиме отладки, если явно не отключите его с помощью аргумента `-O` в командной строке. Это означает, что еще до вывода диагностических выходных данных вы можете протестировать специальную переменную `__debug__`.

Том Кристиансен (Tom Christiansen) предлагает следующие пять золотых правил для генерирования полезных сообщений об ошибках.

- Сообщения об ошибках должны направляться в канал `STDERR`, а не `STDOUT`.
- Включайте в сообщение об ошибке имя программы.
- Указывайте виновника (имя функции или операции) ошибочного действия.
- Если к сбою приводит системный вызов, включите строку `perror` (`$!` в языке Perl).
- Если код завершения некоторой операции не равен нулю, обеспечьте выход из программы.

В среде Perl несложно следовать всем этим пяти правилам.

```
die "can't open $filename: $!";
```

## 2.7. Рекомендуемая литература


- Brooks, Frederick P. JR. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1995.


### Основы работы в командной оболочке и написание сценариев в среде bash


- Albing, Carl, JP Vossen and Cameron Newham. *Bash Cookbook*. Sebastopol, CA: O'Reilly Media, 2007.
- Kernighan, Brian W. and Rob Pike. *The UNIX Programming Environment*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- Newham, Cameron and Bill Rosenblatt. *Learning the bash Shell (3rd Edition)*, Sebastopol, CA: O'Reilly Media, 2005.
- Powers, Shelley, Jerry Peek, Tim O'reilly and Mike Loukides. *Unix Power Tools, (3rd Edition)*, Sebastopol, CA: O'Reilly Media, 2002.


### Регулярные выражения


- Friedl, Jeffrey. *Mastering Regular Expressions (3rd Edition)*, Sebastopol, CA: O'Reilly Media, 2006.
- Goyvaerts, Jan and Steven Levithan. *Regular Expressions Cookbook*. Sebastopol, CA: O'Reilly Media, 2009.


### Написание сценариев на языке Perl


- Wall, Larry, Tom Christiansen and Jon Orwant. *Programming Perl (3rd Edition)*, Sebastopol, CA: O'Reilly Media, 2000.
- Schwartz, Randal L., Tom Phoenix and Brian D Foy. *Learning Perl (5th Edition)*, Sebastopol, CA: O'Reilly Media, 2008.
- Blank-Edelman, David. *Automating System Administration with Perl*, Sebastopol, CA: O'Reilly Media, 2009.
- Christiansen Tom and Nathan Torkington. *Perl Cookbook (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.


### Написание сценариев на языке Python


- Beazley, David M. *Python Essential Reference (4th Edition)*, Reading, MA: Addison-Wesley, 2009.
- Gift, Noah and Jeremy M. Jones. *Python for Unix and Linux System Administrators*, Sebastopol, CA: O'Reilly Media, 2008.
- Martelli, Alex, Anna Martelli Ravenscroft and David Ascher. *Python Cookbook (2nd Edition)*, Sebastopol, CA: O'Reilly Media, 2005.
- Pilgrim, Mark. *Dive Into Python*. Berkeley, CA: Apress, 2004.

  К этой книге можно также получить бесплатный доступ на веб-сайте `diveintopython.org`.

## 2.8. Упражнения

2.1 UNIX позволяет в именах файлов использовать пробелы. Как найти файлы, имена которых содержат пробелы? Как удалить такие файлы? Можно ли рассчитывать на встроенные в оболочку **bash** и языки Perl и Python средства обработки пробелов в именах файлов или нужно самим предпринимать специальные меры предосторожности? Перечислите основные правила написания сценариев.

2.2. Напишите простой **bash**-сценарий (или два сценария) для резервирования и восстановления вашей системы.

2.3 Используя регулярные выражения, напишите Perl- или Python-сценарий для анализа даты в форме, генерируемой командой **date** (например, `Tue Oct 20 18:09:33 PDT 2009`), и определения ее корректности (например, чтобы не "прошло" несуществующее 30 февраля). Существует ли готовая библиотека или готовый модуль, который позволит вам сделать это одной строкой кода? Если да, то поясните, как инсталлировать их, и перепишите свой сценарий с их использованием.

2.4. Напишите сценарий, который перечисляет пользователей системы и группы, используя файлы **/etc/passwd** и **/etc/group** (или их эквиваленты в сетевых базах данных). Для каждого пользователя выведите его идентификатор (UID) и группы, членом которых он является.

2.5. Усовершенствуйте подпрограмму `get_string` (см. подраздел "Прием входных данных и проверка их достоверности" раздела 2.4 этой главы) так, чтобы она принимала только целые числа. Новая версия подпрограммы должна принимать три параметра: строку-приглашение, нижний предел и верхний предел допустимых целых чисел.

2.6. Найдите недокументированный сценарий, который используется в вашей среде. Прочитайте его и убедитесь, что понимаете его. Добавьте комментарии и напишите man-страницу для этого сценария.

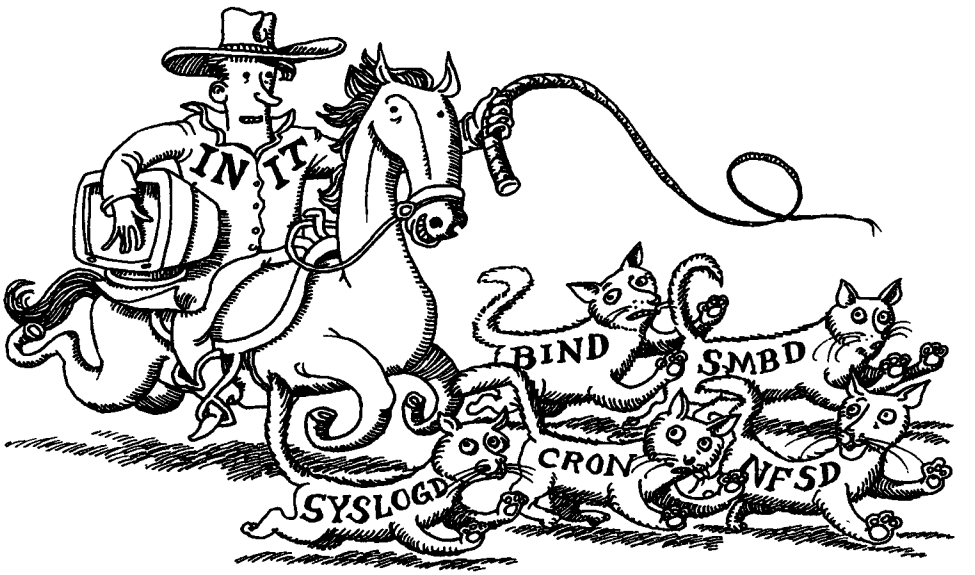
2.7. ✯ Напишите сценарий, который отображает короткую (длиной в одну страницу) суммарную сводку данных о состоянии оборудования по одной из следующих категорий: ЦП, память, диск или сеть. Ваш сценарий с помощью системных команд и файлов должен создать легко воспринимаемую инструментальную панель, которая содержала бы максимально возможный объем информации.

2.8. ✯ Создайте управляемый в режиме меню интерфейс, который упрощает выбор ключей командной строки для команд **top**, `sar` или инструмента анализа производительности.

2.9. ✯ Напишите сценарий для тестирования возможности сетевого соединения на сервере и служб обратных потоков, от которых она зависит (например, служба доменных имен — DNS, файловая служба, протокол LDAP или другая каталоговая служба). В случае обнаружения каких-либо проблем обеспечьте отправку вам сообщения по электронной почте или текстового сообщения.

# Глава 3

# Запуск и останов системы




Как и практически все операции в UNIX, процедуры включения/выключения системы превратились в тщательно спроектированные процессы, которые учитывают множество возможных непредвиденных ситуаций. Как администраторы мы рассматриваем сложности процесса загрузки во многих аспектах, чтобы предотвратить потенциальные проблемы и устранить возникшие. Эффективный системный администратор, прежде всего, "зрит в корень".

Процесс начальной загрузки системы всегда был достаточно сложным, но все же он был несколько проще в те дни, когда изготовители определяли буквально все аспекты аппаратного и программного обеспечения. Теперь, когда Linux и Solaris управляют аппаратным обеспечением персональных компьютеров, процедура загрузки должна выполняться по их правилам. При этом приходится иметь дело с множеством возможных конфигураций. Несмотря на то что в этой главе мы обсуждаем процедуру загрузки для всех рассматриваемых нами операционных систем, все же больше внимания уделяется версиям UNIX, ориентированным на персональные компьютеры, чем "стационарным" системам.

Хотя эта глава в книге — одна из первых, в ней мы иногда оперируем понятиями, которые подробно рассматриваются лишь через несколько сотен страниц. Поэтому рекомендуем также прочесть главы 6 и 13.

## 3.1. Начальная загрузка

Под начальной загрузкой (bootstrapping), понимается запуск системы при включении питания. Поскольку обычные средства операционной системы на данном этапе еще недоступны, система должна "самозагрузиться", в буквальном смысле "обслужить себя

самостоятельно". В ходе этого процесса ядро системы загружается в память и активизируется. Затем выполняется ряд инициализационных задач, после чего система готова к обслуживанию пользователей.

Начальная загрузка — это период особой уязвимости системы. Ошибки в конфигурации, сбои в работе или отсутствие нужного оборудования, повреждения файловых систем могут помешать компьютеру нормально начать работу. Настройка режимов загрузки часто является одной из первых задач, которую приходится решать администратору в новой системе, особенно при добавлении нового оборудования. К несчастью, эта задача — одна из наиболее сложных, и для ее решения необходимо хорошо знать многие другие аспекты системы.

Когда включается питание, запускается на выполнение загрузочный код, хранящийся на постоянном запоминающем устройстве. Он должен запустить ядро. Ядро опрашивает состояние аппаратных устройств, а затем запускает демон `init`, идентификатор которого всегда равен 1.

Прежде чем система полностью загрузится, должны быть проверены и смонтированы файловые системы и запущены системные демоны. Соответствующие процедуры реализуются с помощью сценариев интерпретатора команд, которые последовательно запускаются демоном `init`. Конкретная структура сценариев и способ их выполнения зависят от системы. Все эти вопросы будут рассмотрены в данной главе.

## Главное — активизировать командную оболочку

При нормальной работе системы сами выполняют начальную загрузку в автономном режиме, после чего к ним могут получить удаленный доступ администраторы и пользователи. Однако администраторам необходимо иметь инструмент восстановления системы в случае, если неожиданный отказ дисковода или какая-нибудь проблема конфигурации помешает системе нормально выполнить процесс начальной загрузки. Системы UNIX (вместо загрузки "по полной программе") могут ограничиться только самым необходимым, а именно активизацией командной оболочки на системной консоли. Этот вариант называют входом системы в так называемый "однопользовательский режим", режим восстановления или профилактический режим — все эти термины в этой главе взаимозаменяемы. Однопользовательский режим не позволяет выполнять сетевые операции, а для использования системной консоли вам нужно иметь физический доступ к ней.

В большинстве систем для входа в однопользовательский режим во время начальной загрузки необходимо передать ядру в командной строке определенный параметр. Если система уже загружена и работает, вы можете перевести ее в однопользовательский режим с помощью команды **`shutdown`** или **`telinit`**.

## Этапы загрузки

Типичная процедура начальной загрузки состоит из шести отдельных этапов:

- считывание начального загрузчика с главной загрузочной записи;
- загрузка и инициализация ядра;
- обнаружение и конфигурирование устройств;
- создание процессов ядра;
- вмешательство администратора (только в однопользовательском режиме);
- выполнение системных сценариев запуска.

Почти все этапы не требуют интерактивного контроля со стороны администратора. Ведь администраторы имеют возможность управлять загрузкой системы, редактируя файлы конфигурации для сценариев запуска системы или изменяя аргументы, передаваемые загрузчиком ядру.

## Инициализация ядра

🕮 Подробнее о ядре рассказывается в главе 13.

Ядро представляет собой программу, и в качестве первой задачи начальной загрузки выполняется запись этой программы в память для последующего выполнения. Имя файлу ядра дает его изготовитель, но по традиции оно называется **/unix** или **/vmunix**. В Linux-системах ядро обычно получает путевое имя в виде вариации на тему **/boot/vmlinuz**.

В большинстве систем загрузка осуществляется в два этапа. Сначала в память компьютера с диска считывается (с помощью кода, записанного на постоянном запоминающем устройстве) небольшая программа начальной загрузки (именуемая *начальным загрузчиком*), которая затем выполняет собственно загрузку ядра. Эта процедура совершается вне UNIX-домена и поэтому не стандартизирована среди систем.

Ядро выполняет тесты, позволяющие определить, сколько памяти имеется в распоряжении системы. Часть внутренних структур ядра имеет фиксированный размер, поэтому ядро резервирует определенный объем физической памяти для самого себя. Эта память недоступна пользовательским процессам. Ядро выдает на консоль сообщение об общем объеме физической памяти и объеме памяти, доступной пользовательским процессам.

## Конфигурирование аппаратных средств

Одна из первых задач ядра — исследование компонентов аппаратного обеспечения. В процессе тестирования различных системных шин и инвентаризации оборудования ядро выводит на консоль краткую информацию о каждом обнаруженном устройстве. Во многих случаях ядро загружает драйверы устройств как независимые модули ядра. Для систем, ориентированных на персональные компьютеры, дистрибутивы включают ядро, которое способно работать в большинстве аппаратных конфигураций компьютеров, требуя при этом минимальной настройки или же вообще обходясь без нее.

Процесс конфигурации аппаратного обеспечения должен быть относительно прозрачным для администраторов, особенно в среде Linux. Готовые ядра имеют модульную структуру и автоматически обнаруживают большую часть устройств. Тем не менее вы можете встретить нераспознанные устройства. О том, как справиться с конфигурацией драйверов вручную, прочитайте в главе 13.

## Создание процессов ядра

После завершения этапа базовой инициализации ядро создает в области памяти, выделенной для пользовательских программ, несколько "самовыполняющихся" процессов. Это происходит "в обход" стандартного системного механизма **fork** (описан в разделе 5.2).

Количество таких процессов зависит от операционной системы, хотя демон **init** всегда имеет идентификатор процесса (PID), равный 1. Большинство систем UNIX использует **sched** в качестве процесса с идентификатором 0.

В Linux процесс с идентификатором PID, равным 0, отсутствует. Демон **init** работает в сопровождении с различными обработчиками памяти и сигналов ядра, в том числе с теми, которые представлены в табл. 3.1. Все эти процессы имеют идентификаторы с низкими номерами, а их имена в листингах команды **ps** заключены в квадратные скобки (например, `[kacpid]`). Иногда имена процессов могут содержать в конце символ косой черты и цифру, как, например, `[kblockd/0]`. Число указывает процессор, на котором выполняется данный процесс. Эта информация может быть полезна при настройке многопроцессорной системы.

**Таблица 3.1. Некоторые наиболее часто встречающиеся процессы ядра в Linux-системах**

| Процесс | Назначение |
|---|---|
| **kjournald** | Записывает обновления журнала на диск [a] |
| **kswapd** | Выполняет подкачку процессов при недостаточном объеме физической памяти |
| **ksoftirqd** | Обрабатывает мягкие прерывания, если ими нельзя заняться во время переключения контекста |
| **khubd** | Выполняет конфигурирование USB-устройств |

[a] Для каждой смонтированной файловой системы ext3 или ext4 существует один процесс **kjournald**.

Из этих процессов только **init** является полноценным пользовательским процессом; остальные фактически представляют собой части ядра, которые были сделаны процессами из концептуальных соображений.

Системы UNIX создают подобные процессы ядра, но поскольку эти процессы отображают особенности конкретной реализации ядра, никакие имена или функции не могут быть одинаковыми для разных систем. К счастью, администраторам никогда не приходится взаимодействовать с этими процессами напрямую.

После создания этих процессов ядро больше не принимает участия в процедуре начальной загрузки системы. К этому моменту, однако, еще не создан ни один из процессов, управляющих базовыми операциями (например, регистрацией пользователей в системе), и большинство демонов не запущено. Обо всем этом позаботится (в некоторых случаях косвенно) демон **init**.

## Действия оператора (только в режиме восстановления)

🕮 Подробнее об учетной записи **root** рассказывается в главе 4.

Если систему нужно запустить в режиме восстановления, оператор устанавливает в командной строке специальный флаг, а ядро передает эту информацию демону **init** в качестве уведомления о запуске. Во время однопользовательской загрузки вы должны получить приглашение ввести пароль пользователя **root**. Если он введен правильно, запускается интерпретатор команд с правами суперпользователя. Можно не задавать пароль, а просто нажать комбинацию клавиш <Ctrl+D>, после чего загрузка продолжится в многопользовательском режиме (подробнее см. раздел 3.4).

🕮 Подробнее о файловых системах и их монтировании речь пойдет в главе 6.

В однопользовательском режиме команды выполняются почти так же, как и в полностью загруженной системе. Однако иногда монтируется только корневой раздел. Для того чтобы можно было использовать программы, находящиеся вне каталогов **/bin**, **/sbin** или **/etc**, необходимо вручную смонтировать остальные файловые системы.

Во многих однопользовательских средах корневой каталог файловой системы монтируется в режиме только для чтения. Если каталог **/etc** является частью корневой файловой системы (обычная ситуация), редактирование многих файлов конфигурации будет невозможно.[1] Чтобы выйти из положения, придется начать однопользовательский сеанс с повторного монтирования каталога / в режиме чтения/записи. Нужное действие в Linux-системах выполняет следующая команда.

```
# mount -o rw,remount /
```

В большинстве других систем вы можете выполнить команду **mount /**, чтобы реализовать обращение к файлу **fstab** или **vfstab** и выяснить, как должна быть смонтирована файловая система.

В системе Red Hat загрузка в однопользовательском режиме выполняется несколько активнее, нежели обычно. До того момента, когда на экран будет выведено приглашение интерпретатора команд, этот дистрибутив попытается смонтировать все локальные файловые системы. Хотя на первый взгляд такой подход кажется удобным, но при использовании недостаточно продуманной файловой системы он может порождать проблемы.

Команда **fsck**, которая проверяет и восстанавливает поврежденные файловые системы, обычно выполняется в ходе автоматической загрузки. Если система запускается в однопользовательском режиме, команду **fsck** придется вводить вручную. Подробнее эта команда описана в разделе 8.9.

Когда интерпретатор команд однопользовательского режима завершит свою работу, система продолжит загрузку в нормальном режиме.

## Выполнение сценариев запуска системы

В тот момент, когда система сможет выполнять сценарии запуска, ее уже можно назвать UNIX. Это еще не полностью загруженная система, но "загадочных" этапов процесса загрузки больше не осталось. Файлы сценариев представляют собой обычные командные файлы, которые выбираются и запускаются демоном `init` по сложному, но, в общем-то, понятному алгоритму.

Точное местонахождение, содержимое и организация сценариев запуска заслуживают отдельного изучения. Подробнее об этом рассказывается в разделе 3.6. Краткий курс написания сценариев можно пройти, изучив материал главы 2.

## Завершение процесса загрузки

🕮 Подробнее процесс регистрации в системе описан в разделе 31.8.

После выполнения сценариев инициализации система полностью готова к работе. Такие системные демоны, как DNS- и SMTP-серверы, принимают и обслуживают подключения. Не забывайте о том, что демон **init** продолжает играть важную роль даже по завершении начальной загрузки.

У демона **init** есть один однопользовательский и несколько многопользовательских "уровней выполнения", определяющих, какие ресурсы системы будут доступны пользователю. Уровни выполнения рассматриваются в разделе 3.5.

[1] Например, однопользовательский режим используется для восстановления утраченного пароля суперпользователя. Эта операция требует модификации файла **/etc/shadow**.

## 3.2. Загрузка системы на персональном компьютере

До этого момента описывалась общая схема начальной загрузки. Теперь некоторые наиболее важные (и сложные) ее этапы необходимо рассмотреть подробнее, проанализировав особенности функционирования Intel-систем.

Загрузка системы на персональном компьютере — это многоступенчатый процесс. Когда включается компьютер, начинает выполняться код, записанный на постоянном запоминающем устройстве. Точное его местонахождение и структура зависят от типа оборудования. В компьютерах, созданных специально для UNIX или другой коммерческой операционной системы, код "прошивается" разработчиком, который заранее задает алгоритм подключения устройств, базовой инициализации сети и распознавания локальных файловых систем. Это очень удобно для системного администратора. Ему достаточно ввести лишь имя нового файла ядра, а код постоянного запоминающего устройства автоматически обнаружит и прочитает этот файл.

На персональных компьютерах код начальной загрузки представлен в виде базовой подсистемы ввода-вывода — BIOS (Basic Input/Output System), которая чрезвычайно упрощена в сравнении с фирменным кодом UNIX-станций. В действительности в системе BIOS есть несколько уровней кода: для самого компьютера, для видеоплаты, для SCSI-адаптера, если таковой имеется, и иногда для других периферийных устройств вроде сетевых плат.

Встроенному коду BIOS известно об устройствах, расположенных на материнской плате, в частности о IDE- и SATA-контроллерах (и жестких дисках), плате сетевого адаптера, измерителе мощности и температуры, контроллере клавиатуры, последовательных и параллельных портах. SCSI-адаптеры знают только об устройствах, подключенных непосредственно к ним. К счастью, за последние несколько лет сложные взаимодействия, необходимые для обеспечения совместной работы этих устройств, были стандартизованы и теперь почти не требуют вмешательства оператора.

В режиме конфигурирования можно выбрать, с какого устройства требуется загружаться. Обычно последовательность загрузки задается в виде правила, например: "сначала — DVD, затем — "флешка" (USB drive), в последнюю очередь — жесткий диск". Распространенным вариантом также считается сетевая загрузка с помощью среды PXE (см. раздел 12.1).

Когда компьютер определил, с какого устройства следует загружаться, считывается его (устройства) первый блок. Этот сегмент (512 байт) называется *главной загрузочной записью* (master boot record — MBR). В ней хранится программа, которая сообщает компьютеру о том, в каком разделе диска расположена программа вторичной загрузки (загрузчик операционной системы). Дополнительная информация о разделах дисков на персональных компьютерах и главной загрузочной записи приведена в главе 8.

Стандартная программа, находящаяся в главной загрузочной записи, дает компьютеру указание извлечь загрузчик операционной системы из первого раздела диска. В некоторых системах поддерживаются более сложные программы, которые могут работать с несколькими операционными системами и ядрами. Найдя раздел, с которого будет выполняться загрузка системы, программа главной загрузочной записи пытается запустить загрузочную программу, связанную с этим разделом. В случае успеха этой программе передаются полномочия по дальнейшей загрузке ядра.

## 3.3. GRUB: универсальный загрузчик

Программа GRUB (GRand Unified Boot loader), разработанная в рамках проекта GNU, используется как стандартный загрузчик в большинстве UNIX- и Linux-систем, оснащенных процессорами Intel. GRUB поставляется в составе многих дистрибутивов Linux и Solaris (с архитектурой x86) начиная с версии 10. Задача GRUB — выбрать из предварительно подготовленного списка ядро и загрузить его с опциями, заданными администратором.

Существует две ветви развития GRUB: оригинальная, официально именуемая "GRUB Legacy", и более новая — GRUB 2. Название "GRUB 2" может ввести в заблуждение, поскольку выпуски GRUB 2 в действительности имеют номера версий 1 и 2. Все наши примеры систем в настоящее время используют исходный вариант — GRUB Legacy, и именно эта версия описывается в данной книге. Версия GRUB 2 в концептуальном плане подобна исходной, но отличается синтаксисом файла конфигурации.

По умолчанию GRUB читает параметры загрузки из файла **/boot/grub/menu.lst** или **/boot/grub/grub.conf**. Считывание содержимого файла конфигурации происходит в период запуска (что само по себе уже впечатляет), и это позволяет реализовать динамические изменения при каждой загрузке системы. Файлы **menu.lst** и **grub.conf** отличаются незначительно, но имеют аналогичный синтаксис. Системы Red Hat используют файл **grub.conf**, а Solaris, SUSE и Ubuntu — файл **menu.lst**. Вот образец файла `grub.conf`.

```
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Enterprise Linux Server (2.6.18-92.1.10.el5)
     root (hd0,0)
     kernel /vmlinuz-2.6.18-92.1.10.el5 ro root=LABEL=/
```

В этом примере конфигурируется единственная операционная система, которая будет загружена автоматически (`default=0`), если в течение 10 секунд с клавиатуры не поступят никакие данные (`timeout=10`). Корневая файловая система для конфигурации "Red Hat Enterprise Linux Server" находится в разделе (`hd0,0`), что для GRUB означает обращение к *первому* разделу *первого* жесткого диска системы (нумерация "первый раздел" и "первый диск"определяется в BIOS).

Программа GRUB загрузит ядро из файла **/vmlinuz-2.6.18-92.1.10.el5**, а затем выведет начальный образ экрана, хранящийся в файле **/boot/grub/splash.xpm.gz**. Путь поиска ядра указывается относительно раздела загрузки, который обычно монтируется в каталоге **/boot**.

Программа GRUB поддерживает мощный интерфейс командной строки и ряд функций, которые позволяют редактировать записи файла конфигурации в ходе загрузки. Для того чтобы перейти в режим командной строки, в окне загрузки GRUB необходимо ввести команду **c**. Из командной строки можно загружать операционные системы, не отраженные в файле **grub.conf**, выводить на экран информацию о системе и выполнять простейшую проверку файловой системы. Загрузчик предоставляет также ряд функций, подобных функциям интерпретатора команд, в том числе — функции завершения не полностью введенных команд и перемещения курсора. Любые действия, которые могут быть выполнены с помощью файла **grub.conf**, могут быть выполнены и посредством командной строки загрузчика GRUB.

Нажатие клавиши <Tab> позволяет вывести на экран краткий список доступных команд. Некоторые из наиболее полезных команд перечислены в табл. 3.2.

**Таблица 3.2. Параметры командной строки GRUB**

| Команда | Назначение |
|---|---|
| `reboot` | Мягкая перезагрузка системы |
| `find` | Поиск файла во всех смонтированных логических разделах |
| `root` | Указание корневого устройства (логического раздела) |
| `kernel` | Загрузка ядра с корневого устройства |
| `help` | Вывод интерактивной справки по команде |
| `boot` | Загрузка системы с указанного образа ядра |

Для получения подробной информации о загрузчике GRUB и его параметрах командной строки обратитесь к официальному руководству: **`gnu.org/software/grub/manual`**.

## Параметры ядра

Программа загрузчика GRUB позволяет передавать ядру параметры командной строки. Как правило, эти параметры изменяют значения параметров ядра, вынуждают его опросить конкретные устройства, указывают путь поиска демона `init` или назначают конкретное корневое устройство. Несколько примеров этих параметров приведены в табл. 3.3.

**Таблица 3.3. Примеры параметров ядра времени загрузки**

| Параметр | Назначение |
|---|---|
| `acpi=off` | Отключает компоненты Advanced Configuration (усовершенствованный интерфейс конфигурации) и Power Interface (интерфейс управления питанием) |
| `init=/bin/bash` | Заставляет ядро запускать только интерпретатор **`bash`**; используется при восстановлении системы в случае сбоев |
| `root=/dev/foo` | Сообщает ядру о том, что корневым является устройство **`/dev/foo`** |
| `single` | Задает режим однопользовательской загрузки (только для Linux. Используйте ключ `-s` в системах Solaris — он предназначен для администраторов, знакомых со стандартом OpenBoot в других CPU-архитектурах) |

Параметры ядра, отредактированные во время загрузки, не сохраняются. Чтобы сохранить изменения на будущие перезагрузки, отредактируйте строку **`kernel`** в файле **`grub.conf`** или **`menu.lst`**.

Ядро Linux постоянно совершенствуется с помощью заплат, повышающих уровень безопасности, исправлений ошибок и новых функций. Но, в отличие от других программных пакетов, старые версии ядра обычно не заменяются новыми. Новые ядра инсталлируются наряду со старыми, чтобы в случае возникновения проблемы вы могли тут же вернуться к старому ядру. Это соглашение помогает администраторам отказаться от модернизации, если заплата ядра разрушает их систему. По прошествии некоторого времени загрузочные меню GRUB заполняются различными версиями ядра. Обычно выбор версии ядра по умолчанию не представляет опасности, но если окажется, что ваша система не загружается после добавления заплаты, позаботьтесь о возможности выбора другого ядра.

## Мультисистемная загрузка

Поскольку на одном персональном компьютере может быть установлено несколько операционных систем, привычной является ситуация, когда компьютер загружается в мультисистемном режиме. Для этого загрузчик должен быть правильно сконфигурирован, чтобы распознать имеющиеся на локальных дисках операционные системы. В следующих разделах мы опишем некоторые проблемные блоки мультисистемной загрузки, а затем рассмотрим конкретные примеры конфигурации.

В каждом разделе диска может храниться собственный вторичный загрузчик, однако загрузочный диск должен иметь только одну главную загрузочную запись. Поэтому необходимо решить, какой загрузчик будет "главным". Как правило, выбор диктуется особенностями имеющихся операционных систем. Для Intel-ориентированных UNIX- и Linux-систем лучше всего в качестве главного загрузчика выбрать GRUB. При двухвариантной загрузке Windows-системы всегда используйте загрузчик GRUB (а не загрузчик Windows).

Конфигурирование загрузчика GRUB для выполнения мультисистемной загрузки во многом аналогично конфигурированию загрузки только одной операционной системы. Прежде чем вносить изменения в файл **grub.conf** или **menu.lst**, нужно установить желаемые операционные системы.

Записанные в файле **grub.conf** параметры конфигурации для загрузки операционной системы Windows отличаются от параметров для загрузки UNIX или Linux.

```
title Windows XP
    rootnoverify (hd0,0)
    chainloader +1
```

Параметр `chainloader` загружает утилиту начальной загрузки из указанного места (в данном случае из сектора 1 первого раздела первого IDE-диска). Параметр `rootnoverify` предотвращает попытки загрузчика GRUB выполнить монтирование указанного раздела.

Ниже приведен полный текст файла **grub.conf** для случая, когда система Windows XP загружается (по умолчанию) из первого раздела, Red Hat Enterprise Linux — из второго.

```
default=0
timeout=5
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
hiddenmenu
title Windows XP
    rootnoverify (hd0,0)
    chainloader +1
title Red Hat
    root (hd0,1)
    kernel /vmlinuz
```

Тот факт, что загрузчик GRUB решает многие потенциальные проблемы мультисистемной загрузки, в действительности не смягчает наш врожденный скептицизм в отношении мультисистемных конфигураций. Дополнительно об этом можно прочитать в разделе 30.3.

# 3.4. Загрузка в однопользовательском режиме

То, как начинается процесс загрузки, зависит от конкретной системы. Системы с не Intel-процессорами используют специально разработанные программы загрузки, в то время как на персональных компьютерах, в основном, загрузчики стандартизированы (благодаря GRUB).

## Однопользовательский режим при использовании GRUB

Для того чтобы выполнить загрузку в однопользовательском режиме при использовании загрузчика GRUB, не нужно применять опции командной строки. Авторы этого загрузчика пришли к выводу, что параметры начальной загрузки должны легко поддаваться изменению и что клавиша <a> — вполне подходящее средство для решения этой задачи. Когда откроется экран начальной загрузки GRUB, выделите нужное ядро и нажмите клавишу <a>, чтобы дополнить опции начальной загрузки. Для того чтобы обеспечить загрузку в однопользовательском режиме, добавьте флаг **single** (или **-s** в системе Solaris) в конец существующих опций ядра. Пример типичной конфигурации мог бы выглядеть следующим образом.

```
grub append> ro root=LABEL=/ rhgb quiet single
```

## Однопользовательский режим в архитектуре SPARC

Для того чтобы прервать процедуру загрузки и войти в среду OpenBoot PROM на Sun-оборудовании, нажмите одновременно клавиши <L1> и <a>. На современных Sun-клавиатурах клавиша <L1> иногда имеет надпись "STOP". Для того чтобы выполнить загрузку в однопользовательском режиме из среды OpenBoot PROM, вы можете ввести команду **boot -s**.

Для загрузки альтернативного ядра под управлением Solaris, как правило, нужно ввести полное Solaris-имя устройства и файла. Имя Solaris-устройства — это длинная странного вида строка символов, которую можно увидеть, введя команду `ls -l` для файла `/dev`.

```
% ls -l /dev/rdsk/c0t0d0s0
lrwxrwxrwx   1 root     root      55 Jan 15     1998 /dev/rdsk/c0t0d0s0 ->
    ../../devices/sbus@1f,0/SUNW,fas@e,8800000/sd@0,0:a,raw
```

Для того чтобы загрузить ядро, хранимое в виде файла **/kernel/backup** на том же диске, введите следующую команду в режиме монитора OpenBoot PROM.

```
boot /devices/sbus@1f,0/SUNW,fas@e,8800000/sd@0,0:a,raw/kernel/backup
```

Некоторые полезные команды, которые можно ввести из Sun-среды OpenBoot PROM, кратко описаны в табл. 3.4.

**Таблица 3.4. Некоторые PROM-команды для выполнения на Sun-оборудовании**

| Команда | Назначение |
|---|---|
| `boot /path_to_kernel` | Загружает альтернативное ядро |
| `boot -s` | Загружает систему в однопользовательском режиме |
| `boot -r` | Распознает ядро и определяет наличие новых устройств |
| `boot -a /etc/system.bak` | Принуждает ядро читать файл **/etc/system.bak** вместо **/etc/system** |
| `probe-scsi` | Отображает список всех подключенных SCSI-устройств |

## Однопользовательский режим на рабочих станциях HP-UX

Процедура загрузки в однопользовательском режиме на рабочих станциях HP-UX зависит от типа машины. Следующий пример взят с рабочей станции HP 9000/735.

Сначала прервите процесс загрузки после соответствующего напоминания. Получив приглашение на ввод команд, введите команду **boot pri isl**. Эта команда сгенерирует следующее приглашение, которое позволит вам загрузить систему в однопользовательском режиме. Это приглашение должно выглядеть примерно так.

```
ISL> prompt:
```

Следующая команда выбирает ядро и загружает систему в однопользовательском режиме.

```
ISL> prompt: hpux -iS /stand/vmunix
```

## Однопользовательский режим в системах AIX

В системах AIX однопользовательский режим называется "профилактическим". Для его установки, если система еще не загружена, достаточно выбрать соответствующий пункт из меню загрузки. Если система уже загружена, используйте команду **telinit S**.

# 3.5. Работа со сценариями запуска системы

После выхода из однопользовательского режима (или — при стандартной загрузке — по завершении работы интерпретатора команд, запущенного с правами суперпользователя) демон **init** выполняет сценарии запуска системы. Они являются сценариями интерпретатора **sh** (на самом деле **bash**), а их местонахождение, содержимое и организация зависят от изготовителя системы.

В большинстве систем используется подход, в котором сценарии нумеруются и выполняются по порядку. Сценарии хранятся в каталоге **/etc/init.d**, а ссылки на них созданы в каталогах **/etc/rc0.d**, **/etc/rc1.d** и т.д. Такая организация совершенно ясна, и поскольку сценарии выполняются по порядку, система может обеспечивать соответствующие зависимости между службами. Эти "пусковые" сценарии как запускают службы, так и останавливают их, поэтому такая архитектура также позволяет надлежащим образом остановить систему.

Ниже приведен перечень задач, которые часто выполняются этими сценариями.

- Задание имени компьютера
- Установка часового пояса
- Проверка дисков с помощью команды **fsck**
- Монтирование дисков систем
- Удаление старых файлов из каталога **/tmp**
- Конфигурирование сетевых интерфейсов
- Запуск демонов и сетевых служб

Сценарии запуска обычно довольно многословны и выводят подробную информацию о выполняемых ими задачах. Это может оказать существенную помощь при отладке сценариев или поиске причин зависания системы в процессе загрузки.

Администраторам не следует заниматься модификацией сценариев запуска. Те сценарии, которые принимают данные конфигурации, читают их в форме отдельного файла конфигурации (специфичного для конкретного узла). Вы можете модифицировать вспомогательный сценарий конфигурации и должны быть уверены, что он не будет перезаписан обновлениями.

Сценарии процесса `init` используются в определенной степени всеми шестью операционными системами, которые рассматриваются здесь в качестве примеров. Процесс запуска системы Solaris 10 был полностью переработан (см. раздел 3.6). Ubuntu использует "версию" процесса `init`, известную как Upstart, но мы все же рассмотрим ее в этом разделе, поскольку она имеет сходство с традиционным процессом `init`.

В следующих разделах мы сначала представим общие принципы системы, а затем опишем индивидуальные особенности каждой системы.

## Демон init и его уровни выполнения

Демон `init` — это первый процесс, который выполняется после завершения загрузки системы, и во многих отношениях он считается самым важным. Его идентификационный номер (PID) всегда равен 1, и сам он является базовым для всех пользовательских и почти для всех системных процессов. Реализации демона `init` незначительно различаются в разных системах.

Этот демон определяет по меньшей мере семь уровней выполнения, на каждом из которых должен выполняться конкретный набор системных служб. Точное определение каждого уровня в разных системах имеет свои особенности, но следующая информация справедлива для всех систем.

- Уровень 0 означает, что система полностью прекратила работу.
- Уровень 1 и S означает однопользовательский режим.
- Уровни 2–5 предназначены для поддержки работы в сети.
- Уровень 6 определяет этап перезагрузки системы.

Уровни 0 и 6 характерны тем, что система не может на них оставаться. Если она на них переходит, то в качестве побочного эффекта происходит завершение работы или перезагрузка. В большинстве случаев система по умолчанию находится на уровне 2 или 3. В Linux уровень 5 используется регистрационными процессами X Windows. Уровень 4 используется редко.

Однопользовательскому режиму традиционно соответствовал уровень 1. На нем запрещены все сетевые сеансы и процессы удаленной регистрации, а в системе выполняется минимальный набор программ. Но поскольку в этом режиме доступ к системе осуществляется с правами суперпользователя, администраторам необходимо, чтобы при загрузке система запрашивала ввод пароля.

Для этой цели был введен уровень S: здесь создается отдельный процесс, отображающий приглашение ввести пароль. В Solaris и AIX уровень S означает, что однопользовательский режим "в действии", но в Linux уровень S носит переходный характер и завершается сразу после ввода пароля.

Создается впечатление, что уровней выполнения больше, чем нужно. Обычно это объясняется тем, что в телефонном коммутаторе 7 уровней, поэтому считалось, что

в UNIX-системе должно быть как минимум столько же. В действительности в системах Linux и AIX поддерживается до десяти уровней, хотя большинство уровней не определено. В стандартной конфигурации системы AIX значимым является только уровень 2. Уровни 0 и 1 зарезервированы для операционной системы, а уровни 3–9 открыты для использования администраторами.

В файле **/etc/inittab** содержатся параметры, определяющие, что должен делать демон **init** на каждом уровне. Формат файла зависит от системы, но основная идея состоит в том, что в нем задаются команды, которые должны быть выполнены (или продолжить выполнение), когда система переходит на конкретный уровень.

В процессе загрузки демон **init** последовательно продвигается от уровня 0 к уровню по умолчанию, заданному в файле **/etc/inittab**. Чтобы осуществить переход между соседними уровнями, демон выполняет команды из этого файла. Аналогичные действия, только в обратном порядке, происходят при останове системы.

Команда **telinit** позволяет изменить уровень выполнения демона **init**, когда система полностью функциональна. Например, команда **telinit 3** переводит демон **init** на уровень 3. Самым полезным аргументом команды **telinit** является **-q**, который предписывает **init** перечитать файл **/etc/inittab**.

К сожалению, структура файла **inittab** несколько недоработана и недостаточно хорошо согласуется с реальным механизмом запуска и остановки служб в системах UNIX. Для того чтобы сделать этот файл более эффективным, в демоне **init** реализован дополнительный, абстрактный, уровень. Он представлен в виде команды, которая исходит из файла **inittab** и осуществляет смену уровней. Эта команда, в свою очередь, запускает сценарии из каталога, зависящего от целевого уровня. Они-то и переводят систему в новое состояние.


Этот дополнительный уровень не до конца разработан в системах AIX. Поэтому управление службами в системах AIX определяется только содержимым файла **inittab**. Кроме того, AIX-сценарии запуска немного отличаются от соответствующих сценариев в других системах.

Большинство современных дистрибутивов Linux по умолчанию выполняет загрузку на уровень 5, что может быть не подходящим для систем, которым не требуется запускать дисплейный сервер (в среде X Window System). Изменение используемого по умолчанию уровня выполнения не представляет сложности. Следующая строка из файла **inittab** компьютера, работающего под управлением системы SUSE, определяет уровень 5 выполнения в качестве используемого по умолчанию.

```
id:5:initdefault:
```

Системным администраторам обычно не приходится работать непосредственно с файлом **/etc/inittab**, так как существующие сценарии подходят для большинства случаев. Далее в главе мы практически не будем упоминать об этом файле и взаимодействии демона **init** со сценариями запуска системы. Просто, когда мы говорим о том, что демон **init** выполняет такой-то сценарий, нужно понимать следующее: связь со сценарием может быть косвенной.

## Обзор сценариев запуска

Основные копии сценариев запуска хранятся в каталоге **/etc/init.d**. Каждый сценарий отвечает за запуск одного демона или определенной подсистемы. Сценариям можно передавать аргументы **start** и **stop**, которые означают, что соответствующая

служба должна быть запущена либо остановлена. Большинство сценариев понимает также аргумент **restart**, который эквивалентен связке **stop+start**. Обладая правами системного администратора, можно вручную запускать или останавливать нужные службы, вызывая соответствующий сценарий из каталога **init.d** и передавая ему требуемый аргумент.

Ниже показан несложный сценарий, позволяющий запускать, останавливать и перезапускать демон **sshd**.

```
#!/bin/sh
test -f /usr/bin/sshd || exit 0
case "$1" in
   start)
       echo -n "Starting sshd: sshd"
       /usr/sbin/sshd
       echo "."
       ;;
   stop)
       echo -n "Stopping sshd: sshd"
       kill `cat /var/run/sshd.pid`
       echo "."
       ;;
   restart)
       echo -n "Stopping sshd: sshd"
       kill `cat /var/run/sshd.pid`
       echo "."
       echo -n "Starting sshd: sshd"
       /usr/sbin/sshd
       echo "."
       ;;
   *)
       echo "Usage: /etc/init.d/sshd start|stop|restart"
       exit 1
       ;;
esac
```

Сценарии каталога **/etc/init.d** способны запускать и останавливать отдельные службы, но, чтобы перейти на требуемый уровень, главный управляющий сценарий, выполняемый демоном **init**, должен получить дополнительную информацию о том, какие сценарии и с какими аргументами нужно запустить. Управляющий сценарий не просматривает непосредственно каталог **init.d**, а обращается к каталогу **rc***уровень***.d**, где *уровень* — это номер требуемого уровня выполнения, на который осуществляется переход (**rc0.d**, **rc1.d** и т.д.).

В каталогах **rc***уровень***.d** обычно содержатся символические ссылки на сценарии каталога **init.d**. Имена ссылок начинаются с префикса **S** или **K**, за которым следует номер и имя службы, управляемой сценарием (например, **S34named**).

Если демон **init** переходит на более высокий уровень, он выполняет все сценарии с префиксом **S** ("*s*tart" — запуск) в порядке возрастания номеров, причем каждому сценарию передается аргумент **start**. Когда осуществляется переход на более низкий уровень, запускаются сценарии с префиксом **K** ("*k*ill" — уничтожить) в порядке убывания номеров, и всем им передается аргумент **stop**.

Эта схема позволяет администраторам очень точно управлять порядком запуска служб. Например, запуск сценария **sshd** до запуска сетевых интерфейсов лишен смысла. Хотя в системе Red Hat и сеть, и сценарий **sshd** выполняются на уровне 3, сценарию

**network** присваивается порядковый номер 10, а сценарию **sshd** — 55. Поэтому можно не сомневаться, что сценарий **network** будет запущен раньше. При добавлении новых служб необходимо учитывать подобные взаимосвязи.

Чтобы сообщить системе, когда следует запускать тот или иной демон, нужно создать символическую ссылку в соответствующем каталоге. Например, следующие команды информируют систему о том, что демон печати `cupsd` должен быть запущен на уровне 2 и остановлен при завершении работы системы.

```
# ln -s /etc/init.d/cups /etc/rc2.d/S80cups
# ln -s /etc/init.d/cups /etc/rc0.d/K80cups
```

Первая ссылка свидетельствует о том, что сценарий запуска **/etc/init.d/cups** должен быть выполнен одним из последних при переходе на уровень 2 и ему должен быть передан аргумент **start**. Вторая ссылка сообщает, что в процессе завершения работы системы сценарий **/etc/init.d/cups** должен быть запущен относительно рано, причем с аргументом **stop**. В некоторых системах процессы останова и перезагрузки трактуются по-разному, поэтому необходимо также создать символическую ссылку в каталоге **/etc/rc6.d**, чтобы обеспечить корректный останов демона при перезагрузке системы.

## Сценарии запуска в системах Red Hat

Сценарии запуска в дистрибутивах Linux сильно отличаются друг от друга. В основу сценариев Red Hat положен подход, реализованный в демоне **init**, причем в некоторых строках разобраться довольно трудно.

На каждом уровне выполнения демон **init** вызывает сценарий **/etc/rc.d/rc**, передавая ему номер уровня в качестве аргумента. Этот сценарий может выполняться как в обычном режиме, так и в режиме подтверждения, когда перед выполнением каждого сценария выдается запрос.

Сценарии запуска хранят файлы блокировок в каталоге **/var/lock/subsys**. Наличие файла блокировок (с таким же именем, как у сценария запуска) служит признаком того, что данная служба уже "занята". Сценарии запуска создают файлы блокировок при выдаче команды **start** и удаляют их при выполнении команды **stop**.

В системах Red Hat управлять службами можно с помощью команды **chkconfig**. Эта команда добавляет или удаляет сценарии запуска из системы, управляет уровнями выполнения, на которых они работают, и выводит уровни, для которых сконфигурирован сценарий. Для получения информации о применении этого простого и удобного средства можно использовать команду **man chkconfig**.

В Red Hat также предусмотрен сценарий **/etc/rc.d/rc.local** (не каталог), который запускается во время загрузки. Этот последний сценарий, выполняемый как часть процесса загрузки, позволяет добавлять нюансы, характерные для конкретного узла или для выполнения задач после загрузки.

Когда появится сообщение "Welcome to Red Hat Enterprise Linux", можно нажать клавишу <i>, чтобы продолжить загрузку в режиме подтверждения. Однако подтверждение о нажатии клавиши не отображается. Система продолжит монтировать локальные файловые системы, активизировать разделы подкачки, загружать таблицы клавиш и вести поиск модулей ядра. Только после перехода на уровень 3 система начнет выдавать запросы на подтверждение.

Режимы интерактивной и однопользовательской загрузки начинаются в одной и той же точке. Если в процессе загрузки возникли серьезные проблемы и этой точки достичь невозможно, воспользуйтесь для загрузки DVD-диском или устройством USB (флешкой).

Можно передать ядру параметр **init=/bin/sh**, чтобы заставить его вызвать интерпретатор команд однопользовательского режима еще до того, как будет запущен демон **init**[2]. В этом случае все действия по запуску системы придется осуществлять вручную, включая выполнение команды **fsck** и монтирование локальных файловых систем.

Повлиять на процесс начальной загрузки в Red Hat можно путем модификации конфигурационных файлов, находящихся в каталоге **/etc/sysconfig**. Назначение элементов этого каталога описано в табл. 3.5.

**Таблица 3.5. Файлы и подкаталоги каталога /etc/sysconfig в Red Hat**

| Файл/подкаталог | Назначение или содержимое |
|---|---|
| clock | Задает тип системных часов (почти всегда UTC) [a] |
| console | Загадочный каталог, который всегда пуст |
| crond | Перечисляет аргументы, передаваемые демону cron |
| i18n | Содержит региональные установки системы (форматы представления даты/времени, языки и т.д.) |
| init | Определяет способ отображения сообщений, поступающих от сценариев запуска системы |
| keyboard | Задает тип клавиатуры (используйте идентификатор "us" для стандартной 101-клавишной клавиатуры) |
| mouse | Задает тип мыши; используется системой X и программой gpm |
| network | Задает глобальные сетевые опции (имя компьютера, шлюз, переадресация и т.д.) |
| network-scripts | Каталог, содержащий вспомогательные сценарии и сетевые конфигурационные файлы |
| sendmail | Задает параметры программы sendmail |

[a] При использовании мультисистемной загрузки совершенно непонятно, какой часовой пояс следует устанавливать для системных часов.

Для некоторых элементов списка необходимы дополнительные пояснения.

Файл **network** содержит адрес шлюза, имя хоста и другие важные установки, которые действуют по умолчанию и применяются ко всем сетевым интерфейсам.

Каталог **network-scripts** содержит вспомогательные файлы, связанные с конфигурацией сети. Единственное, что может потребоваться в нем изменить, — это файлы с именами **ifcfg-***интерфейс*. Например, файл **network-scripts/ifcfg-eth0** хранит параметры конфигурации интерфейса **eth0**, в частности его IP-адрес и сетевые опции. Подробнее конфигурирование сетевых интерфейсов рассматривается в разделе 14.10.

Файл **sendmail** содержит две переменные: DAEMON и QUEUE. Если переменная DAEMON равна значению yes, система запустит программу **sendmail** в режиме демона (**-bd**) в процессе начальной загрузки. Переменная QUEUE информирует программу **sendmail** о том, каков интервал обработки очереди почтовых сообщений (**-q**). Стандартная установка — 1 час.

[2] Однажды в нашей системе был поврежден файл, содержащий таблицу клавиш, и поскольку система Red Hat загружала этот файл даже в однопользовательском режиме, переход в этот режим не позволял решить проблему. Передача параметра *init=/bin/sh* оказалась единственным безопасным способом загрузить систему и исправить ошибку. Этот прием эффективен и в других

## Сценарии запуска в системах SUSE

Сценарии запуска систем SUSE подобны сценариям Red Hat, по крайней мере, по части общей организации. Эти сценарии не только четко организованы, но надежны и хорошо документированы. Стоит сказать спасибо разработчикам, которые отвечают за эту часть операционной системы.

Как и в системе Red Hat, на каждом уровне выполнения демон **init** вызывает сценарий **/etc/rc.d/rc**, передавая ему номер уровня в качестве аргумента. Сценарии данного пакета хранятся в каталоге **/etc/init.d**, а их файлы конфигурации — в каталоге **/etc/sysconfig**. Прекрасное краткое описание процесса запуска системы SUSE можно найти в файле **/etc/init.d/README**.

Хотя файлы конфигурации запуска как системы SUSE, так и Red Hat собраны в каталоге **/etc/sysconfig**, конкретные файлы внутри этого каталога в значительной мере различны. (В частности, в целом файлы SUSE сопровождаются достаточно подробными комментариями.) Необходимые параметры вызываются посредством установки переменных среды интерпретатора, к которым затем обращаются сценарии, хранящиеся в каталоге **/etc/init.d**. Некоторые подсистемы требуют более подробного конфигурирования. Такие системы, нуждающиеся в нескольких файлах конфигурации, имеют приватные подкаталоги вроде **sysconfig/network**.

Файл **windowmanager** — типичный пример файла, хранящегося в каталоге **sysconfig**.

```
## Path:        Desktop/Window manager
## Type:        string(gnome,startkde,startkde3,startxfce4,twm)
## Default:          kde
## Config:           profiles,kde,susewm

# Здесь можно установить стандартный оконный менеджер (kde, fvwm, ...)
# Эти изменения требуют перерегистрации

DEFAULT_WM="gnome"

## Type:        yesno
## Default:     yes

# инсталляция расширения SuSE для новых пользователей
# (тема и дополнительные функции)

INSTALL_DESKTOP_EXTENSIONS="yes"

## Path:              Desktop
## Description:       стандартный вид курсора мыши
## Type:              string
## Default:

# Название вида курсора мыши для системы X11. Возможные изображения
# можно найти в каталоге /usr/share/icons/

X_MOUSE_CURSOR="DMZ"
KDE_USE_IPV6="yes"
```

Каждой переменной предшествует информация о конфигурации, считываемая утилитой YaST[3], и подробное описание назначения переменной. Например, в файле **windowmanager** переменная DEFAULT_WM определяет диспетчер окна рабочего стола, используемый системой X.

Пакет SUSE содержит также команду **chkconfig**, которая позволяет управлять сценариями запуска системы. Она коренным образом отличается от версии, предоставляемой системой Red Hat, но также является эффективным средством управления сценариями.

## Сценарии запуска Ubuntu и демон Upstart

Начиная с выпуска "Feisty Fawn" в начале 2007 года, в системах Ubuntu традиционный демон init был заменен службой начальной загрузки на основе событий Upstart, которая используется и некоторыми другими дистрибутивами Linux. Служба Upstart обрабатывает переходы из одного состояния системы в другое (например, при изменении состава оборудования) более элегантно, чем это делает демон **init**. Кроме того, Upstart значительно сокращает время загрузки.

Служба Upstart организует запуск и останов других служб и задач в ответ на такие события системы, как добавление устройства или отключение сетевого диска. Из соображений совместимости служба Upstart также эмулирует традиционные уровни выполнения демона **init**. Но сценарии запуска и останова обрабатываются несколько не так, как при использовании демона **init**.

Демон Upstart использует файлы определения заданий из каталога **/etc/event.d** (а не файл **inittab**). Задание в концептуальном плане подобно сценарию загрузки, т.е. оно выполняет некоторую последовательность команд, а затем возвращает управление демону Upstart. Коллекция заданий в системе Ubuntu выглядит так.

```
ubuntu$ ls /etc/event.d
control-alt-delete  last-good-boot  logd  rc0  rc1  rc2  rc3  rc4  rc5  rc6
  rc-default  rcS  rcS-sulogin  sulogin  tty1  tty2  tty3  tty4  tty5  tty6
```

Со временем все больше сценариев загрузки будут преобразованы в Upstart-задания, а пока для загрузки системы демон Upstart использует сценарии эмуляции уровней выполнения. Например, сценарий **rc2** (в виде файла **/etc/rc2.d/rc**) выполняет все сценарии запуска для уровня 2.

Из-за необходимости поддержки этой совместимости Ubuntu-администраторы должны использовать Ubuntu-команду **update-rc.d** для обслуживания ссылок на сценарии запуска в рамках **rc**-каталогов. Вот как выглядит синтаксис этой команды.

**update-rc.d** *служба* { **start** | **stop** } *последовательность уровней* .

Команда **update-rc.d** принимает в качестве аргументов порядковый номер (имеется в виду порядок выполнения сценариев загрузки) и нужные уровни выполнения. Для обозначения конца списка уровней используется символ точки.

Службы, которые при смене уровня стартуют позже, должны останавливаться раньше при выходе системы с данного уровня. Например, если сервер печати CUPS во время загрузки запускается с порядковым значением 80, он должен остановиться приблизи-

[3] YaST — специализированная утилита графической конфигурации SUSE, которая управляет многими аспектами этой системы.

тельно с номером 20, т.е. поближе к началу процесса останова. Команда **update-rc.d**, добавляющая соответствующие ссылки, должна иметь такой вид.

```
ubuntu$ sudo update-rc.d cups start 80 2 3 4 5 . stop 20 S 1 6 .
   Adding system startup for /etc/init.d/cups ...
     /etc/rc1.d/K20cups -> ../init.d/cups
     /etc/rc6.d/K20cups -> ../init.d/cups
     /etc/rcS.d/K20cups -> ../init.d/cups
     /etc/rc2.d/S80cups -> ../init.d/cups
     /etc/rc3.d/S80cups -> ../init.d/cups
     /etc/rc4.d/S80cups -> ../init.d/cups
     /etc/rc5.d/S80cups -> ../init.d/cups
```

Эта команда добавляет "стартовые" экземпляры сценариев с порядковым номером 80 на уровнях выполнения 2–5, а также — "финишные" экземпляры с порядковым номером 20 на уровнях S, 1 и 6.

Устанавливаемый по умолчанию уровень управляется двумя **telinit**-строками в файле **/etc/event.d/rc-default**. Изменить уровень выполнения можно, отредактиров файл **rc-default** в любом текстовом редакторе.

Демон Upstart также управляет логинами на терминалах, используя задания с **tty***-именами.

## Сценарии запуска в системах HP-UX

В системах HP-UX сценарии запуска хранятся в каталоге **/sbin/init.d**. Каталоги уровней выполнения также хранятся в каталоге **/sbin**. Файлы конфигурации, связанные со сценариями запуска, обычно "прописаны" в каталоге **/etc/rc.config.d**. Их имена соответствуют именам сценариев запуска, хранимых в каталоге **/sbin/init.d**. Например, сценарий **/sbin/init.d/SnmpMaster** получает информацию о конфигурации системы из файла **/etc/rc.config.d/SnmpMaster**, а реально вызывается демоном **init** посредством ссылок **/sbin/rc2.d/S560SnmpMaster** и **/sbin/rc1.d/K440SnmpMaster**.

Система HP-UX сохраняет результаты выполнения сценариев запуска в файле **/etc/rc.log**. В случае неудачного завершения одного из сценариев запуска просмотрите содержимое файла **/etc/rc.log**, чтобы узнать, содержит ли он какие-либо релевантные сообщения об ошибках или подсказки относительно источника проблемы. Это сохранение результатов выполнения сценария запуска — самая полезная и замечательная функция, которую, к тому же, просто реализовать. Удивительно, что другие разработчики систем не ухватились за эту идею.

Файлы конфигурации, хранимые в каталоге **/etc/rc.config.d**, местами могут вводить в заблуждение, и это притом, что они в целом снабжены неплохими комментариями. Назначение некоторых чаще всего модифицируемых файлов описано в табл. 3.6.

Для большинства этих файлов вполне подходят стандартные установки. Наиболее часто вам придется работать с такими файлами, как **netconf**, **netdaemons** и, возможно, **nddconf**.

**Таблица 3.6. Конфигурационные файлы HP-UX из каталога /etc/rc.config.d**

| Файлы | Назначение |
|---|---|
| **SnmpMaster** | Включает или отключает поддержку протокола SNMP |
| **Snmp*** | Другие параметры, связанные с протоколом SNMP |

*Окончание табл. 3.6*

| Файлы | Назначение |
|---|---|
| **acct** | Включает или отключает подсистему учета процессов; см. `acct(1M)` |
| **auditing** | Управляет работой подсистемы аудита; см. `audsys` и `audevent` |
| **cde** | Содержит настройки CDE (Common Desktop Environment — Единая настольная среда) |
| **clean*** | Управляет операциями очистки, выполняемыми на этапе загрузки |
| **hpetherconf** | Конфигурирует Ethernet-интерфейсы; см. `lanadmin` |
| **lp** | Включает или отключает подсистему буферизации печати |
| **mailservs** | Запускает утилиту sendmail или задает почтовый сервер |
| **nameservs** | Конфигурирует или запускает демон службы имен |
| **nddconf** | Задает параметры ядра, устанавливаемые на этапе загрузки с помощью демона `ndd` |
| **netconf** | Задает параметры конфигурации сети (IP-адрес и т.п.) |
| **netdaemons** | Указывает на то, какие сетевые демоны следует запустить |
| **nettl** | Конфигурирует подсистемы сетевой трассировки и регистрации[a] |
| **nfsconf** | Задает параметры NFS (Network File System — Сетевая файловая система) |
| **sshd** | Конфигурирует параметры демона `SSH` |
| **vt** | Запускает демон `vtdaemon`; зависит от `ptydaemon` |
| **xfs** | Включает и отключает сервис шрифтов X Windows |

[a] См. `nettle(1M)`, `nettlconf(1M)` и `nettlgen.conf(4)`

## Запуск систем AIX


В системах AIX используется более "прямолинейный" подход (по сравнению с другими описываемыми здесь системами) к процессу загрузки. Во время запуска система AIX выполняет сценарий **/sbin/rc.boot**, который написан на языке, интерпретируемом командной оболочкой **ksh**.

Сценарий **rc.boot** (его код отличается скудными комментариями) выполняется в три этапа:

- инициализация системного оборудования;
- монтирование системных файлов;
- выполнение демона **/etc/init**, который обрабатывает записи файла **/etc/inittab**.

Система AIX в большей степени полагается на содержимое файла **/etc/inittab**, чем другие члены семейства UNIX-подобных систем. Демон **init** читает по очереди строки файла **inittab** и выполняет их. В некоторых случаях файл **inittab** запускает демоны напрямую. Например, при выполнении следующей строки запускается или перезапускается демон **cron** на уровнях выполнения 2–9.

```
cron:23456789:respawn:/usr/sbin/cron
```

Другие строки файла **inittab** содержат последовательность команд. Например, **bsh**-сценарий **/etc/rc.tcpip** запускает на выполнение сетевые демоны.

```
rctcpip:23456789:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
```

Здесь результат выполнения сценария **rc.tcpip** передается системной консоли. Демон **init** ожидает завершения этого сценария, а уж потом переходит к обработке следующей строки файла.

Для управления файлом **inittab** в системе AIX предусмотрены четыре простые команды: **mkitab**, **chitab**, **lsitab** и **rmitab**. Эти команды позволяют добавлять записи в файл, изменять их, выводить в виде списка и удалять из файла. Мы не понимаем, в чем состоит их преимущество, и предпочитаем редактировать файл **inittab** с помощью какого-нибудь текстового редактора (например, **vi**).

# 3.6. Загрузка систем SOLARIS

С помощью механизма управления службами SMF (Service Management Facility) компания Sun модернизировала процесс загрузки для систем Solaris 10 и OpenSolaris. Механизм SMF обеспечивает комплексный и принципиально новый подход к управлению службами в системах UNIX. Его уникальность состоит в надстройке нового логического уровня над службами, который управляет зависимостями между ними и автоматически обрабатывает ошибки в данных конфигурации и программные сбои.

Механизм SMF в значительной степени изменяет процедуру загрузки системы. Традиционная схема работы демона **init** и его **rc**-сценариев уходит в прошлое. Разработчики Sun заявляют, что современные приложения и их взаимозависимости стали слишком сложными для стандартной схемы управления ими. И они отчасти правы. С другой стороны, стандартная архитектура гораздо проще, и мы даже диву даемся, как Linux и другим популярным операционным системам удавалось работать при старой организации.

Прежде чем рассматривать процесс загрузки системы, опишем в общих чертах механизм SMF.

## Механизм управления службами в системе Solaris

Sun определяет службу как "средство предоставления приложениям некоторого списка возможностей и других служб, локальных и удаленных". С нашей точки зрения, служба — это аналог демона: веб-сервер, системный регистратор **syslogd** или даже **init**. Любая служба в системе может быть представлена несколькими экземплярами. Например, вы могли бы запустить на выполнение несколько почтовых серверов с различными конфигурационными данными и IP-адресами. Кроме того, службу можно определить как коллекцию других служб. Это свойство позволяет SMF взять на себя роль традиционных уровней выполнения демона **init**.

Каждый экземпляр службы уникальным образом определяется FMRI-идентификатором (*f*ault *m*anagement *r*esource *i*dentifier — идентификатор управляемого ресурса в системе управления сбоями). Например, следующие эквивалентные FMRI-идентификаторы ссылаются на службу `ssh`.

```
svc:/network/ssh:default
network/ssh:default
```

Служба `ssh` принадлежит категории `network`, и данный конкретный FMRI-идентификатор описывает стандартный экземпляр. В SMF определено несколько категорий, например `application`, `device`, `network` и `system`. Специальная категория `milestone` инкапсулирует концепцию уровней выполнения.

Экземпляр службы в каждый момент времени может находиться в одном из возможных семи состояний. Узнать состояние службы можно с помощью команды `svcs`. Чтобы получить полный список служб, определенных в системе, используйте команду `svcs -a`. Если опустить флаг `-a`, вы увидите только те службы, которые выполняются в данный момент. Команду `svcs` можно использовать и для отдельно взятой службы. Например, следующая команда проверяет состояние службы `ssh`.

```
solaris$    svcs -l svc:/network/ssh:default
fmri        svc:/network/ssh:default
name        SSH server
enabled     true
state       online.
next_state  none
state_time  Mon Jul 13 15:56:19 2009
logfile     /var/svc/log/network-ssh:default.log
restarter   svc:/system/svc/restarter:default
contract_id 65
dependency  require_all/none svc:/system/filesystem/local (online)
dependency  optional_all/none svc:/system/filesystem/autofs (online)
dependency  require_all/none svc:/network/loopback (online)
dependency  require_all/none svc:/network/physical (online)
dependency  require_all/none svc:/system/cryptosvc (online)
dependency  require_all/none svc:/system/utmp (online)
dependency  require_all/restart file://localhost/etc/ssh/sshd_config (online)
```

В этой командной строке используется полный FMRI-идентификатор (в отличие от сокращенного), но поскольку существует только один экземпляр этой службы, можно было бы ограничиться более лаконичной командой **`svcs -l ssh`**.

Состояние службы может принимать одно из следующих значений:

- `online` — запуск разрешен, экземпляр службы успешно работает;
- `disabled` — запуск запрещен; экземпляр службы не запущен;
- `degraded` — запуск разрешен, но работа происходит с ограниченной функциональностью;
- `legacy_run` — служба работает; некоторые унаследованные (от предыдущих версий системы) службы не подлежат управлению через SMF, однако их работу можно наблюдать стандартными для SMF средствами; это состояние возможно только для унаследованных служб;
- `uninitialized` — в этом состоянии находятся все службы до того, как их конфигурация будет считана из репозитория;
- `maintenance` — экземпляр службы аварийно завершился и не может быть запущен заново без вмешательства администратора;
- `offline` — запуск разрешен, но экземпляр службы еще не запущен или нет возможности его запустить из-за ожидания "неудовлетворенной зависимости" или по другой причине.

В дополнение к текущему состоянию службы, команда `svcs -l` позволяет узнать местоположение журнала регистрации службы, ее зависимости и другие важные данные.

С помощью зависимостей описываются взаимосвязи (различной сложности) между службами. Это средство, по сути, заменяет систему пронумерованных сценариев запуска, используемую традиционным демоном **`init`**, в котором сценарий с префиксом **`S20`** должен выполняться раньше сценария с префиксом **`S90`**.

Из приведенного выше примера видно, что служба SSH использует элементы локальных файловых систем, сетевые интерфейсы, службы **cryptosvc** и **utmp**, а также файл **sshd_config**. Средство автоматического монтирования файловых систем **autofs** отмечено как имеющее необязательную зависимость. Это означает, что служба SSH будет выполняться в случае, если автомонтировщик не работает (по намерению администратора) или если он нормально работает.

Команда **svcadm** изменяет состояние службы. Чтобы запретить запуск сервера SSH (не пытайтесь делать это удаленно!), используйте такую команду.

```
solaris$ sudo svcadm disable ssh
```

В этом примере используется сокращенный FMRI-идентификатор для сервера SSH, так как он уникальный. Такой запрет является устойчивым изменением; для временного же запрещения запуска службы SSH используйте команду **svcsadm -t**.

В системе SMF службы конфигурируются на основе XML-файлов, именуемых *манифестами* и *профилями*. Манифест содержит полное описание всех свойств службы (зависимостей и инструкций для ее запуска и останова). Файлы манифестов хранятся в каталоге **/var/svc/manifest**. Каждый экземпляр службы имеет собственный файл манифеста.

Строки **exec_method** в файле манифеста обычно указывают на сценарии, которые запускают или останавливают службы, т.е. во многом подобно тому, как используются сценарии в системе **init.d**. Например, процесс запуска для демона SSH определяется в файле **/var/svc/manifest/ssh.xml** следующим образом.

```
<exec_method
type='method'
name='start'
exec='/lib/svc/method/sshd start'
timeout_seconds='60'/>
```

Сценарий **/lib/svc/method/sshd**, на который здесь содержится ссылка, является **sh**-сценарием, запускающим службу **sshd**. Все это подозрительно напоминает сценарий, который когда-то хранился в каталоге **/etc/rc.d**, — чем больше мы изменяем жизнь, тем больше она остается прежней.

Файл профиля службы определяет, разрешено ли запускать данный экземпляр или запрещено. Профили хранятся в каталоге **/var/svc/profile**.

Постоянные данные о конфигурации для служб в действительности хранятся в виде файла SQLite-базы данных **/etc/svc/repository.db**. Таким образом, вам необязательно непосредственно модифицировать содержимое XML-файлов. Вместо этого вы можете управлять манифестами и профилями с помощью команды **svccfg**. Для управления службами, находящимися под контролем демона **inetd**, используйте команду **inetadm**. Для получения дополнительной информации о командах **svccfg**, **inetadm** и **svc.configd** обратитесь к соответствующим man-страницам.

Одной из самых интересных особенностей механизма SMF является использование так называемых "пускателей" (restarters), которые составляют часть разрекламированной в Solaris технологии "прогнозируемого самовосстановления". В результате анализа тщательно определенной системы зависимостей механизм SMF может предположительно определить причину сбоя службы и при необходимости перезапустить ее. Причиной отказа службы может служить ошибка в программном обеспечении, сбой в работе оборудования, проблема с зависимым элементом или даже ошибка администратора. Обозначенный в SMF процесс "пускателей", тесно связанный с ядром, автоматически выполняет действия, необходимые для восстановления.

## Прекрасный новый мир: загрузка с помощью механизма SMF

Начальная стадия загрузки в системе Solaris 10 (или более поздней версии) подобна обычному процессу самозагрузки. Загрузка в архитектуре SPARC несколько отличается от загрузки в Intel-системах, но общий принцип состоит в том, что низкоуровневая встроенная программа (PROM для SPARC, BIOS для Intel) считывает загрузочную запись, которая загружает ядро операционной системы.

Ядро просматривает каталог **/etc/system** в поиске загружаемых модулей ядра, затем активизирует демон **init**, который немедленно запускает процесс **svc.startd**. Этот процесс является главным SMF-пускателем, который отвечает за процессы запуска в порядке следования зависимостей, определенных в репозитории конфигурации SMF.

К сожалению, система уровней выполнения и **init**-сценарии из предыдущих версий Solaris не совсем "похоронены". Некоторые службы (например, те, которые в результатах выполнения команды **svcs -a** демонстрируют состояние **legacy-run**) по-прежнему опираются на сценарии, хранимые в каталоге **/etc/rc.d**. Это значит, что коллизия между SMF-службами и традиционными уровнями выполнения оставила массу "осколков".

Для того чтобы не "пораниться" об их острые края, следует иметь в виду следующие моменты.

- Службы в состоянии **legacy_run** были запущены из сценария **rc**.
- Solaris определяет восемь уровней выполнения. Подробнее см. man-страницу для демона **init**.
- Чтобы изменить уровни выполнения, используйте команду **init** *n*, где *n* — новый уровень выполнения. Не пытайтесь использовать механизм SMF для изменения служб, что, по мнению разработчиков Sun, "может вызвать неожиданное поведение".
- Демон **init** управляется с помощью содержимого файла **/etc/inittab** (практически как в Linux).

# 3.7. Перезагрузка и останов системы

Традиционные UNIX- и Linux-системы были очень требовательны в отношении процедуры выключения. Современные системы более терпимы (особенно если речь идет о высоконадежной файловой системе), но все же лучше корректно завершать работу, если это возможно. Неправильное выключение компьютера может привести к появлению трудно обнаруживаемых, неочевидных ошибок, а иногда и к полному краху системы. Базы данных (которые не прекращают свою работу корректно) "славятся" проблемами разрушения данных и их целостности.[4]

Перезагрузка системы на персональном компьютере — средство решения почти всех проблем. Проблемы, возникающие в Linux, как правило, скрытые и сложные, поэтому перезагрузка дает ожидаемый результат гораздо реже, чем в других системах.

Если вы модифицируете один из сценариев запуска системы или вносите в систему существенные изменения, то нужно перезагрузиться хотя бы для того, чтобы проверить, успешно ли функционирует система после вашего вмешательства. Если какая-либо про-

[4] Теоретически базы данных должны быть особенно устойчивыми к такой форме разрушения, но наш опыт свидетельствует об обратном.

блема не проявится в течение нескольких ближайших недель, впоследствии вы вряд ли вспомните подробности последних изменений.

## Команда shutdown: корректный способ останова системы

Команда **shutdown** — самый безопасный и корректный способ остановить или перезагрузить систему либо вернуться в однопользовательский режим. Она переносит нас во времена использования систем, работающих в режиме разделения времени, поэтому такой подход порой кажется анахроничным на настольных компьютерах.

К сожалению, почти все изготовители систем решили приложить "свою руку" к аргументам команды **shutdown**. Мы рассмотрим эту команду в общих чертах, а затем обсудим ее синтаксис и аргументы, используемые на каждой платформе.

Можно дать команде **shutdown** указание делать паузу перед остановом системы. Во время ожидания команда посылает зарегистрированным пользователям через постепенно укорачивающиеся промежутки времени сообщения, предупреждая о приближающемся событии. По умолчанию в сообщениях говорится о том, что система заканчивает работу, и указывается время до момента останова. При желании администратор может добавить собственное короткое сообщение, в котором поясняется, почему система останавливается и сколько примерно времени придется подождать, прежде чем снова можно будет войти в систему. После выполнения команды **shutdown** пользователи будут лишены возможности входа в систему, но они будут видеть сообщение, предусмотренное администратором.

С помощью команды **shutdown** (в большинстве ее версий) можно указать, что должна сделать система после выполнения команды: остановиться, перейти в однопользовательский режим или перезагрузиться. Можно также задать, должна ли после перезагрузки выполняться принудительная проверка дисков с помощью команды **fsck**. В современных системах с объемными дисками полное выполнение команды **fsck** может занять много времени; и эту проверку можно пропустить. (Большинство систем автоматически пропускает эту проверку, если файловые системы были корректно демонтированы.)

В табл. 3.7 приведены аргументы команды **shutdown** в разных системах.

**Таблица 3.7. Множество "ликов" команды shutdown**

| Система | Путь | Время | R[a] | H | S | F |
|---|---|---|---|---|---|---|
| Linux | **/sbin/shutdown** | time | **-r** | **-h** | **-** | **-f**[6] |
| Solaris | **/usr/sbin/shutdown** | **-g**secs | **-i6** | **-i0** | **-iS** | **-** |
| HP-UX | **/etc/shutdown** | secs | **-r** | **-h** | **--** | |
| AIX | **/sbin/shutdown** | +time | **-r** | **-h** | **-m** | **-** |

[a] R — перезагрузка, H — останов, S — вход в однопользовательский режим, F — пропуск команды **fsck**.
[6] Системы Red Hat и SUSE, но не Ubuntu.

Например, следующая Linux-команда напоминает пользователям о запланированной процедуре сервисного обслуживания и отключает систему в 9:30 утра.

```
$ sudo shutdown -h 09:30 "Going down for scheduled maintenance.
        Expected downtime is 1 hour."
```

Можно также задать относительное время отключения. Например, приведенная ниже команда запустит процесс выключения через 15 минут:

```
$ sudo shutdown -h +15 "Going down for emergency disk repair."
```

## Команды halt и reboot: более простой способ останова

Команда **halt** выполняет все основные операции, необходимые для останова системы. Обычно она вызывается командой **shutdown -h**, но может применяться и сама по себе. Команда регистрирует в журнальном файле факт останова, уничтожает несущественные процессы, выполняет системный вызов **sync**, дожидается завершения операций записи на диск, а затем прекращает работу ядра.

При наличии опции **-n** системный вызов **sync** подавляется. Команда **halt -n** используется после восстановления корневого раздела командой **fsck**, чтобы ядро не могло затереть исправления старыми версиями раздела, хранящимися в кеше.

Команда **reboot** почти идентична команде **halt**. Разница лишь в том, что система перезагружается, а не останавливается. Режим перезагрузки вызывается также командой **shutdown -r**.

# 3.8. Упражнения

3.1. Можно ли UNIX- или Linux-системы выключать нажатием кнопки питания на корпусе компьютера? А что скажете насчет выключения вилки компьютера из розетки на стене? Поясните свой ответ. Постарайтесь определить вероятность получения отрицательного ответа на поставленные вопросы с помощью Интернета.

3.2. Используйте опции командной строки загрузчика GRUB для загрузки ядра, информация о котором отсутствует в файле **grub.conf**.

3.3. ★ Объясните концепцию уровней выполнения. Перечислите уровни выполнения, существующие в одной из ваших локальных систем, и кратко опишите каждый из них. Чем отличаются концепции уровней выполнения в Ubuntu и других дистрибутивах Linux?

3.4. ★ Напишите сценарий, предназначенный для запуска некоего сетевого демона **/usr/local/sbin/foo**. Покажите, как организовать автоматический запуск сценария на этапе начальной загрузки системы.

3.5. ★★★ Если система находится на уровне выполнения 3 и вводится команда **telinit 1**, какие действия предпримет демон **init**? Каким будет конечный результат?

3.6. ★★★ Нарисуйте схему, показывающую очередность запуска демонов в вашей системе.

3.7. ★★★ Опишите последовательность действий, необходимых для инсталляции на одном компьютере операционных систем Linux и Windows. Используйте программу GRUB.

# Глава 4

# Управление доступом: сила привилегий


Управление доступом к ресурсам относится к сфере, которая активно исследуется в наши дни и долгое время была одной из самых проблемных в проектировании операционных систем. Для отдельных пользователей операционные системы определяют учетные записи, которые позволяют выполнять множество операций: редактирование текстовых файлов, регистрация на удаленных компьютерах, установка имени главного узла, инсталляция новых программ и т.д. Систему управления доступом можно рассматривать как черный ящик, который (на входе) принимает потенциальные действия (пары "пользователь/операция") и выдает (на выходе) решения в зависимости от допустимости каждого действия.

В системах UNIX и Linux не существует единого черного ящика, который реализует управление доступом. В действительности здесь можно говорить о целом хранилище черных ящиков — и это хранилище "съедает" память. В данной главе мы сначала вернемся к истокам UNIX (чтобы понять, как сложилась нынешняя ситуация с управлением доступом), а затем рассмотрим современные системы управления доступом в UNIX и Linux (как в теории, так и на практике), после чего остановимся на инструментах, которые помогают сделать администрирование сферы управления доступом (и особенно в части, связанной с привилегиями всемогущего суперпользователя) относительно безболезненным.

О том, как предотвратить несанкционированный доступ на уровне суперпользователя, рассказывается в главе 22. В главе 32 описываются административные аспекты этого процесса.

# 4.1. Традиционные методы управления доступом в системах UNIX

Даже в первых и самых простых версиях UNIX никогда не было единого "центра" управления доступом. Тем не менее существовали общие правила, которые оказывали влияние на проектирование систем.

- Объекты (например, файлы и процессы) имеют владельцев. Владельцы обладают обширным (но необязательно неограниченным) контролем над своими объектами.
- Вы являетесь владельцами новых объектов, создаваемых вами.
- Пользователь `root` с особыми правами, известный как *суперпользователь*, может действовать как владелец любого объекта в системе.
- Только суперпользователь может выполнять административные операции особого значения.

В системах UNIX единого черного ящика, управляющего доступом, не существует, поскольку код, который принимает решения в этой области, рассредоточен по всей системе. Одни системные вызовы (например, **settimeofday**) ограничены пользователем `root` (при выполнении таких системных вызовов просто проверяется "личность" текущего пользователя, и, если он окажется не суперпользователем, операция отвергается). Другие системные вызовы (например, **kill**) реализуют различные вычисления, которые требуют как соответствия владельцу, так и обеспечения специальных условий для суперпользователя. Наконец, файловая система реализует собственную систему управления доступом, причем эта система сложнее любой другой, реализованной в ядре. Например, только файловая система использует принцип UNIX-групп для управления доступом.

🕮 Подробнее о файлах устройств рассказывается в разделе 6.4.

Усложнение ситуации привело к переплетению ядра и файловой системы. Например, вы управляете большинством устройств и передаете им информацию посредством файлов, которые представляют их в каталоге **/dev**. Поскольку эти файлы устройств являются объектами файловой системы, они используются и в ее семантике управления доступом.

## Управление доступом в файловой системе

Каждый файл в традиционной модели UNIX-подобных систем принадлежит владельцу и группе, иногда именуемой "групповым владельцем". Владелец файла имеет особую привилегию, недоступную другим пользователям системы: ему разрешено менять права доступа к файлу. В частности, владелец может задать права доступа так, что никто, кроме него, не сможет обращаться к файлу[1]. Мы еще вернемся к теме прав доступа в главе 6.

🕮 Подробнее о группах написано в разделе 7.1.

Владельцем файла всегда является один человек, тогда как в группу владельцев могут входить несколько пользователей. По традиции информация о группах хранилась в файле **/etc/group**, но теперь ее чаще хранят на сетевом сервере NIS или LDAP. Подробнее эти вопросы рассматриваются в главе 19.

[1] В действительности права доступа можно установить такими строгими, что даже владелец файла не сможет им и воспользоваться.

Владелец файла должен определить, какие операции могут выполнять над файлом члены группы. Такая схема допускает совместное использование файлов членами одной группы. Например, мы применяем группу для управления доступом к файлам исходного кода веб-сайта `www.admin.com`

Узнать идентификаторы владельцев файла можно с помощью команды **`ls -l`** *имя_файла*.

Например:

```
aix$ ls -l /home/garth/todo
-rw-------  1  garth  staff  1258  Jun 4 18:15  /home/garth/todo
```

Файлом владеет пользователь **`garth`**, а группа, которой он принадлежит, называется **`staff`**. Буквы и дефисы в первом столбце означают права доступа к файлу (подробнее см. раздел 6.5).

🕮 Подробнее о файле **`/etc/passwd`** можно прочитать в разделе 7.1, а о файле **`/etc/group`** — в разделе 7.3.

Ядро и файловая система отслеживают не текстовые имена владельцев и групп, а их идентификаторы. В самом общем случае идентификаторы пользователей (сокращенно UID — User ID) и соответствующие им имена хранятся в файле **`/etc/passwd`**, а идентификаторы и названия групп (GID — group ID) находятся в файле **`/etc/group`**. Текстовые эквиваленты идентификаторов UID и GID определяются исключительно для удобства пользователей. Для того чтобы команда вроде **`ls`** могла вывести информацию о владельце файла в удобочитаемом виде, она должна просмотреть базу данных идентификаторов и найти в ней нужные имена.

## Владение процессом

Владелец может посылать процессу сигналы (описываются в разделе 5.3), а также понижать его приоритет. С процессами связано несколько идентификаторов: реальный, текущий и сохраненный идентификатор пользователя; реальный, текущий и сохраненный идентификатор группы, а в системе Linux — "идентификатор пользователя файловой системы", который используется только для определения прав доступа к файлам. Вообще говоря, реальные номера применяются для учета использования системных ресурсов, а текущие — для указания прав доступа. В большинстве случаев реальные и текущие идентификаторы совпадают.

Сохраненные идентификаторы не оказывают никакого непосредственного влияния. Они позволяют программам "приберегать" неактивные идентификаторы для последующего использования, тем самым способствуя расчетливому применению расширенных полномочий. Вообще говоря, идентификатор пользователя файловой системы — это особенность реализации сетевой файловой системы (Network File System — NFS), и обычно он совпадает с текущим идентификатором пользователя.

🕮 Подробнее о системе NFS написано в разделе 18.1.

## Учетная запись суперпользователя

Учетная запись `root` в UNIX принадлежит "всесильному" пользователю-администратору, известному как *суперпользователь*, хотя его настоящее имя — "root".

Определяющей характеристикой учетной записи суперпользователя является значение UID, равное нулю. Ничто не запрещает менять имя этой учетной записи или созда-

вать другую запись с нулевым идентификатором, но такие действия ни к чему хорошему не приведут. Их следствием будет возникновение новых брешей в системе защиты, а также растерянность и гнев других администраторов, которым придется разбираться с особенностями конфигурирования такой системы.

Традиционная система UNIX позволяет суперпользователю (т.е. всякому процессу, текущий идентификатор пользователя которого равен нулю) выполнять над файлом или процессом любую допустимую операцию[2].

Вот примеры операций, доступных лишь суперпользователю:

- изменение корневого каталога процесса с помощью команды **chroot**;
- создание файлов устройств;
- установка системных часов;
- увеличение лимитов использования ресурсов и повышение приоритетов процессов;
- задание сетевого имени компьютера;
- конфигурирование сетевых интерфейсов;
- открытие привилегированных сетевых портов (номера которых меньше 1024);
- останов системы.

Процессы суперпользователя обладают способностью изменять свои идентификаторы. Один из таких процессов — программа **login** и ее графические эквиваленты, которые отображают на экране приглашение ввести пароль при входе в систему. Если введенные пароль и имя пользователя правильны, программа заменяет свои идентификаторы соответствующими идентификаторами указанного пользователя и запускает интерпретатор команд. После того как процесс суперпользователя, сменив владельца, станет обычным пользовательским процессом, восстановить свое предыдущее привилегированное состояние он уже не сможет.

## Использование битов "setuid" и "setgid"

Традиционная система управления доступом в UNIX дополнена системой смены полномочий, которая реализуется ядром в сотрудничестве с файловой системой. Более детально эта система описана в разделе 6.5, но, если кратко, то эта система позволяет выполнять специально подготовленные файлы с использованием привилегий более высокого уровня (обычно это привилегии суперпользователя). Этот механизм разрешает разработчикам и администраторам создавать условия для непривилегированных пользователей, при которых они могут выполнять привилегированные операции.

Дело в том, что существует два специальных бита, устанавливаемых в маске прав доступа к файлу: "setuid" (Set User ID — бит смены идентификатора пользователя) и "setgid" (Set Group ID — бит смены идентификатора группы). Если запускается исполняемый файл, у которого установлен один из этих битов, то текущими идентификаторами создаваемого процесса становятся идентификаторы владельца файла, а не идентификаторы пользователя, запустившего программу. Смена полномочий действительна только на время работы программы.

Например, пользователи должны иметь возможность изменять свои пароли. Но поскольку пароли хранятся в защищенном файле **/etc/shadow**, пользователям нужно использовать команду **passwd** с полномочиями **setuid**, чтобы "усилить" свои права

[2] Следует подчеркнуть важность слова "допустимую". Некоторые операции (например, запуск файла, для которого не установлен бит выполнения) запрещены даже суперпользователю.

доступа. Команда **passwd** проверяет, кто ее выполняет, и, в зависимости от результата, настраивает свое поведение соответствующим образом, поэтому обычные пользователи могут менять только собственные пароли, а суперпользователь — любые. (Это, между прочим, еще один пример "специального случая" в UNIX-системе управления доступом — правила "встроены" в код команды **passwd**.)

## 4.2. Современная организация управления доступом

В предыдущем разделе были рассмотрены практически все основные принципы построения традиционной UNIX-модели. Несмотря на то что систему управления доступом можно описать буквально двумя страницами, она выдержала проверку временем благодаря своей простоте, предсказуемости и способности удовлетворять большинство требований, предъявляемых к управлению доступом на среднестатистическом узле. Все UNIX- и Linux-версии продолжают поддерживать эту модель, которая широко используется в наши дни. И если не считать разделы нашей книги, в которых уделяется внимание некоторым специальным альтернативным вариантам, мы предполагаем, что вы используете именно традиционную модель.

Тем не менее мы не можем не упомянуть очевидные недостатки этой модели.

- С точки зрения безопасности, суперпользователь представляет единственную учетную запись, которая может являться причиной потенциального отказа системы. Если суперпользователь дискредитирует себя, может нарушиться целостность всей системы. Взломщик в этом случае может нанести системе колоссальный вред.
- Единственный способ разделить специальные привилегии суперпользователя состоит в написании **setuid**-программ. К сожалению, как показывает непрекращающийся интернет-поток обновлений средств защиты систем, довольно трудно написать по-настоящему безопасное программное обеспечение. К тому же, вам не следует писать программы, назначение которых можно было бы выразить такими словами: "Хотелось бы, чтобы эти три пользователя могли выполнять задачи резервирования на файловом сервере".
- Модель безопасности не является достаточно прочной для использования в сети. Ни один компьютер, к которому непривилегированный пользователь имеет физический доступ, не может гарантировать, что он точно представляет принадлежность выполняемых процессов. Кто может утверждать, что такой-то пользователь не переформатировал диск и не инсталлировал собственную копию Windows или Linux со своими UID?
- Во многих средах с высокими требованиями к защите системы действуют соглашения, которые просто не могут быть реализованы с использованием традиционных UNIX-систем безопасности. Например, согласно государственным стандартам США компьютерные системы должны запрещать привилегированным пользователям (например, тем, кто относится к высшей категории допуска) публиковать документы особой важности на более низком уровне безопасности. Традиционная же организация безопасности в UNIX зависит от доброй воли и профессиональных навыков отдельных пользователей.
- Поскольку многие правила, связанные с управлением доступа, встроены в код отдельных команд и демонов, невозможно переопределить поведение системы, не модифицируя исходный код и не перекомпилируя программы. Но это реально не практикуется.

- Для отслеживания действий пользователей предусмотрена минимальная поддержка. Вы можете легко узнать, к каким группам принадлежит тот или иной пользователь, но вы не в состоянии точно определить, какие действия разрешены пользователю членством в этих группах.

Из-за этих недостатков UNIX- и Linux-системы многократно претерпевали различные изменения. Их цель — усовершенствовать различные аспекты подсистемы управления доступом и сделать UNIX-системы более пригодными для сайтов с высокими требованиями к безопасности. Одни корректирующие технологии, такие как PAM (подробнее чуть ниже), в настоящее время имеют практически универсальную поддержку. Другие системные расширения можно назвать уникальными. Самые популярные из них рассмотрим в следующем разделе.

## Управление доступом на основе ролей

Управление доступом на основе ролей (*r*ole-*b*ased *a*ccess *c*ontrol — RBAC) — теоретическая модель, формализованная в 1992 г. Дэвидом Ферраильо (David Ferraiolo) и Риком Куном (Rick Kuhn). В основе этой модели лежит идея добавления уровня косвенности в механизм управления доступом. Вместо того чтобы назначать полномочия непосредственно пользователям, они назначаются промежуточным конструкциям, именуемым "ролями", а роли, в свою очередь, назначаются пользователям. Для того чтобы принять решение по управлению доступом, специальная библиотека перечисляет роли текущего пользователя и узнает, имеет ли хотя бы одна из этих ролей соответствующие полномочия.

Между ролями и UNIX-группами можно заметить некоторое сходство, и нет единого мнения насчет того, отличаются ли эти конструкции по сути. На практике роли оказываются более полезными, чем группы, поскольку системы, в которых они реализованы, разрешают использовать их вне контекста файловой системы. Роли могут также иметь иерархические взаимоотношения друг с другом, что значительно упрощает администрирование. Например, вы могли бы определить роль "старшего администратора", который имеет все полномочия "администратора", а также дополнительные полномочия X, Y и Z.

Модель RBAC позволяет на практике реализовать управление большими коллекциями возможных полномочий. Большая часть усилий тратится на иерархическое определение роли, но это делается один раз за проект. Повседневное же администрирование пользователей в этом случае упрощается. Следовательно, системы, которые поддерживают модель RBAC, обычно пользуются ее преимуществами для "расщепления" могущества суперпользователя на множество различных фрагментов, которые могут назначаться пользователям по отдельности.

В системах Solaris для реализации ролей используются группы (`/etc/group`), авторизации (`/etc/security/auth_attr`), профили (`/etc/security/prof_attr`), а также связывания между пользователями, авторизациями и профилями (`/etc/user_attr`). Авторизации имеют такие имена, как `solaris.admin.diskmgr`, `solaris.admin.patchmgr` и `solaris.admin.printer`. Многие авторизации имеют также специальный уровень структуризации `.read` или `.write`. В файле `auth_attr` определено 158 авторизаций. Для управления ролями в Solaris предусмотрены такие команды, как `roleadd`, `rolemod` и `roledel`.

Со времени выхода сборки 99 OpenSolaris в мае 2008 года Solaris-система RBAC вела себя достаточно устойчиво и позволяла работать без учетной записи `root`.

В среде HP-UX для определения "мелкоструктурированных" `root`-привилегий также используются авторизации, которые затем назначаются ролям, связанным с отдельными пользователями и группами. Авторизациям даются имена вроде **`hpux.admin.process.kill`**, **`hpux.admin.network.config`** и **`hpux.admin.device.install`**. В файле **`/etc/rbac/auths`** определено 137 авторизаций. Для управления авторизациями используются команды **`roleadm`**, **`authadm`**, **`cmdprivadm`**, **`privrun`** и **`privedit`**.

В системах AIX используются роли с такими именами: **`DomainAdmin`**, **`BackupRestore`**, **`AccountAdmin`**, **`SysConfig`** и **`SecPolicy`**. Авторизации здесь имеют примерно такой же уровень структуризации, как в Solaris или HP-UX. Вот, например, как звучат в среде AIX имена авторизаций: **`aix.device`**, **`aix.proc`**, **`aix.fs.manage.export`** и **`aix.system.config.cron`**. В AIX-инструменте SMIT ("правая рука" системного администратора) роли связываются с экранами результатов выполнения команд. Пользователи здесь могут "играть" до восьми ролей сразу. Примеры основанных на ролях AIX-команд: **`mkrole`**, **`chrole`**, **`rmrole`**, **`rolelist`** и **`swrole`**.

## SELinux: Linux-системы с улучшенной безопасностью

Операционная система SELinux (как проект) был разработан Агентством национальной безопасности США (АНБ), но с конца 2000 года был передан разработчикам открытого кода. Система SELinux включена в состав ядра Linux (с версии 2.6) и поэтому в настоящее время доступна в большинстве дистрибутивов (но часто в не совсем дееспособном состоянии).

SELinux — это реализация системы принудительного управления доступом доступа (mandatory access control — MAC), в которой все привилегии назначаются администраторами. В среде MAC пользователи не могут делегировать свои права и не могут устанавливать параметры контроля доступа на объекты, которыми они (пользователи) владеют. И поэтому такая операционная система подходит больше для узлов со специальными требованиями[3].

Систему SELinux можно также использовать для контроля доступа на основе ролей, хотя это не главная цель системы. Подробнее см. раздел 22.9.

## Возможности POSIX (Linux)

Системы Linux — даже те, которые не используют SELinux-расширения — теоретически позволяют распределять привилегии учетной записи суперпользователя по "возможностям" в соответствии со стандартом POSIX. Кроме того, спецификации характеристик можно назначить исполняемым программам, а затем программы при выполнении будут запрашивать заданные характеристики. По сути, это — форма выполнения **`setuid`**-команд с нижним уровнем риска.

[3] Вот отзыв одного из технических рецензентов: "Это была не самоцель. В действительности средние узлы, на которых выполняется базовая служба `DNS/web/email`, лучше всего работают именно в среде SELinux. Если же вы решаете какие-то неординарные задачи, все может закончиться очень печально, и вам придется отключить эту систему. В настоящее время SELinux ведет себя гораздо лучше. Но я все же ее отключаю...".

По различным причинам, в том числе из-за проблем, присущих текущей реализации, эта функция оказалась не столь полезной или ценной для системных администраторов, как ожидалось вначале. Подробнее эти возможности рассмотрены в разделе 20.14.

## Подключаемые модули аутентификации

Подключаемые модули аутентификации (Pluggable Authentication Modules — PAM) образуют технологию аутентификации, а не технологию управления доступом. Другими словами, технология PAM призвана искать ответ не на вопрос: "Имеет ли право пользователь X выполнить операцию Y?", а на вопрос: "Как узнать, что это действительно пользователь X?" Технология PAM — это важное звено в цепи управления доступом в большинстве систем.

Раньше пароли пользователей проверялись с помощью содержимого файла `/etc/shadow` (или его сетевого эквивалента) в момент регистрации, чтобы можно было установить соответствующий UID-идентификатор для командной оболочки пользователя или оконной системы. В этом случае программы, выполняемые пользователем, вынуждены были принимать указанный UID на веру. В современном мире сетей, криптографии и устройств биометрической идентификации нужна более гибкая и открытая система. Поэтому и появилась технология PAM.

Технология PAM — это набор разделяемых библиотек, которые позволяют интегрировать различные низкоуровневые методы аутентификации. Администраторы указывают те методы аутентификации, которые (с их точки зрения) должна использовать система в соответствующих контекстах. Программы, которые собираются аутентифицировать пользователя, просто вызывают систему PAM, а не реализуют собственные формы аутентификации. Система PAM, в свою очередь, вызывает специальную библиотеку аутентификации, заданную системным администратором.

Подробнее о системе PAM можно прочитать в разделе 22.4.

## Сетевой протокол криптографической аутентификации Kerberos

Подобно PAM, протокол Kerberos призван решать проблемы аутентификации, а не управления доступом. (Он назван в честь трехголового пса, который защищал вход в царство Аида, — Цербера, или Кербера.) Но если PAM можно назвать структурной оболочкой аутентификации, то Kerberos — это конкретный метод аутентификации.

Реализация Kerberos использует проверенный (сторонний) сервер для выполнения задач аутентификации в масштабах всей сети. Вместо самоаутентификации на своем компьютере вы предоставляете свои мандаты (билеты) Kerberos-службе, а она выдает вам криптографические мандаты, которые вы можете презентовать другим службам как подтверждение вашей идентичности. Больше о протоколе Kerberos написано в разделе 22.10.

## Списки управления доступом

Управление доступом к файловой системе — важнейшая часть систем UNIX и Linux, и поэтому ее совершенствование составляет первоочередную цель разработчиков этих систем. Особое внимание было уделено поддержке списков доступа к файлам и каталогам (*a*ccess *c*ontrol *l*ists — ACL) как обобщению традиционной модели привилегий пользователя/группы/"всех остальных", устанавливаемых сразу для нескольких пользователей и групп. Списки ACL являются частью реализации файловой системы, поэтому

их поддержку в явном виде должна обеспечивать используемая вами файловая система. К счастью, в настоящее время практически все файловые системы UNIX и Linux поддерживают ACL-списки в той или иной форме.

🕮 Подробнее об NFS написано в главе 18.

Поддержка ACL-списков в общем случае реализуется в двух формах: в виде проекта стандарта POSIX, который (хоть и без формального подтверждения) получил широкое признание, и системы, стандартизированной протоколом NFSv4, который базируется на ACL-списках Microsoft Windows. Оба стандарта ACL более детально описаны в разделе 6.6.

# 4.3. Управление доступом в реальном мире

Несмотря на наличие описанных выше превосходных возможностей операционных систем, в большинстве узлов для задач системного администрирования по-прежнему используется учетная запись суперпользователя. Многие жалобы на традиционную систему имеют реальную почву, но и альтернативным вариантам присущи серьезные проблемы. Поэтому особое значение приобретают такие дополнительные программные инструменты, как **sudo** (подробнее чуть ниже), которые в некоторой степени позволяют преодолеть разрыв между критериями простоты использования и безопасности.

Часто для принятия решений в особых обстоятельствах используются возможности POSIX или средства управления доступом на основе ролей (например, когда необходимо разрешить переустановку принтера или демона каждому, кто работает в конкретном отделе), в то время как при решении каждодневных задач ваша административная команда продолжает полагаться на утилиту **sudo** и учетную запись суперпользователя. В некоторых случаях (например, если это оговорено в контракте) для узлов необходимо использовать такие мощные и "ударопрочные" системы, как SELinux.

Поскольку доступ с правами суперпользователя является обязательным условием системного администрирования и центральной точкой для безопасности систем, очень важно правильно пользоваться правами и обязанностями учетной записи `root`.

## Пароль суперпользователя

Если вы используете процедуры и инструменты, описанные в этой главе, вероятно, будете удивлены, как в действительности используется пароль суперпользователя. В большинстве случаев администраторам не нужно знать его вообще.

Тем не менее для учетной записи `root` пароль необходим. Он должен быть секретным и в то же время запоминающимся, чтобы вы могли им воспользоваться (т.е. не забыть) даже через большие интервалы времени. Для "запоминания" паролей можно использовать какой-нибудь хранитель паролей или систему депонирования паролей (см. ниже в этой главе).

🕮 Подробнее о взломе паролей читайте в разделе 22.8.

Самой важной характеристикой хорошего пароля является его длина. Пароль пользователя `root` должен состоять как минимум из восьми символов, так как даже семисимвольные пароли взламываются достаточно легко. В системах, где применяются пароли DES (Data Encryption Standard — стандарт шифрования данных), задавать более длинный пароль не имеет смысла, потому что обрабатываются только первые восемь

символов. В начале главы 7 рассказывается о том, как разрешить использование систем шифрования MD5 или Blowfish, которые позволяют паролям иметь большую длину.

Теоретически наиболее безопасный пароль состоит из случайного набора букв, знаков препинания и цифр. Такой пароль, однако, тяжело запомнить и, как правило, неудобно вводить, поэтому, если системный администратор записывает пароль на бумажке или вводит его слишком медленно, об оптимальном уровне безопасности системы говорить не приходится.

Сегодня наиболее широкое распространение получил подход, заключающийся в выборе пароля по принципу "шокирующего абсурда". Этот принцип был определен Грейди Уордом (Grady Ward) в ранней версии списка часто задаваемых вопросов (FAQ), посвященного выбору идентификационной фразы по системе PGP (Pretty Good Privacy).

*Принцип "шокирующего абсурда" заключается в составлении короткой фразы (или предложения), которая лишена смысла и в то же время вызывает шок у пользователя в данной социальной среде. То есть она должна содержать совершенно неприличные или расистские высказывания либо состоять из абсолютно бессвязных выражений. Применять такой подход не предосудительно, поскольку подразумевается, что пароль никогда не станет известен людям, чьи чувства могут быть им оскорблены*[4].

*Маловероятно, чтобы такой пароль был повторен кем-то еще, ведь он уникален по своей сути. При этом он легко запоминается, потому что имеет яркую эмоциональную окраску. Сдержанный пример «шокирующего абсурда» выглядит так: «Моллюски отгрызли мои гарцующие гениталии». Любой читатель без труда придумает гораздо более шокирующие фразы.*

В системах, которые поддерживают пароли произвольной длины, вы можете использовать в качестве пароля всю абсурдную фразу. Можно также преобразовать эту фразу в пароль, записав только вторые буквы каждого слова или применив какое-нибудь другое легко запоминающееся преобразование. Безопасность пароля значительно возрастет, если включить в него цифры, знаки препинания и прописные буквы.

Если ваш узел включает сотни компьютеров, то нужно ли вам готовить сотни root-паролей? Это зависит от среды и толерантности к риску, но, скорее всего, нет. Практика показывает, что для компьютеров-клонов (например, настольных АРМ) вполне подойдет один и тот же root-пароль. При этом серверам следует подбирать уникальные пароли. Все важные части сети и инфраструктура, обеспечивающая маршрутизацию, должны быть защищены по отдельности.

Потрудитесь точно записать, какие компьютеры совместно используют root-пароль. Важно также иметь структурированный способ изменения root-паролей для таких компьютеров. Небрежность в этом вопросе напрямую связана с большим риском нарушения безопасности и головной болью администраторов.

Пароль суперпользователя следует менять в следующих случаях:

- минимум каждые три месяца;
- всякий раз, когда кто-либо, знающий пароль, увольняется из организации;
- когда, по вашему мнению, безопасности системы что-то угрожает.

Часто говорят, что пароли никогда не следует записывать, однако эту мысль следует выразить точнее: пароли никогда не должны быть доступными для случайных людей.

[4] Данный список FAQ был написан для отдельных пользователей системы PGP. В контексте системного администрирования вам следует учесть потенциальные возможности для нападения. Как бы восприняли ваш вариант "шокирующего абсурда" присяжные, которым, не дай бог, придется рассматривать ваше дело о сексуальных домогательствах?

Пароли суперпользователя и другие важные пароли лучше записывать в зашифрованном виде, чтобы администраторы могли воспользоваться ими при крайней необходимости. (Подробнее чуть ниже.)

## Регистрация под именем root

Поскольку суперпользователь является таким же членом системы, как и остальные пользователи, можно войти в систему непосредственно под именем `root`. Однако оказывается, что это довольно неудачное решение. Во-первых, не будет сделано никаких записей о том, какие операции выполнял суперпользователь. Согласитесь, не очень приятно обнаружить, что вчера ночью в 3:00 вы что-то поменяли, но никак не можете вспомнить, что именно. Еще хуже, если такой доступ был несанкционированным и необходимо выяснить, какой ущерб системе нанес нарушитель. Во-вторых, сценарий регистрации суперпользователя не предполагает сбора дополнительной идентифицирующей информации. Когда под именем `root` в систему могут входить несколько пользователей, не существует способа определить, кто именно и когда это делал.

Вследствие упомянутых причин в большинстве систем регистрация под именем `root` может быть запрещена на терминалах, в оконных системах и по сети, т.е. везде, кроме системной консоли[5]. Мы рекомендуем следовать этой схеме. О том, как реализовать эту политику в конкретной системе, читайте в разделе 22.4.

## Команда su: замена идентификатора пользователя

Лучше получать доступ к учетной записи `root` с помощью команды **su**. Будучи вызванной без аргументов, она выдает приглашение на ввод пароля суперпользователя, а затем запускает интерпретатор команд с правами пользователя `root`. Интерпретатор будет выполняться в привилегированном режиме, пока не завершит работу (при нажатии комбинации клавиш <Ctrl+D> или по команде **exit**). Команда **su** не фиксирует действия, производимые в среде интерпретатора, но добавляет запись в журнальный файл с указанием, кто и когда вошел в систему под именем суперпользователя.

Команда **su** способна также подставлять вместо имени `root` имена других пользователей. Иногда единственный способ помочь пользователю в решении проблемы — войти с помощью команды **su** в его среду и разобраться в происходящем, так сказать, "на месте".

Зная чей-либо пароль, можно непосредственно зарегистрироваться в системе под его именем, введя команду **su** *имя_пользователя*. В ответ будет выдан запрос на ввод пароля. При использовании в качестве ключа команды **su** дефиса (-) интерпретатор перейдет в режим регистрации. Точная реализация этого режима зависит от конкретного интерпретатора команд, но обычно в этом режиме меняются количество и имена файлов запуска, считываемых интерпретатором. Например, в режиме регистрации интерпретатор **bash** считывает файл **~/.bash_profile**, а вне этого режима — файл **~/.bashrc**. При диагностике проблем конкретного пользователя режим регистрации позволяет максимально точно воспроизвести среду, в которой он работал.

В одних системах пароль `root` позволяет регистрироваться с помощью команд **su** или **login** под любым именем. В других нужно сначала стать суперпользователем, вос-

[5] Система Ubuntu Linux пошла еще дальше. По умолчанию система не содержит никакого действующего пароля для учетной записи ***root***, и в ней необходимо использовать команду ***sudo***, подробно описанную в последующих абзацах этого раздела.

пользовавшись в явном виде командой **su**, а затем с помощью этой же команды перейти в другую учетную запись. В таком случае вводить пароль не потребуется.

Рекомендуем взять за правило при вводе команды указывать полное имя, например **/bin/su** или **/usr/bin/su**, а не просто **su**. Это послужит определенной защитой от тех программ с именем **su**, которые преднамеренно были прописаны в переменной среды **path** злоумышленником, намеревавшимся собрать хороший "урожай" паролей[6].

В некоторых системах, чтобы использовать команду **su**, необходимо быть членом группы `wheel`.

Команду **su** в значительной степени может заменить утилита **sudo** (см. следующий раздел), саму же команду **su** лучше всего оставить для крайних (аварийных) случаев.

## Утилита sudo: ограниченный вариант команды su

Без организации управления доступом на основе ролей (RBAC) или такой системы, как SELinux, трудно предоставить кому-то право выполнять конкретную административную операцию (например, создавать резервные копии), не предоставив возможность свободно работать в системе. Если же учетная запись `root` доступна группе администраторов, то вы и понятия не будете иметь о том, кто ею пользуется и что он при этом делает.

Чаще всего для получения прав суперпользователя применяется утилита **sudo**. Она работает во всех рассматриваемых нами системах, а ее исходный код можно получить на веб-сайте `sudo.ws`.


В системе Solaris предусмотрена команда **pfexec**, предоставляющая возможности, подобные возможностям программы **sudo**, которая реализована на базе собственной системы RBAC.

Утилита **sudo** в качестве аргумента принимает командную строку, которая подлежит выполнению от имени `root`-пользователя (или другого уполномоченного пользователя). Утилита обращается к файлу **/etc/sudoers**, где содержится список пользователей, имеющих разрешение на ее выполнение, и перечень команд, которые они могут вводить на конкретном компьютере. Если запрашиваемая команда разрешена, утилита **sudo** приглашает пользователя ввести его собственный пароль и выполняет команду от имени суперпользователя.

Далее утилита **sudo** позволяет, не вводя пароль, выполнять другие команды, но только до тех пор, пока не наступит пятиминутный период бездействия (его продолжительность можно менять). Такая мера служит защитой от тех привилегированных пользователей, которые оставляют свои терминалы без присмотра.

🕮 Подробнее о **syslog** читайте в главе 11.

Утилита **sudo** ведет журнал, где регистрируются выполненные команды, компьютеры, на которых они выполнялись, и вызвавшие их пользователи, а также каталоги, из которых запускались команды, и время их вызова. Эта информация может направляться в системный журнальный файл **syslog** или сохраняться в любом другом файле по усмотрению пользователя. Мы рекомендуем направлять ее на защищенный центральный компьютер.

---

[6] По аналогичной причине настоятельно рекомендуем не включать запись "." (текущий каталог) в переменную **path** интерпретатора команд. Несмотря на очевидное удобство, подобная конфигурация приводит к тому, что можно непреднамеренно запустить "специальную" версию какой-нибудь системной команды, которую злоумышленник оставил в качестве приманки. Пользователю `root` следует быть бдительным вдвойне.

Строка журнального файла, содержащая данные о пользователе **randy**, который выполнил команду **sudo /bin/cat etc/sudoers**, может выглядеть следующим образом.

```
Dec 7 10:57:19 tigger sudo: randy: TTY=ttyp0 ; PWD=/tigger/users/randy;
   USER=root ; COMMAND=/bin/cat /etc/sudoers
```

Существует единая версия файла **/etc/sudoers**, которая используется на всех компьютерах. Сам файл выглядит примерно так.

```
# Define aliases for machines in CS & Physics departments
Host_Alias CS = tigger, anchor, piper, moet, sigi
Host_Alias PHYSICS = eprince, pprince, icarus

# Define collections of commands
Cmnd_Alias DUMP = /sbin/dump, /sbin/restore
Cmnd_Alias PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias SHELLS = /bin/sh, /bin/tcsh, /bin/bash, /bin/ksh, /bin/bsh

# Permissions
mark, ed PHYSICS = ALL
herb CS = /usr/sbin/tcpdump  : PHYSICS = (operator) DUMP
lynda ALL = (ALL) ALL, !SHELLS
%wheel ALL, !PHYSICS = NOPASSWD: PRINTING
```

Первые пять строк (не считая комментариев) определяют набор компьютеров и команд, на которые имеются ссылки в спецификациях прав доступа. Списки элементов можно включать в спецификации в полном виде, но работать с псевдонимами проще, к тому же они делают файл **/etc/sudoers** понятнее и в него легче вносить изменения. Разрешается также создавать псевдонимы для различных групп пользователей.

В каждую спецификацию прав доступа включается информация о следующем:

- пользователях, к которым относится запись;
- компьютерах, на которых пользователям разрешено выполнять соответствующие действия;
- командах, которые могут выполняться указанными пользователями;
- пользователях, от имени которых могут выполняться команды.

Первая строка спецификаций применяется к пользователям `mark` и `ed`, которые регистрируются в системе на компьютерах группы `PHYSICS` (`eprince`, `pprince` и `icarus`). Встроенный псевдоним `ALL` разрешает им вводить любые команды. Поскольку дополнительный список пользователей в скобках не указан, утилита `sudo` будет выполнять команды только от имени суперпользователя.

Пользователь `herb` может запускать утилиту **tcpdump** на компьютерах группы `CS`, а также выполнять команды резервного копирования на компьютерах группы `PHYSICS`. Обратите внимание, однако, что вторая группа команд получит привилегии не пользователя `root`, а пользователя `operator`. Реальная команда, которую пришлось бы ввести пользователю `herb`, выглядит примерно так.

```
ubuntu$ sudo -u operator /usr/sbin/dump 0u /dev/sda1
```

Пользователь `lynda` имеет право выполнять команды от имени любого пользователя на любом компьютере, но он не может запускать большинство распространенных интерпретаторов команд. Означает ли это, что он не может получить доступ к интерпретатору? Конечно, нет.

```
aix$ cp -p /bin/sh /tmp/sh
aix$ sudo /tmp/sh
```

Вообще говоря, попытка разрешить "все команды, кроме..." обречена на провал, по крайней мере, с технической точки зрения. Тем не менее имеет смысл использовать подобную форму спецификации, так как это послужит хотя бы напоминанием о том, что вызывать интерпретатор команд в режиме суперпользователя не рекомендуется.

В последней строке пользователям группы `wheel` разрешается выполнять команды печати **`lpc`** и **`lprm`** от имени суперпользователя на всех компьютерах, за исключением `eprince`, `pprince` и `icarus`. Более того, от пользователей не требуется вводить пароль.

Обратите внимание на то, что команды в файле **`/etc/sudoers`** приводятся с указанием полного имени, чтобы пользователи не могли выполнять свои собственные программы и сценарии от имени суперпользователя. Разрешается также указывать допустимые аргументы команд (в приведенных выше примерах это не показано). В целом возможности файла `sudoers` очень велики, а рассмотренная конфигурация иллюстрирует лишь основные из них.


В системах AIX полезно включать в раздел **`Defaults`** файла **`sudoers`** следующую строку.

```
Defaults env_keep = "ODMDIR"
```

Наличие этой строки не позволит утилите **`sudo`** удалить переменную среды `ODMDIR`, которая указывает многим административным командам на специальную базу объектов ODM (Object Data Manager), в которой хранится значительная часть параметров ОС AIX, например список поддерживаемых и сконфигурированных устройств, информация о драйверах и пр.

Для модификации файла **`/etc/sudoers`** предназначена специальная команда `visudo`, которая проверяет, не редактируется ли файл кем-то посторонним, затем открывает его в редакторе, а перед инсталляцией файла выполняет синтаксический контроль. Последний этап особенно важен, поскольку ошибка в файле **`/etc/sudoers`** может не позволить повторно вызвать утилиту `sudo` для исправления файла.

Использование утилиты **`sudo`** имеет следующие преимущества.

- Благодаря регистрации команд значительно повышается степень административного контроля над системой
- Операторы могут выполнять специальные задачи, не имея неограниченных привилегий
- Настоящий пароль суперпользователя могут знать всего один-два человека[7]
- Вызывать утилиту **`sudo`** быстрее, чем выполнять команду **`su`** или входить в систему под именем **`root`**
- Пользователя можно лишить привилегий, не меняя пароль суперпользователя
- Ведется список всех пользователей с правами пользователя **`root`**
- Меньше вероятность того, что интерпретатор команд, запущенный суперпользователем, будет оставлен без присмотра
- Управлять доступом ко всей сети можно с помощью одного файла

🕮 Дополнительную информацию о взломе паролей можно найти в разделе 22.8.

Утилита **`sudo`** имеет и недостатки. Самый большой из них заключается в том, что любая брешь в системе защиты того или иного привилегированного пользователя экви-

[7] Или даже никто, если вы используете один из проверенных вариантов системы хранения паролей Password Vault.

валентна нарушению безопасности самой учетной записи **root**. Противостоять этому нелегко. Можно лишь предупредить тех, кто имеет право выполнять утилиту **sudo**, о необходимости максимально защищать свои учетные записи. Можно также регулярно запускать какой-нибудь программный взломщик паролей, проверяя "стойкость" паролей привилегированных пользователей.

Другой недостаток — возможность обмануть утилиту **sudo** с помощью таких уловок, как временный выход в интерпретатор команд из разрешенной программы либо выполнение команд **sudo sh** или **sudo su**, если они допустимы.

Во многих версиях систем UNIX утилита **sudo** (как способ разделения привилегий суперпользователя) является лучшим инструментом для встроенных систем управления доступом на основе ролей (RBAC). И вот почему.

- Вы можете точно определить, как именно будут разделены привилегии суперпользователя. Ваш вариант может отличаться в ту или иную сторону от предлагаемого встроенной системой RBAC.
- Простота конфигурации — общепризнанный факт (в основном, это касается установки, обслуживания и понимания происходящего).
- Псевдонимы утилиты **sudo** для групп компьютеров, пользователей и команд функционально подобны ролям в системе RBAC.
- Утилита **sudo** выполняется во всех системах UNIX и Linux. Вам не нужно беспокоиться об использовании различных систем RBAC на разных платформах.
- Вы можете совместно использовать единственный на весь узел файл конфигурации.
- Вы получаете бесплатную возможность высококачественного процесса регистрации в системе.

Основной недостаток **sudo**-ориентированного управления доступом состоит в том, что система остается уязвимой в случае, если будет нарушена защита учетной записи суперпользователя.

## Хранилища паролей и их депонирование

В пятистах милях на север от материковой части Норвегии, на одном из гористых островов архипелага Шпицберген, был вырыт огромный "погреб" — всемирный банк-хранилище посадочного материала всех существующих в мире сельскохозяйственных растений на случай планетарной катастрофы. Системным администраторам не нужно такое большое и холодное (на Шпицбергене вечная мерзлота как никак) спецхранилище для паролей, но какое-то хранилище им все же нужно иметь.

Хранилище паролей может быть реализовано в виде некоторого программного (или программно-аппаратного) комплекса, который сохраняет пароли для конкретной организации более безопасным способом, чем тот, что предлагает Windows (например, с помощью флажка "Запомнить ваш пароль?"). Некоторые разработки сделали хранилище паролей почти необходимостью.

- Пароли теперь нужны не только для входа в системы компьютеров, но и для доступа к определенным веб-страницам, для задания конфигурации сетевых маршрутизаторов и брандмауэров, а также для администрирования удаленных служб
- Возросла необходимость для сильных (т.е. "не очень запоминающихся") паролей, ведь слабые пароли легко взламываются

- Существуют правила, которые требуют доступа к определенным данным, отыскиваемым для одного-единственного пользователя, — без применения таких "общих" регистрационных имен, как `root`

Системы управления паролями возникли после того, как в Сенате США был зарегистрирован законопроект "Акт о кибербезопасности 2009" (Cybersecurity Act of 2009), который предлагает значительно расширить полномочия федеральных властей в сфере безопасности компьютерных сетей. Этот законопроект (в случае его принятия) может существенно повлиять на структуру и саму суть современного Интернета, поскольку авторы законопроекта предлагают решать вопросы идентифицируемости и безопасности на государственном уровне с учетом особенностей различных производственных секторов. В некоторых случаях этот законопроект требует двухфакторной аутентификации, например пароль или парольная фраза плюс обмен сообщениями "вызов/ответ" (для проверки правильности реакции пользователя на непредсказуемый запрос системы). Хранилища паролей — это большое подспорье для компаний, которые должны безопасно и прозрачно управлять паролями не только собственных компьютеров, но и компьютеров своих клиентов.

В настоящее время доступно несколько программ для хранения паролей. Среди них программа KeePass — бесплатный, легкий и удобный в работе менеджер паролей с открытым исходным кодом, предназначенный для отдельных пользователей. Программа KeePass хранит пароли локально, предоставляет категорический (все или ничего) доступ к базе данных паролей, не выполняя при этом входа в систему. Продукты, предназначенные для крупных корпораций (например, от компании Cyber-Ark), могут стоить десятки тысяч долларов. Многие коммерческие предложения ориентированы либо на удовлетворение специальных требований пользователя, либо просто на количество запоминаемых паролей.

Мы используем собственную веб-ориентированную систему, которая имеет несколько ценных функций. Одна их них (наша любимая) называется "разбить стекло" (по аналогии с надписью на противопожарном оборудовании в отелях: "в случае крайней необходимости разбить стекло и дернуть за большую красную ручку").

В нашем случае "разбить стекло" означает получить пароль, к которому обычно нет доступа. В случае аварии вы можете взять на себя ответственность и запросить пароль. Система предоставит список системных администраторов и зарегистрирует все действия, совершаемые вами с помощью "аварийного" пароля. "Разрулив" ситуацию, вы должны изменить пароль и поместить в хранилище новый пароль.

"Низкотехнологичный" способ реализации депонирования паролей "позаимствован" у полиции, которая для хранения вещественных доказательств, собранных на месте преступления, использует специальные пронумерованные пакеты. Такие пакеты можно найти с помощью Интернета. До тех пор пока пакет не открыт, вы точно знаете, что никто не получил доступа к паролю, хранящемуся внутри него.

## 4.4. О псевдопользователях

Только пользователь `root` имеет для ядра Linux особый статус. Есть, однако, еще несколько псевдопользовательских учетных записей, которые применяются для системных целей. Эти фиктивные учетные записи можно идентифицировать по значениям UID, которые обычно меньше 100. Как правило, учетные записи с UID меньше 10 принадлежат системе, а значения UID от 10 до 100 отведены для псевдопользователей, связанных со специальными программами.

Пароли этих псевдопользователей в файле **/etc/shadow** обычно заменяют звездочкой, чтобы нельзя было войти в систему под служебным именем. Чтобы защититься от средств атаки на основе дистанционного входа в систему (когда вместо паролей используются файлы SSH-ключей), укажите в качестве командных интерпретаторов (вместо **/bin/bash** или **/bin/sh**) вариант **/bin/false** или **/bin/nologin**.

Файлы и процессы, которые являются частью операционной системы, но не должны принадлежать пользователю root, иногда передаются во владение пользователям bin или daemon. Считается, что это поможет избежать риска, связанного с действиями от имени суперпользователя. Однако ввиду неубедительности подобной аргументации в современных системах используется просто учетная запись root.

В некоторых системах пользователь sys владеет специальными файлами (образами памяти ядра), которые хранятся в каталоге **/dev**. Доступ к этим файлам имеют лишь немногие программы, и все они изменяют текущий идентификатор пользователя на sys. Иногда вместо пользователя sys создается группа kmem или sys.

🕮 Дополнительную информацию об учетной записи nobody можно найти в конце раздела 18.2.

Сетевая файловая система — NFS (Network File System) — использует учетную запись nobody для представления суперпользователей в других системах. Чтобы лишить суперпользователей их исключительных прав, излишних в данном контексте, NFS должна на время сеанса удаленного доступа заменить нулевой идентификатор чем-то другим. Этой цели как раз и служит учетная запись nobody. В среде сервера NFSv4 эта учетная запись также применяется для удаленных пользователей, которые не имеют допустимые локальные учетные записи.

Пользователю nobody не нужны специальные права доступа, и он не должен владеть никакими файлами. Если бы ему принадлежали какие-то файлы, то над ними получили бы контроль суперпользователи, регистрирующиеся в удаленных системах.

Традиционно пользователю nobody соответствовал идентификатор -1 или -2, и это соглашение по-прежнему соблюдается в ядре Linux, где идентификатор этого пользователя равен 65534 (16-битовый беззнаковый эквивалент -2). В других дистрибутивах просто выбирается маленькое значение идентификатора (например, в Red Hat используется значение, равное 99), что при использовании 32-битовых идентификаторов пользователей более оправдано. В системах Solaris применяется значение идентификатора 60001, которое, по крайней мере, легко запомнить.

Единственный недостаток переопределения nobody-идентификатора состоит в том, что, судя по всему, команда **exportfs** не обращает внимания на файл **passwd**, поэтому с помощью опции **anonuid** ей нужно явно указывать о необходимости использования другого идентификатора для пользователя nobody.

# 4.5. Упражнения

4.1. Воспользуйтесь командой **find** с флагом **-perm**, чтобы найти в своей системе пять файлов с установленным битом "setuid". Объясните, почему каждая из команд не сможет правильно функционировать без этого бита.

4.2. Создайте две записи в файле конфигурации **sudoers**.

а) первая запись разрешает пользователям matt и lisa конфигурировать принтер, очищать его очередь и перезапускать демоны печати на компьютере printserver

б) вторая запись позволяет пользователям `drew`, `smithgr` и `jimlane` удалять задания и перезагружать компьютеры в студенческой лаборатории

4.3. Составьте три фразы по принципу "шокирующего абсурда", а затем пропустите их через утилиту **`md5sum`** и сравните результаты. Не безопаснее ли использовать пароли в формате MD5 (учитывая достижения технологии шифрования)? Почему?

4.4. ★ Опишите последовательность команд, позволяющих модифицировать чужой пароль, и покажите, как "замести следы". Предполагается, что доступ к учетной записи `root` имеется только через утилиту **`sudo`** (разрешены все команды, кроме интерпретаторов и **`su`**).

4.5. ★ Инсталлируйте утилиту **`sudo`**, которая будет направлять вам по электронной почте сообщения о попытках несанкционированного доступа. Используйте ее для проверки записей, созданных в упражнении 4.2, и убедитесь в том, что утилита правильно регистрирует сообщения в Syslog. (Необходим доступ с правами суперпользователя; возможно, понадобится также модифицировать файл **`/etc/syslog.conf`**.)

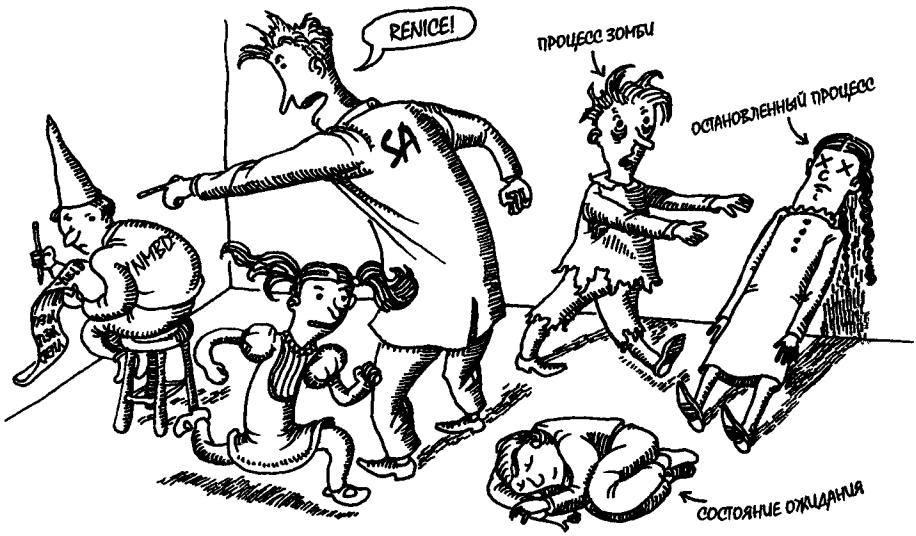
4.6. ★ В системе Solaris, HP-UX или AIX создайте RBAC-роль, которая бы позволяла монтировать и демонтировать файловые системы. Назначьте на эту роль двух пользователей. (Необходим доступ с правами суперпользователя.)

а) какие действия необходимо выполнить для этого? Можете ли вы ограничить список разрешенных операций для определенных файловых систем или типов файловых систем?

б) переопределите свое решение в виде `sudo`-конфигурации. Оно сложнее, чем RBAC-решение, или проще? Можете ли вы ограничить список разрешенных операций для определенных файловых систем или типов файловых систем?

# Глава 5

# Управление процессами




Процесс — это абстракция, используемая в операционных системах UNIX и Linux для описания выполняющейся программы. Процесс представляет собой системный объект, посредством которого можно контролировать обращения программы к памяти, центральному процессору и ресурсам ввода-вывода.

Концепция, согласно которой как можно больше работы должно выполняться в контексте процессов, а не в ядре, является частью философии UNIX. Системные и пользовательские процессы подчиняются одним и тем же правилам, благодаря чему управление ими осуществляется с помощью единого набора команд.

## 5.1. Атрибуты процесса

Процесс состоит из адресного пространства и набора структур данных, содержащихся внутри ядра. Адресное пространство представляет собой совокупность страниц памяти[1], которые были выделены ядром для выполнения процесса. В него загружается код процесса и используемые им библиотеки функций, а также переменные, содержимое стеков и различная вспомогательная информация, необходимая ядру во время работы процесса. Поскольку в системах UNIX и Linux поддерживается концепция виртуальной памяти, страницы адресного пространства процесса в конкретный момент времени могут находиться либо в физической памяти, либо в разделе подкачки, т.е. на диске.

В структурах данных ядра хранится всевозможная информация о каждом процессе. К наиболее важным относятся следующие сведения:

- таблица распределения памяти;
- текущий статус (неактивен, приостановлен, выполняется и т.п.);

[1] Страницы — это базовые блоки памяти, размер которых составляет от 1 до 8 Кбайт.

- приоритет;
- информация об используемых ресурсах;
- информация о файлах и сетевых портах, открытых процессом;
- маска сигналов (запись о том, какие сигналы блокируются);
- имя владельца процесса.

Поток выполнения, обычно именуемый просто *потоком*, представляет собой результат разветвления в выполнении процесса. Поток наследует многие атрибуты своего процесса (например, адресное пространство процесса), причем в рамках одного процесса могут выполняться одновременно (параллельно) сразу несколько потоков — такая модель выполнения получила название *многопоточности* (multithreading).

В старых однопроцессорных системах параллельное выполнение моделируется (точнее, имитируется) ядром, но в мультиядерных и многопроцессорных архитектурах потоки могут выполняться одновременно в различных ядрах. Такие многопоточные приложения, как BIND и Apache, извлекают максимальную пользу из мультиядерных систем, поскольку эти приложения могут отрабатывать несколько запросов одновременно. Все наши примеры операционных систем поддерживают многопоточный режим.

Многие характеристики процесса непосредственно влияют на его выполнение. В частности, имеет значение, сколько времени выделяется ему центральным процессором, к каким файлам он имеет доступ и т.д. В следующих подразделах рассмотрим наиболее интересные с точки зрения системного администратора характеристики процессов. Они поддерживаются во всех версиях систем UNIX и Linux.

## Идентификатор процесса (PID)

Ядро назначает каждому процессу уникальный идентификатор[2]. Большинство команд и системных вызовов, работающих с процессами, требует указания конкретного идентификатора, чтобы был ясен контекст операции. Идентификаторы присваиваются по порядку по мере создания процессов.

## Идентификатор родительского процесса (PPID)

Ни в UNIX, ни в Linux нет системного вызова, который бы инициировал новый процесс для выполнения конкретной программы. Для того чтобы породить новый процесс, существующий процесс должен клонировать сам себя. Клон может заменить выполняемую программу другой.

В операции клонирования исходный процесс называют родительским, а его клон — дочерним. Помимо собственного идентификатора, каждый дочерний процесс имеет атрибут PPID (Parent Process ID), который совпадает с идентификатором породившего его родительского процесса[3].

Идентификатор родительского процесса — весьма полезная информация, если приходится иметь дело с неизвестным (и, возможно, нестандартно ведущим себя) процес-

[2] Как отмечено нашим рецензентом Джоном Корбетом (Jon Corbet), в ядре Linux 2.6.24 представлены пространства имен процессов, которые позволяют существовать одновременно нескольким процессам с одинаковыми идентификаторами PID. Это свойство реализовано для поддержки виртуализации на основе контейнеров.

[3] По крайней мере, первоначально. Если родительский процесс по какой-то причине завершается раньше потомка, демон `init` (процесс с номером 1) подставляет себя на место предка (подробности описаны в разделе 5.2).

сом. Отслеживание истоков процесса может облегчить понимание его назначения и значимости.

## Идентификатор пользователя (UID) и текущий идентификатор пользователя (EUID)

Дополнительная информация об идентификаторах пользователя приведена в разделе 7.1.

UID (User ID) — это идентификатор пользователя, создавшего данный процесс, точнее, копия значения UID родительского процесса. Менять атрибуты процесса могут только его создатель (владелец) и суперпользователь.

EUID (Effective User ID) — это текущий пользовательский идентификатор процесса, предназначенный для того, чтобы определить, к каким ресурсам и файлам у процесса есть право доступа в данный момент. У большинства процессов значения UID и EUID одинаковы. Исключение составляют программы, у которых установлен бит смены идентификатора пользователя (`setuid`).

Зачем нужны два идентификатора? Просто имеет смысл разграничить понятия персонификации и прав доступа. К тому же программы с установленным битом `setuid` не всегда выполняются с расширенными привилегиями. В большинстве систем значение EUID можно устанавливать, чтобы предоставлять процессу дополнительные полномочия, и сбрасывать, чтобы отбирать их.

В большинстве систем хранится начальный идентификатор, т.е. копия значения EUID, который имел процесс в начальной точке. Если процесс не удалит сохраненный идентификатор, его можно будет использовать в качестве реального или текущего идентификатора. Благодаря этому "консервативно" написанная программа с установленным битом `setuid` может большую часть времени работать с обычными привилегиями, переходя в режим расширенных привилегий лишь в определенные моменты.

В системе Linux есть еще и нестандартный параметр FSUID, определяющий возможности работы с файловой системой, но вне ядра он используется редко и не переносится на другие системы UNIX.

## Идентификатор группы (GID) и текущий идентификатор группы (EGID)

Дополнительная информация о группах приведена в разделе 7.1.

GID (Group ID) — это идентификатор группы, к которой принадлежит владелец процесса. Текущий идентификатор группы (Effective Group ID — EGID) связан с атрибутом GID так же, как значение EUID связано с UID, т.е. они будут отличаться в случае программы, у которой установлен бит смены идентификатора группы (`setgid`). В ядре назначение сохраненного GID аналогично назначению сохраненного атрибута UID.

В значительной степени атрибут GID процесса является рудиментарным. С точки зрения определения прав доступа процесс одновременно может относиться к нескольким группам. Список групп хранится отдельно от значений GID и EGID. При анализе прав доступа проверяется текущий идентификатор и дополнительный список групп, но не значение GID. Если пользователь пытается получить доступ к какому-либо ресурсу, весь список проверяется на предмет того, принадлежит ли пользователь к группе, членам которой разрешается использовать данный ресурс.

Атрибут GID используется только при создании процессом новых файлов. В зависимости от установленных прав доступа в данной файловой системе новые файлы могут принимать атрибут GID создающего их процесса. Подробнее эти вопросы рассмотрены в разделе 6.5.

## Приоритет и фактор уступчивости

Приоритет процесса определяет, какую долю времени центрального процессора получает программа. Ядро применяет динамический алгоритм вычисления приоритетов, учитывающий, сколько времени центрального процессора уже использовал процесс и сколько времени он ожидает своей очереди. Кроме того, учитывается заданный администратором так называемый *фактор уступчивости* (устанавливается с помощью команды **nice**), который определяет, в какой степени программа может "делиться" процессором с другими программами. Подробнее этот механизм рассматривается в разделе 5.6.

Для улучшения поддержки приложений с небольшой временной задержкой в системе Linux в традиционную модель планирования задач UNIX добавлены "классы планирования". В настоящее время существует три класса планирования, причем каждый процесс относится только к одному из них. К сожалению, классы реального времени пока не находят широкого применения, а их поддержка из командной строки оставляет желать лучшего. Все системные процессы используют традиционный планировщик, ориентированный на применение фактора уступчивости, который мы и рассматриваем в этой книге. Более подробную информацию по вопросам, связанным с планированием в реальном времени, можно найти на веб-сайте `www.realtimelinuxfoundation.org`.

## Управляющий терминал

С большинством процессов, не являющихся демонами, связан управляющий терминал, который определяет базовую конфигурацию стандартных каналов ввода, вывода и ошибок. Когда пользователь запускает какую-либо программу в интерпретаторе, его терминал, как правило, становится управляющим терминалом процесса. От управляющего терминала зависит также распределение сигналов, о чем пойдет речь в разделе 5.3.

# 5.2. Жизненный цикл процесса

Для создания нового процесса существующий процесс, как правило, клонирует сам себя с помощью системного вызова **fork**. В результате формируется копия исходного процесса, имеющая лишь некоторые отличия. В частности, новому процессу присваивается собственный идентификатор, и учет ресурсов ведется для него независимо от предка.

Системный вызов **fork** имеет уникальное свойство: он возвращает два разных значения. В дочернем процессе это будет 0, а в родительском — идентификатор процесса-потомка. Поскольку в остальном процессы идентичны, они должны проверять результат вызова, чтобы определить, какую роль следует играть в дальнейшем.

После завершения системного вызова **fork** дочерний процесс обычно запускает новую программу с помощью одного из системных вызовов семейства **exec**[4]. Все вызовы этого семейства выполняют приблизительно одинаковые действия: они замещают сег-

[4] В действительности все они, кроме одного, являются библиотечными функциями.

мент кода процесса и устанавливают сегменты памяти в исходное состояние. Формы вызовов **exec** различаются только способами указания аргументов командной строки и переменных среды, передаваемых новой программе.

🕮 Подробная информация о начальной загрузке и демоне **init** приведена в главе 3.

Когда система загружается, ядро самостоятельно запускает несколько процессов. Наиболее важный из них — это демон **init**, идентификатор которого всегда равен 1. Демон **init** отвечает за выполнение сценариев запуска системы, хотя характер его действий неодинаков в UNIX и Linux. Все процессы, кроме тех, что создаются ядром, являются потомками демона **init**.

Демон **init** играет и другую важную роль в управлении процессами. Завершающийся процесс вызывает функцию **_exit**, чтобы уведомить ядро о своей готовности прекратить работу. В качестве параметра функции **_exit** передается код завершения — целое число, обозначающее причину останова процесса. Нулевой код свидетельствует об успешном завершении процесса.

Ядро системы требует, чтобы, прежде чем процесс окончательно исчезнет, его удаление было подтверждено родительским процессом с помощью системного вызова **wait**. Этот вызов возвращает код завершения потомка (или сообщает о причине его уничтожения, если завершение не было естественным) и, в случае необходимости, статистику использования ресурсов.

Описанный механизм работает нормально, если родительский процесс завершается позже порожденных им процессов и выполняет системные вызовы **wait** для их уничтожения. Если же родительский процесс завершается раньше срока, то ядро "понимает", что вызова **wait** не последует, и переназначает все "осиротевшие" процессы демону **init**. Он берет их под свой контроль, осуществляя для каждого из них вызов **wait**.

## 5.3. Сигналы

Сигналы — это запросы на прерывание, реализуемые на уровне процессов. Определено свыше тридцати различных сигналов, и они находят самое разное применение.

- Сигналы могут посылаться между процессами как средство коммуникации.
- Сигналы могут посылаться драйвером терминала для уничтожения или приостановки процессов, когда пользователь нажимает специальные комбинации клавиш, такие как <Ctrl+C> или <Ctrl+Z>[5].
- Сигналы могут посылаться в самых разных целях пользователем или администратором с помощью команды **kill**.
- Сигналы могут посылаться ядром, когда процесс выполняет нелегальную инструкцию, например деление на нуль.
- Сигналы могут посылаться ядром для уведомления процесса о "представляющем интерес" событии, таком как прекращение дочернего процесса или доступность данных в канале ввода-вывода.

🕮 Дамп памяти — это файл, содержащий образ памяти процесса. Его можно использовать для отладки.

[5] Функции, связанные с этими комбинациями клавиш, могут назначаться другим клавишам с помощью команды **stty**, но на практике такое встречается очень редко. В этой главе мы подразумеваем, что с данными клавишами связаны их стандартные функции.

Когда поступает сигнал, возможен один из двух вариантов развития событий. Если процесс назначил сигналу подпрограмму обработки, то после вызова ей предоставляется информация о контексте, в котором был сгенерирован сигнал. В противном случае ядро выполняет от имени процесса действия, заданные по умолчанию. Эти действия зависят от сигнала. Многие сигналы приводят к завершению процесса, а в некоторых случаях при этом еще и создается дамп памяти.

Процедура вызова обработчика называется *перехватом сигнала*. Когда выполнение обработчика завершается, процесс возобновляется с той точки, где был получен сигнал.

Для того чтобы определенные сигналы не поступали в программу, нужно задать их игнорирование или блокирование. Игнорируемый сигнал просто пропускается и не влияет на работу процесса. Блокируемый сигнал ставится в очередь на обработку, но ядро не требует от процесса никаких действий до явного разблокирования сигнала. Обработчик вызывается для разблокированного сигнала только один раз, даже если в течение периода блокировки поступило несколько аналогичных сигналов.

В табл. 5.1 перечислены сигналы, которые должны быть известны любому системному администратору. Традиционно имена сигналов записываются прописными буквами. Иногда к именам добавляется префикс `SIG` (например, `SIGHUP`).

**Таблица 5.1. Сигналы, которые должен знать каждый администратор**[a]

| № | Имя | Описание | Реакция по умолчанию | Перехватывается? | Блокируется? | Дамп памяти? |
|---|---|---|---|---|---|---|
| 1 | HUP | Отбой | Завершение | Да | Да | Нет |
| 2 | INT | Прерывание | Завершение | Да | Да | Нет |
| 3 | QUIT | Выход | Завершение | Да | Да | Да |
| 9 | KILL | Уничтожение | Завершение | Нет | Нет | Нет |
| -[б] | BUS | Ошибка на шине | Завершение | Да | Да | Да |
| 11 | SEGV | Ошибка сегментации | Завершение | Да | Да | Да |
| 15 | TERM | Запрос на завершение | Завершение | Да | Да | Нет |
| -[б] | STOP | Останов | Останов | Нет | Нет | Нет |
| -[б] | TSTP | Сигнал останова, посылаемый с клавиатуры | Останов | Да | Да | Нет |
| -[б] | CONT | Продолжение после останова | Игнорируется | Да | Нет | Нет |
| -[б] | WINCH | Изменение окна | Игнорируется | Да | Да | Нет |
| -[б] | USR1 | Определяется пользователем | Завершение | Да | Да | Нет |
| -[б] | USR2 | Определяется пользователем | Завершение | Да | Да | Нет |

[a] Список названий сигналов и номеров также можно получить с помощью встроенной в **bash** команды **kill -l**.

[б] Зависит от используемой системы. Подробнее см. файл **/usr/include/signal.h** или man-страницы интерактивного руководства для системного вызова `signal()`.

Существуют и другие сигналы, не показанные в табл. 5.1; большинство из них сообщает о всяких загадочных ошибках, например "неверная инструкция". По умолчанию такие сигналы, как правило, приводят к завершению программы и созданию дампа памяти. Перехват и блокирование сигналов обычно разрешены, так как есть достаточно "умные" программы, устраняющие последствия ошибок.

Сигналы BUS и SEGV также посылаются при возникновении ошибок. Мы включили их в таблицу, поскольку они чрезвычайно распространены: обычно программа аварийно завершается именно из-за них. Сами по себе эти сигналы не имеют диагностической ценности. Они лишь указывают на факт неправильного обращения к памяти[6].

Сигналы KILL и STOP нельзя ни перехватить, ни заблокировать, ни проигнорировать. Сигнал KILL приводит к уничтожению процесса, которому он посылается, а сигнал STOP приостанавливает выполнение процесса до получения сигнала CONT. Сигнал CONT можно перехватить и проигнорировать, но нельзя заблокировать.

Сигнал TSTP представляет собой более "гибкую" версию сигнала STOP. Проще всего описать его как запрос на останов. Он генерируется драйвером терминала при нажатии пользователем комбинации клавиш <Ctrl+Z>. Программы, перехватывающие этот сигнал, обычно выполняют операции очистки, а затем посылают сами себе сигнал STOP. С другой стороны, программы могут игнорировать сигнал TSTP, чтобы их нельзя было остановить командой с клавиатуры.

Эмуляторы терминалов посылают сигнал WINCH, когда происходит изменение их конфигурационных параметров (например, числа строк на виртуальном терминале). Это позволяет программам, которые взаимодействуют с эмуляторами, таким как текстовые редакторы, автоматически переконфигурировать себя в ответ на изменения. Если размер окна не удается правильно изменить, убедитесь в том, что сигнал WINCH генерируется и корректно обрабатывается[7].

Хотя назначение сигналов KILL, INT, TERM, HUP и QUIT может показаться одинаковым, в действительности они совершенно различны.

- Сигнал KILL не блокируется и приводит к безусловному завершению процесса на уровне ядра. По сути, процесс не успевает даже принять этот сигнал.
- Сигнал INT посылается драйвером терминала при нажатии пользователем комбинации клавиш <Ctrl+C> и служит запросом на завершение текущей операции. Перехватив этот сигнал, простые программы должны завершить работу или позволить уничтожить себя стандартному обработчику сигнала. Программы, в которых есть интерактивный режим командной строки, должны прекратить текущую операцию, выполнить очистку и снова перейти в режим ожидания.
- Сигнал TERM представляет собой запрос на завершение программы. Предполагается, что процесс, получивший этот сигнал, осуществляет очистку и завершается.
- У сигнала HUP есть две распространенные интерпретации. Во-первых, многие демоны воспринимают его как команду сброса. Если демон способен повторно прочесть свой конфигурационный файл и адаптироваться к изменениям без перезапуска, сигнал HUP позволяет менять его поведение.

---

[6] Точнее говоря, ошибки на шине возникают в результате нарушения различных требований синхронизации или использования бессмысленных адресов. Нарушения правил сегментирования представляют собой нарушения правил защиты, например к ним относятся попытки выполнить запись в части адресного пространства, доступной только для чтения.

[7] На практике это может оказаться не такой уж простой задачей. И эмулятор терминала (например, **xterm**), и драйвер терминала, и команды пользовательского уровня — все они могут вносить свой вклад в распространение сигнала *SIGWINCH*. Распространенные проблемы — отправка сигнала только фоновому процессу терминала (а не всем связанным с ним процессам) без рассылки по сети уведомления об изменении размера удаленным компьютерам. Такие протоколы, как *TELNET* и *SSH*, явным образом распознают изменения размера локального терминала и передают эту информацию удаленному узлу. Более простые протоколы (например, протоколы прямого последовательного подключения) лишены этой возможности.

- Во-вторых, этот сигнал иногда генерируется драйвером терминала при попытке уничтожить связанные с терминалом процессы. В основном это поведение сохранилось со времен использования проводных соединений терминалов и модемов. Отсюда и название "отбой".
- Интерпретаторы семейства **csh** (**tcsh** и другие) обычно делают фоновые процессы невосприимчивыми к сигналу HUP, чтобы они могли продолжать свою работу, даже когда пользователь выходит из системы. Пользователи интерпретаторов семейства **sh** (**ksh**, **bash** и так далее) могут эмулировать такое поведение с помощью команды **nohup**.
- Сигнал QUIT напоминает сигнал TERM, за исключением того, что по умолчанию стандартный обработчик создает дамп памяти.

Сигналы USR1 и USR2 не имеют стандартного назначения. Ими можно пользоваться в различных целях. Например, веб-сервер Apache интерпретирует сигнал USR1 как запрос на перезапуск.

## 5.4. Отправка сигналов: команда kill

Команду **kill** чаще всего используют для уничтожения процессов. Эта команда может послать процессу любой сигнал, но по умолчанию это сигнал TERM. Команду **kill** могут выполнять как рядовые пользователи (для своих собственных процессов), так и суперпользователь (для любого процесса). Она имеет следующий синтаксис:

**kill** [*-сигнал*] *идентификатор*,

где *сигнал* — это номер или символическое имя посылаемого сигнала (см. табл. 5.1), а *идентификатор* — номер искомого процесса.

Команда **kill** без номера сигнала не гарантирует, что процесс будет уничтожен, поскольку сигнал TERM можно перехватывать, блокировать и игнорировать. Команда

**kill** -9 *идентификатор*

"гарантированно" уничтожает процесс, так как сигнал с номером 9 (KILL) не перехватывается. Используйте команду **kill** -9 только в случае, если "вежливый" запрос на завершение программы не был выполнен. Мы написали слово "гарантированно" в кавычках, так как иногда процессы переходят в такое состояние, в котором их нельзя завершить даже таким способом (обычно это связано с блокировкой ввода-вывода, например, при остановке жесткого диска). Единственный выход в такой ситуации — перезагрузка.

Команда **killall** выполняет различные функции в системах UNIX и Linux. В Linux команда **killall** уничтожает процессы, заданные именем. Например, следующая команда уничтожает все процессы веб-сервера Apache.

```
ubuntu$ sudo killall httpd
```

Стандартная UNIX-команда **killall**, предусмотренная в дистрибутивах Solaris, HP-UX и AIX, не принимает параметры и просто уничтожает все процессы текущего пользователя. Ее выполнение от имени суперпользователя приведет к уничтожению процесса init и выключению компьютера. Ух!

Команды **pgrep** и **pkill** в системах Solaris, HP-UX и Linux (но не в AIX) осуществляют поиск процессов, заданных именами (или другими атрибутами). Команда **pgrep** выводит идентификаторы процессов, удовлетворяющих заданному в командной строке критерию, а команда **pkill** посылает найденным процессам сигнал. Например, следую-

щая команда посылает сигнал TERM всем процессам, выполняемым от имени пользователя ben.

```
$ sudo pkill -u ben
```

# 5.5. Состояния процесса

Факт существования процесса не дает ему автоматически права на получение доступа к центральному процессору. Необходимо помнить о четырех состояниях выполнения процесса (табл. 5.2).

**Таблица 5.2. Состояния процесса**

| Состояние | Описание |
|---|---|
| Выполнение | Процесс можно выполнять |
| Ожидание | Процесс ждет выделения ресурса |
| Зомби | Процесс пытается завершиться |
| Останов | Процесс приостановлен (не имеет разрешения на выполнение) |

Выполняемый процесс получил все необходимые ресурсы и ждет, пока системный планировщик предоставит ему доступ к центральному процессору. Если процесс осуществляет системный вызов, который нельзя завершить немедленно (например, чтение части файла), ядро переводит его в состояние ожидания.

Ожидающий процесс ждет наступления определенного события. Интерактивные интерпретаторы команд и системные демоны проводят в этом состоянии большую часть времени, ожидая поступления данных с терминала или из сетевого соединения. Поскольку ожидающий процесс фактически блокирован, доступ к центральному процессору будет предоставлен ему только в случае получения сигнала или ответа на один из его запросов ввода-вывода.

Некоторые операции переводят процесс в состояние непрерывного ожидания. Обычно это состояние является временным и не отображается в выводе команды **ps**, о чем свидетельствует флаг D в столбце STAT (см. табл. 5.4). Однако некоторые нестандартные ситуации могут приводить к сохранению этого состояния надолго. Наиболее распространенная причина возникновения такой ситуации — наличие проблем сервера в файловой системе NFS, смонтированной с применением параметра "hard". Поскольку процессы в состоянии непрерывного ожидания не могут быть активизированы даже для реакции на сигнал, их нельзя уничтожить. Для того чтобы избавиться от них, необходимо устранить породившую их проблему или перезагрузить систему.

Зомби — это процесс, который закончил выполняться, но информация об этом еще не поступила родительскому процессу. При обнаружении зомби с помощью команды **ps** проверьте их атрибуты PPID, чтобы выяснить происхождение этих процессов.

Остановленному процессу временно запрещено выполняться. Процессы останавливаются при получении сигнала STOP или TSTP и возобновляют работу по сигналу CONT. Это состояние аналогично ожиданию, но выход из него возможен только с помощью другого процесса.

## 5.6. Изменение приоритета выполнения: команды nice и renice

Фактор "уступчивости" — это число, по которому ядро определяет свою политику в отношении процессов, конкурирующих за право доступа к центральному процессору. Чем выше фактор уступчивости, тем ниже приоритет процесса и наоборот, отсюда и название термина. Низкое или отрицательное значение означает использование высокого приоритета: процесс ведет себя не слишком уступчиво.

Диапазон допустимых значений фактора уступчивости зависит от используемой системы, и обычно он лежит в пределах от –20 до +19. В некоторых системах используется диапазон такого же размера, но со смещением в область неотрицательных чисел (как правило, от 0 до 39). Диапазоны допустимых значений фактора уступчивости, используемые в наших примерах систем, приведены в табл. 5.3.

Несмотря на числовые различия, все системы обрабатывают значения фактора уступчивости практически одинаково. Если пользователь не предпринимает специальных мер, дочерний процесс наследует приоритет своего родительского процесса. Владелец процесса может увеличить фактор уступчивости, но не может уменьшить его, даже чтобы вернуться к стандартному значению. Это не позволяет процессам с низким приоритетом порождать высокоприоритетных потомков. Суперпользователь может устанавливать произвольные значения фактора уступчивости.

В настоящее время администраторам редко приходится определять приоритеты вручную. Когда операционные системы работали на маломощных компьютерах 70–80 гг., на производительность больше всего влияло то, какой процесс занимал основную часть времени центрального процессора. Сегодня, когда на рабочих столах стоят намного более быстродействующие компьютеры, системный планировщик, как правило, обслуживает все процессы весьма оперативно. Добавление классов планирования предоставляет разработчикам дополнительные средства управления в тех случаях, когда важна быстрая ответная реакция.

К сожалению, уровень производительности подсистемы ввода-вывода растет не так стремительно, как быстродействие центральных процессоров, поэтому жесткие диски стали основным узким местом в большинстве операционных систем. Фактор уступчивости никак не влияет на подсистемы управления памятью и вводом-выводом, поэтому даже низкоприоритетный процесс способен монополизировать эти ресурсы или захватить непропорционально большую их часть.

Фактор уступчивости можно установить при создании процесса. Это делается с помощью команды **nice**. Команда **renice** позволяет изменять приоритет выполняемого процесса. Первая из этих команд принимает в качестве аргумента строку запуска процесса, а вторая — идентификатор процесса либо имя пользователя.

Приведем примеры.

```
$ nice -n 5 ~/bin/longtask      // Понижаем приоритет (увеличиваем
                                // фактор уступчивости) на 5
$ sudo renice -5 8829           // Задаем фактор уступчивости равным -5
$ sudo renice 5 -u boggs        // Задаем фактор уступчивости процессов
                                // пользователя "boggs" равным 5
```

К сожалению, в системах по-разному реализован способ установки желаемого приоритета; более того, даже в рамках одной и той же системы не согласованы механизмы действия команд **nice** и **renice**. Одни команды требуют указания *относительного* значения фактора уступчивости, а другие — *абсолютного*. В одних командах нужно перед

значением фактора уступчивости ставить дефис, в других требуется флаг **-n**, а третьи "довольствуются" просто числовым значением.

Существует также версия команды **nice**, встроенная в интерпретатор **csh** и ряд других популярных интерпретаторов (но не в **bash**). Если не указать полное имя команды, будет вызвана именно встроенная версия, а не системная. Это многих сбивает с толку, так как синтаксис команд различен: встроенная версия требует, чтобы изменение приоритета записывалось в формате *+инкр* или *-декр*, а системная версия ожидает флаг **-n**, за которым следует значение приращения[8].

Все эти вариации и разбросы приведены в табл. 5.3. Элемент *приор* для среды, в которой вызывается команда `nice` или `renice`, означает абсолютное значение фактора уступчивости, а *инкр* — относительное. Везде, где указывается значение *-инкр* или *-декр*, для ввода отрицательных значений можно использовать два дефиса (например `--10`). Знаки "плюс" в команде **nice** обязательны только для интерпретатора команд; в остальных случаях они игнорируются.

**Таблица 5.3. Как выражаются приоритеты в различных версиях команд `nice` и `renice`**

| Система | Диапазон | ОС-команда `nice` | Команда `nice` в `csh` | Команда `renice` |
|---|---|---|---|---|
| Linux | 20–19 | *-инкр* или `-n` *инкр* | *+инкр* или *-инкр* | *приор* |
| Solaris | 0–39 | *-инкр* или `-n` *инкр* | *+инкр* или *-инкр* | *инкр* или `-n` *инкр* |
| HP-UX | 0–39 | *-приор* или `-n` *приор* | *+инкр* или *-инкр* | `-n` *приор*[a] |
| AIX | 20–19 | *-инкр* или `-n` *инкр* | *+инкр* или *-инкр* | `-n` *инкр* |

[a] Используется абсолютное значение приоритета, но к заданному значению добавляется число 20.

Самый распространенный из высокоприоритетных процессов в современных системах — `ntpd`, демон тактовой синхронизации. Поскольку для него быстрый доступ к центральному процессору имеет очень большое значение, этому демону обычно назначается фактор уступчивости, на 12 позиций ниже стандартного уровня.

Если какой-либо процесс займет столько времени центрального процессора, что показатель средней загруженности системы достигнет отметки 65, то перед выполнением команд, необходимых для исследования этой проблемы, понадобится запустить с помощью команды **nice** высокоприоритетный интерпретатор. В противном случае выполнение даже простейших команд может быть затруднено.

## 5.7. Текущий контроль процессов: команда ps

Команда **ps** — основной инструмент, которым системный администратор пользуется для текущего контроля процессов. Версии этой команды различаются аргументами и выходным форматом, но, по сути, выдают одну и ту же информацию. В основном, различие в версиях — это следствие разных путей развития систем UNIX. Кроме того, поставщики систем могут настраивать эту команду с учетом конфигурации системы, так как она тесно связана с особенностями обработки процессов в ядре и поэтому часто отражает изменения в ядре.

С помощью команды `ps` можно получить информацию об идентификаторах, приоритете и управляющем терминале того или иного процесса. Она также позволяет выяснить, какой объем памяти использует процесс, сколько времени центрального процессора заняло его выполнение, каково его текущее состояние (выполняется, остановлен,

[8] Хуже того, системная версия команды интерпретирует строку ***nice -5*** как положительное приращение, а встроенная версия интерпретирует эту же строку как отрицательное приращение.

простаивает и т.д.). Процессы-зомби в листинге команды обозначаются как <exiting> или <defunct>.

Команда **ps** стремительно усложнилась за последние несколько лет. Некоторые поставщики бросили попытки стандартизировать выходной формат этой команды и сделали ее полностью конфигурируемой. Проведя небольшую настройку, можно получить практически любые требуемые результаты. В Linux команда `ps` является настоящим хамелеоном и понимает наборы опций из ряда других систем. Имеется также специальная переменная среды, с помощью которой можно выбрать нужный набор.

Не пугайтесь всех этих сложностей: пусть они будут кошмаром разработчиков ядра, а не системных администраторов! Для повседневной работы достаточно знать лишь несколько наиболее важных опций команды **ps**.

Получить список всех процессов, выполняющихся в системах Linux и AIX, можно с помощью команды **ps aux**. Ключ **a** используется для отображения всех процессов, **x** — для отображения процессов, отсоединенных от терминала, а ключ **u** обеспечивает фильтрование по имени или идентификатору пользователя, который запустил программу. Ниже показаны результаты работы команды **ps aux** в системе Red Hat (результат работы той же команды для системы AIX имеет небольшие отличия).

```
redhat$ ps aux
   USER   PID  %CPU %MEM    VSZ   RSS   TTY STAT TIME COMMAND
   root     1   0.1  0.2   3356   560     ?    S 0:00 init [5]
   root     2     0    0      0     0     ?   SN 0:00 [ksoftirqd/0]
   root     3     0    0      0     0     ?   S< 0:00 [events/0]
   root     4     0    0      0     0     ?   S< 0:00 [khelper]
   root     5     0    0      0     0     ?   S< 0:00 [kacpid]
   root    18     0    0      0     0     ?   S< 0:00[kblockd/0]
   root    28     0    0      0     0     ?   S  0:00 [pdflush]
...
   root   196     0    0      0     0     ?   S  0:00 [kjournald]
   root  1050     0  0.1   2652   448     ? S<s  0:00 udevd
   root  1472     0  0.3   3048  1008     ? S<s  0:00 /sbin/dhclient -1
   root  1646     0  0.3   3012  1012     ? S<s  0:00 /sbin/dhclient -1
   root  1733     0    0      0     0     ?   S  0:00 [kjournald]
   root  2124     0  0.3   3004  1008     ?  Ss  0:00 /sbin/dhclient -1
   root  2182     0  0.2   2264   596     ?  Ss  0:00 syslogd -m 0
   root  2186     0  0.1   2952   484     ?  Ss  0:00 klogd -x
   rpc   2207     0  0.2   2824   580     ?  Ss  0:00 portmap
rpcuser  2227     0  0.2   2100   760     ?  Ss  0:00 rpc.statd
   root  2260     0  0.4   5668  1084     ?  Ss  0:00 rpc.idmapd
   root  2336     0  0.2   3268   556     ?  Ss  0:00 /usr/sbin/acpid
   root  2348     0  0.8   9100  2108     ?  Ss  0:00 cupsd
   root  2384     0  0.6   4080  1660     ?  Ss  0:00 /usr/sbin/sshd
   root  2399     0  0.3   2780   828     ?  Ss  0:00 xinetd -stayalive
   root  2419     0  1.1   7776  3004     ?  Ss  0:00 sendmail: accept
...
```

Команды, имена которых заключены в квадратные скобки, в действительности являются не командами, а потоками ядра, запланированными в качестве процессов. Описание полей приведено в табл. 5.4.

Еще один полезный набор аргументов команды **ps** для систем Linux и AIX — **lax**, предоставляющий дополнительную техническую информацию. Ключи **a** и **x** описаны вы-

ше (отображают все процессы), а ключ `l` означает выбор "длинного" формата вывода данных. Команда **`ps lax`** выполняется быстрее, чем команда **`ps aux`**, так как не сопоставляет идентификаторы процессов с именами пользователей. Это может оказаться весьма важным фактором, если система уже перегружена каким-то процессом.

**Таблица 5.4. Пояснения к выходной информации команды `ps aux`**

| Поле | Содержимое |
|---|---|
| USER | Имя владельца процесса |
| PID | Идентификатор процесса |
| %CPU | Доля времени центрального процессора (в процентах), выделенная процессу |
| %MEM | Часть реальной памяти (в процентах), используемая процессом |
| VSZ | Виртуальный размер процесса |
| RSS | Размер резидентного набора (количество страниц памяти) |
| TTY | Идентификатор управляющего терминала |
| STAT | Текущий статус процесса:<br>R — выполняется<br>D — ожидает записи на диск<br>S — неактивен (< 20 с)<br>T — приостановлен<br>Z — зомби<br>Дополнительные флаги:<br>W — процесс выгружен на диск<br>< — процесс имеет повышенный приоритет<br>N — процесс имеет пониженный приоритет<br>L — некоторые страницы блокированы в оперативной памяти<br>s — процесс является лидером сеанса |
| TIME | Количество времени центрального процессора, затраченное на выполнение процесса |
| COMMAND | Имя и аргументы команды [a] |

[a] Программы могут модифицировать эту информацию, так что она не всегда точно представляет реальную командную строку.

Ниже в сокращенном виде показаны результаты работы команды **`ps lax`**. Обратите внимание на дополнительные поля PPID (идентификатор родительского процесса), NI (фактор уступчивости) и WCHAN (ресурс, которого ожидает процесс).

```
redhat$ ps lax
   F UID   PID PPID PRI  NI  VSZ  RSS  WCHAN STAT TIME COMMAND
   4   0     1    0  16   0 3356  560 select    S 0:00 init [5]
   1   0     2    1  34  19    0    0 ksofti   SN 0:00 [ksoftirqd/0
   1   0     3    1   5 -10    0    0 worker   S< 0:00 [events/0]
   1   0     4    3   5 -10    0    0 worker   S< 0:00 [khelper]
   5   0  2186    1  16   0 2952  484 syslog   Ss 0:00 klogd -x
   5  32  2207    1  15   0 2824  580      -   Ss 0:00 portmap
   5  29  2227    1  18   0 2100  760 select   Ss 0:00 rpc.statd
   1   0  2260    1  16   0 5668 1084      -   Ss 0:00 rpc.idmapd
   1   0  2336    1  21   0 3268  556 select   Ss 0:00 acpid
   5   0  2384    1  17   0 4080 1660 select   Ss 0:00 sshd
   1   0  2399    1  15   0 2780  828 select   Ss 0:00 xinetd -sta
   5   0  2419    I  16   0 7776 3004 select   Ss 0:00 sendmail: a
...
```

В системах Solaris и HP-UX удобно использовать команду **ps -ef**. Ключ **e** позволяет выбрать все процессы, а ключ **f** устанавливает выходной формат. (Команда **ps -ef** работает и в системах AIX и Linux — обратите внимание на использование дефиса.)

```
solaris$ ps -ef
 UID    PID PPID   C     STIME    TTY  TIME  COMD
 root     0    0  80    Dec 21      ?  0:02  sched
 root     1    0   2    Dec 21      ?  4:32  /etc/init-
 root     2    0   8    Dec 21      ?  0:00  pageout
 root   171    1  80    Dec 21      ?  0:02  /usr/lib/sendmail-bd
 trent 8482 8444  35  14:34:10  pts/7  0:00  ps-ef
 trent 8444 8442 203  14:32:50  pts/7  0:01  -csh
...
```

Описание полей (как результата выполнения команды `ps -ef`) приведено в табл. 5.5.

**Таблица 5.5. Пояснения к выходной информации команды `ps -ef`**

| Поле | Содержимое |
|---|---|
| UID | Идентификатор пользователя |
| PID | Идентификатор процесса |
| PPID | Идентификатор родительского процесса |
| C | Информация об использовании/планировании времени центрального процессора |
| STIME | Время запуска процесса |
| TIME | Количество времени центрального процессора, затраченное на выполнение процесса |
| TTY | Терминал |
| COMMAND | Имя и аргументы команды |

Подобно команде **ps lax** в системах Linux и AIX, команда **ps -elf** выводит дополнительные данные по системам Solaris и HP-UX.

```
% ps -elf
 F S   UID PID PPID  C P NI     ADDR  SZ    WCHAN TIME COMD
19 T root   0    0 80 0 SY f00c2fd8   0          0:02 sched
 8 S root   1    0 65 1 20 ff26a800  88 ff2632c8 4:32 init-
 8 S root 142    1 41 1 20 ff2e8000 176  f00cb69 0:00 syslogd
...
```

Чтобы результаты поместились на странице, поля STIME и TTY были опущены; они идентичны полям, сгенерированным командой **ps -ef**. Описание полей приведено в табл. 5.6.

**Таблица 5.6. Пояснения к выходной информации команды `ps -elf`**

| Поле | Содержимое |
|---|---|
| F | Флаги процесса (зависят от системы — редко полезны для системных администраторов) |
| S | Статус процесса:<br>O — выполняется<br>R — готов к выполнению<br>Z — зомби<br>S — неактивен (ожидает событие)<br>T — приостановлен или отслеживается<br>D — полностью неактивен (обычно диск) |

*Окончание табл. 5.6*

| Поле | Содержимое |
|---|---|
| C | Информация об использовании/планировании времени центрального процессора |
| P | Запланированный приоритет (внутренний для ядра, отличный от значения фактора "уступчивости") |
| NI | Значение фактора "уступчивости"или SY для системных процессов |
| ADDR | Адрес процесса |
| SZ | Размер (в страницах) процесса в основной памяти |
| WCHAN | Адрес объекта, которого ожидает процесс |

## 5.8. Динамический мониторинг процессов с помощью команд top, prstat и topas

Команда ps позволяет сделать только разовый, моментальный, "снимок" системы, но получить полную картину всего происходящего в ней довольно сложно. Для этого служит свободно распространяемая утилита **top**, которая работает во многих системах, регулярно обновляя сводку активных процессов и используемых ими ресурсов. В AIX эквивалентную функцию выполняет утилита topas, а в Solaris — **prstat** .

Рассмотрим пример.

```
ubuntu$ top
top - 16:37:08 up  1:42,  2 users,  load average: 0.01, 0.02, 0.06
Tasks:  76 total,   1 running,  74 sleeping,   1 stopped,   0 zombie
Cpu(s):  1.1% us,  6.3% sy,  0.6% ni,  88.6% id,  2.1% wa,  0.1% hi,  1.3% si
 Mem:    256044k total, 254980k used, 1064k free, 15944k buffers
 Swap:   524280k total, 0k used, 524280k free, 153192k cached

  PID USER PR  NI  VIRT RES  SHR S %CPU %MEM   TIME+ COMMAND
 3175 root 15   0 35436 12m 4896 S  4.0  5.2 01:41.9 X
 3421 root 25  10 29916 15m 9808 S  2.0  6.2 01:10.5 rhn-applet-gui
    1 root 16   0  3356 560  480 S  0.0  0.2 00:00.9 init
    2 root 34  19     0   0    0 S  0.0    0 00:00.0 ksoftirqd/0
    3 root  5 -10     0   0    0 S  0.0    0 00:00.7 events/0
    4 root  5 -10     0   0    0 S  0.0    0 00:00.0 khelper
    5 root 15 -10     0   0    0 S  0.0    0 00:00.0 kacpid
   18 root  5 -10     0   0    0 S  0.0    0 00:00.0 kblockd/0
   28 root 15   0     0   0    0 S  0.0    0 00:00.0 pdflush
   29 root 15   0     0   0    0 S  0.0    0 00:00.3 pdflush
   31 root 13 -10     0   0    0 S  0.0    0 00:00.0 aio/0
   19 root 15   0     0   0    0 S  0.0    0 00:00.0 khubd
   30 root 15   0     0   0    0 S  0.0    0 00:00.2 kswapd0
  187 root  6 -10     0   0    0 S    0    0 00:00.0 kmirrord/0
  196 root 15   0     0   0    0 S    0    0 00:01.3 kjournald
..
```

По умолчанию эта информация обновляется каждые десять секунд. Наиболее активные процессы отображаются первыми. Команда **top** позволяет также посылать процессам сигналы и использовать команду **renice**, чтобы пользователь мог наблюдать за тем, как его действия влияют на общее состояние системы.

Пользователь root может запустить команду **top** с опцией **-q**, чтобы обеспечить ей максимально возможный приоритет. Это очень удобно, если нужно обнаружить процесс, который существенно замедлил работу системы.

## 5.9. Файловая система /proc

Версии команд **ps** и **top**, используемые в Linux, считывают информацию о состоянии процессов из каталога **/proc** — псевдофайловой системы, в которой ядро помещает большой объем интересной информации о состоянии системы. Несмотря на имя **/proc** (и имя базового типа файловой системы — "proc"), хранящаяся в этом каталоге информация не ограничивается одними лишь процессами — здесь хранится вся информация о состоянии и статистические сведения, генерируемые ядром. Некоторые параметры можно даже изменять, записывая новые значения в соответствующий файл каталога **/proc**, — ряд примеров приведен в разделе 13.3.

Хотя некоторую информацию проще получать с помощью таких интерфейсных команд, как **vmstat** и **ps**, некоторую менее популярную информацию придется считывать непосредственно из каталога **/proc**. Поэтому имеет смысл просмотреть его, чтобы ознакомиться со всеми помещенными в него файлами. Команда **man proc** позволяет ознакомиться с рядом полезных советов и приемов.

Поскольку ядро создает содержимое файлов каталога **/proc** на лету (во время их считывания), большинство из них выглядят пустыми при их открытии с помощью команды **ls -l**. Для просмотра действительного содержимого этих файлов придется прибегнуть к командам **cat** или **more**. Но будьте осторожны: некоторые файлы содержат двоичные данные либо ссылаются на двоичные данные, непосредственный просмотр которых может поставить в тупик эмулятор терминала.

Информация, относящаяся к конкретным процессам, распределена по подкаталогам, названным по идентификаторам процессов. Например, каталог **/proc/1** всегда содержит информацию о демоне **init**. Наиболее полезные файлы с информацией о процессах перечислены в табл. 5.7.

**Таблица 5.7. Файлы с информацией о процессах каталога /proc (нумерованные подкаталоги)**

| Файл | Содержимое |
|---|---|
| cmd | Команда или программа, выполняемая процессом |
| cmdline[a] | Полная командная строка процесса (разделенная нулями) |
| cwd | Символьная ссылка на текущий каталог процесса |
| environ | Переменные среды процесса (разделенные нулями) |
| exe | Символьная ссылка на файл, который должен выполняться |
| fd | Подкаталог, содержащий ссылки на дескрипторы каждого открытого файла |
| maps | Информация отображения памяти (сегменты совместного использования, библиотеки и т.п.) |
| root | Символьная ссылка на корневой каталог процесса (определенный командой **chroot**) |
| stat | Информация об общем состоянии процесса (для ее получения лучше всего использовать команду **ps**) |
| statm | Информация об использовании памяти |

[a] Может быть недоступна, если запись информации о процессе выполняется из памяти.

Отдельные компоненты внутри файлов **cmdline** и **environ** разделены символами нуля, а не символами перевода строки. Для того чтобы их содержимое было более читабельным, его можно фильтровать с помощью команды **tr "\000" "\n"**.

В подкаталоге **fd** открытые файлы представлены символьными ссылками. Дескрипторы файлов, которые связаны с каналами или сетевыми сокетами, не имеют связанных с ними имен файлов. Вместо этого в качестве целевой ссылки ядро предоставляет обобщенное описание.

Файл **maps** полезен при определении библиотек, с которыми связана или от которых зависит та или иная программа.



В системах Solaris и AIX также используется файловая система на основе каталога **/proc**, но она не включает дополнительные данные о состоянии системы и статистические сведения, генерируемые ядром (как в Linux). Группа инструментов, именуемая утилитами **proc**, отображает некоторую полезную информацию о выполнении процессов. Например, команда **procsig** в AIX и ее Solaris-эквивалент **psig** выводят список сигналов для заданного процесса (сигнальные действия и обработчики сигналов). Самые полезные утилиты `proc` и их функции приведены в табл. 5.8.

**Таблица 5.8. Команды чтения /proc-информации в системах AIX и Solaris**

| Solaris[a] | AIX | Description |
|---|---|---|
| **pcred [pid \| pid]** | **proccred [pid]** | Выводит/устанавливает реальный, текущий и сохраненный идентификаторы |
| **pldd [-F] [pid \| pid]** | **procldd [pid]** | Отображает библиотеки, связанные с процессом (подобно `ldd`) |
| **psig [pid]** | **procsig [pid]** | Перечисляет сигнальные действия и обработчики |
| **pfiles [pid]** | **procfiles [pid]** | Выводит открытые файлы |
| **pwdx [pid]** | **procwdx [pid]** | Выводит текущий рабочий каталог |
| **pwait [pid]** | **procwait [pid]** | Ожидает завершения процесса |

[a] В Solaris некоторые `proc`-утилиты (это, в основном, инструменты отладки) принимают в качестве входных данных файл **core**.

В системе HP-UX не используется файловая система **/proc** или ее эквивалент.

## 5.10. Отслеживание сигналов и системных вызовов: команды strace, truss и tusc

Иногда определение действий, действительно выполняемых данным процессом, может быть затруднительным. Умозаключение приходится делать на основе косвенных данных, полученных от файловой системы и с помощью таких средств, как программа **ps**.

Linux позволяет непосредственно следить за процессом с помощью команды **strace**, которая отображает каждый системный вызов, выполняемый процессом, и каждый получаемый им сигнал. Аналогичной командой для систем Solaris и AIX является команда **truss**. В HP-UX эквивалентом служит команда **tusc**, при этом утилиту **tusc** необходимо инсталлировать отдельно.

Эту команду можно даже связать с выполняемым процессом, последить за ним в течение некоторого времени, а затем отключиться от процесса, не прерывая его[9].

Хотя системные вызовы выполняются на сравнительно низком уровне абстракции, обычно вывод команды **strace** позволяет получить достаточно полную информацию об активности процесса. Например, следующий журнал был получен в результате выполнения команды **strace** применительно к активной копии команды **top**.

```
redhat$ sudo strace -p 5810
gettimeofday( {1116193814, 213881}, {300, 0} )              = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY)  = 7
fstat64(7, {st_mode=S_IFDIR|0555, st_size=0, ...} )         = 0
fcntl64(7, F_SETFD, FD_CLOEXEC)                             = 0
getdents64(7, /* 36 entries */, 1024)                       = 1016
getdents64(7, /* 39 entries */, 1024)                       = 1016
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...} )  = 0
open("/proc/1/stat", O_RDONLY)                              = 8
read(8, "1 (init) S 0 0 0 0 -1 4194560 73"..., 1023)        = 191
close(8)                                                    = 0
...
```

Команда **strace** не только отображает имя каждого системного вызова, выполненного процессом, но и раскрывает аргументы и отображает результирующий код, возвращенный ядром.

Команда **strace** снабжена флагами, которые описаны в соответствующей man-странице. Например, флаг **-f** используется для разветвленных процессов, и его полезно применять для отслеживания демонов (например, **httpd**), которые порождают множество дочерних процессов. Опция **-e** позволяет отображать только файловые операции, что особенно удобно для определения местоположения "неуловимых" файлов конфигурации.

В этом примере команда **top** начинает свою работу с проверки текущего значения времени. Затем она открывает каталог **/proc** и считывает его содержимое, тем самым получая список процессов, выполняемых в текущий момент. Команда **top** обращается к статической копии каталога, представляющей процесс **init**, а затем открывает **/proc/1/stat**, чтобы прочесть информацию о состоянии этого процесса.

Вот даже более простой пример (применительно к команде **date**) использования команды **truss** в системе Solaris.

```
solaris$ truss date
...
time()                                                  = 1242507670
brk(0x00024D30)                                         = 0
brk(0x00026D30)                                         = 0
open("/usr/share/lib/zoneinfo/US/Mountain", O_RDONLY)   = 3
fstat64(3, 0xFFBFFAF0)                                  = 0
read(3, " T Z i f\0\0\0\0\0\0\0\0".., 877)              = 877
close(3)                                                = 0
ioctl(1, TCGETA, 0xFFBFFA94)                            = 0
fstat64(1, 0xFFBFF9B0)                                  = 0
write(1, " S a t   M a y  1 6  1".., 29)               = 29
Sat May 16 14:56:46 MDT 2009
_exit(0)
```

[9] По крайней мере, как правило. Команда `strace` может прерывать системные вызовы. В этих случаях отслеживаемый процесс должен быть подготовлен для повторного запуска этих вызовов. Таково стандартное правило выполнения программ в среде UNIX, но оно не всегда соблюдается.

Здесь после вывода информации о распределении памяти и библиотечных зависимостях (не показана), команда **date** использует системный вызов **time** для считывания системного времени, открывает файл, соответствующий часовому поясу для определения нужного смещения, и выводит отметку даты и времени с помощью системного вызова **write**.

## 5.11. Процессы, вышедшие из-под контроля

Об обработке неуправляемых процессов рассказывается также в разделе 29.5.

Неуправляемые процессы бывают двух видов: пользовательские, потребляющие слишком много системных ресурсов (например, времени центрального процессора или дискового пространства), и системные, которые внезапно выходят из-под контроля и начинают вести себя непредсказуемо. Процесс первого вида может быть вполне работоспособным, просто он некорректно обращается с ресурсами. В то же время системные процессы всегда должны подчиняться определенным правилам.

Процессы, занимающие слишком много времени центрального процессора, можно выявить, проанализировав результаты работы команды **ps** или **top**. Если очевидно, что пользовательский процесс потребляет больше ресурсов, чем ему действительно необходимо, его нужно исследовать. Кроме того, полезно выяснить номер процесса, ожидающего выполнения. Используйте команду **uptime**, которая (помимо текущего времени и времени, отработанного системой) показывает число пользователей в системе и среднюю загруженность системы за последние 1, 5 и 15 минут.

Прежде чем принимать к проблемному процессу серьезные меры, необходимо понять, какие действия он (этот процесс) старается выполнить. Это стоит сделать по двум причинам. Во-первых, процесс может быть вполне законным и действительно важным для пользователя. Нельзя же уничтожать процесс только из-за того, что он занимает много времени центрального процессора. Во-вторых, процесс может нести угрозу системе и выполнять разрушительные действия. В этом случае нужно точно узнать, что он натворил (например, взломал пароли), и устранить повреждения.

Процессы, которые используют значительный объем физической оперативной памяти, могут существенно снижать производительность системы. Размер памяти, занимаемый процессами, можно проверить с помощью команды **top**. Столбец VIRT содержит информацию об общем объеме виртуальной памяти, зарезервированной каждым процессом, а столбец RES — информацию о части этой памяти, которая в текущий момент времени записана в конкретные страницы памяти ("резидентный набор"). В системах Linux приложения, которые используют видеокарту (например, сервер X), получают "незаслуженное обвинение", поскольку видеопамять включается в подсчет объема используемой памяти.

Оба значения могут отражать ресурсы совместного использования, такие как библиотеки, что теоретически может приводить к неправильной оценке реальной ситуации. Более точная информация о памяти, потребляемой процессом, содержится в столбце DATA, который по умолчанию не отображается. Для того чтобы добавить этот столбец к информации, отображаемой командой **top**, после ее запуска нажмите клавишу <f> и выберите столбец DATA из списка. Отображаемое в этом столбце значение представляет объем памяти, занимаемый сегментами данных и стека каждого процесса, поэтому оно сравнительно специфично для отдельных процессов (оно представляет собой остаток деления на объем сегментов памяти совместного использования). По нему можно су-

дить как об относительном увеличении объема используемой памяти, так и об абсолютном ее объеме.

Результаты работы неуправляемых процессов могут заполнить всю файловую систему и вызвать, таким образом, бесчисленные проблемы. При переполнении файловой системы на экран выдается множество сообщений, и попытки записать что-либо на диск тоже повлекут за собой появление сообщений об ошибках.

Первое, что нужно сделать в подобной ситуации, — определить, какая именно файловая система заполнена и каким файлом она заполняется в данный момент. Об использовании файловой системы можно узнать с помощью команды **df -k**. В выходных данных этой команды поищите файловую систему, которая заполнена на 100% или более[10]. Для того чтобы понять, какой каталог занимает самую большую область памяти, используйте команду **du** для идентифицированной файловой системы. Повторяйте выполнение команды du до тех пор, пока не выявите самые большие файлы. Если вы не можете определить, какие процессы используют эти файлы, попробуйте получить больше информации с помощью команд **fuser** и **lsof** (подробнее об этих командах см. раздел 6.2).

Можно приостановить все подозрительные процессы, пока не обнаружится источник проблем. Выявив "нарушителя", удалите все созданные им файлы и не забудьте возобновить "дисциплинированные" процессы. В случае, если некоторый файл содержит полезные или очень важные данные, было бы разумно сжать его с помощью утилиты **gzip** и переименовать.

## 5.12. Рекомендуемая литература


- Bovet Daniel P. and Marko Cesati. *Understanding the Linux Kernel (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2006.
- Mckusick, Marshall Kirk and George V. Neville-Neil. *The Design and Implementation of the FreeBSD Operating System*. Reading, MA: Addison-Wesley Professional, 2004.
- Роберт Лав. *Разработка ядра Linux, 2-е изд.* (пер. с англ.: ИД "Вильямс", 2006 г.).


## 5.13. Упражнения

5.1. Объясните, какая связь существует между значением UID исполняемого файла и реальным и текущим идентификаторами выполняемого процесса. Каково назначение текущего идентификатора процесса, помимо контроля доступа к файлам?

5.2. Предположим, что кто-то из пользователей системы запустил длительный и ресурсоемкий процесс.

а) как можно было бы распознать процесс, который узурпирует ресурсы?

б) вы подозреваете, что этот процесс необходим пользователю и не должен быть уничтожен. С помощью каких команд можно приостановить процесс на время проверки?

в) впоследствии выясняется, что процесс запущен начальником и должен быть

[10] В большинстве реализаций файловых систем часть всей памяти (около 5%) резервируется "на всякий случай", но процессы, выполняемые от имени суперпользователя, могут "вторгаться на эту территорию", в результате чего отчеты могут содержать показатели использования памяти, превышающие 100%.

продолжен. С помощью каких команд можно возобновить выполнение задания?

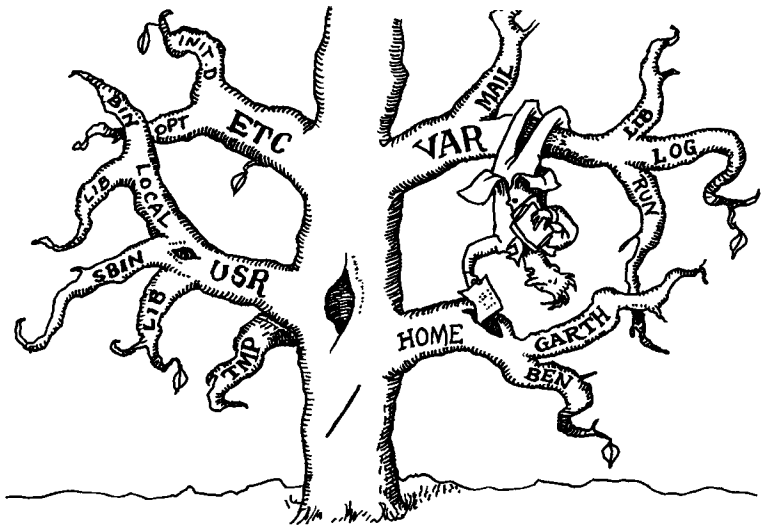
г) теперь предположим, что процесс должен быть уничтожен. Какой сигнал нужно ему послать и почему? Что если необходимо гарантированно уничтожить процесс?

5.3. Найдите программу, в которой существует "утечка памяти" (при необходимости напишите собственную программу). Проследите за работой программы с помощью команды `ps` или `top`, обращая внимание на показатели использования памяти.

5.4. ★ Напишите простой Perl-сценарий обработки вывода команды **ps** для определения суммы значений в столбцах `VSZ` и `RSS` для процесса, выполняющегося в системе. Как эти показатели соотносятся с реальным объемом физической памяти и размером раздела подкачки?

# Глава 6

# Файловая система




Ответьте, не раздумывая, на вопрос: что из перечисленного ниже можно считать элементами файловой системы?

- Процессы
- Аудиоустройства
- Структуры данных ядра и параметры настройки
- Каналы межзадачного взаимодействия

Если речь идет о системах UNIX или Linux, ответ будет таков: все перечисленное выше. Ну и, конечно же, в файловую систему входят собственно файлы[1].

Хотя основным назначением файловой системы является упорядочение хранимых ресурсов системы (т.е. файлов), программистам не хотелось каждый раз заново изобретать колесо при управлении объектами других типов. Очень часто оказывалось удобным представлять такие объекты в виде элементов файловой системы. Подобный унифицированный подход имеет как преимущества (единый программный интерфейс, легкий доступ из интерпретатора команд), так и недостатки (реализация файловых систем по методу доктора Франкенштейна), но независимо от того, нравится он вам или нет, именно такой подход применяется в UNIX (а значит, и в Linux).

Файловую систему можно представить состоящей из четырех основных компонентов:

- пространство имен — методы именования объектов и организации их в виде единой иерархии;

[1] Точнее говоря, эти элементы представлены внутри файловой системы. В большинстве случаев файловая система служит "местом встречи" клиентов с драйверами и серверами, которые им требуются.

- API[2] — набор системных вызовов для перемещения между объектами и управления ими;
- модель безопасности — схема защиты, сокрытия и совместного использования объектов;
- реализация — программный код, который связывает логические модели с дисковой подсистемой.

Сетевая файловая система, NFS, рассматривается в главе 18.

В современных ядрах систем определен абстрактный интерфейс, позволяющий работать с различными файловыми системами. Одни части файлового дерева обрабатываются традиционной дисковой подсистемой, другие — управляются специальными драйверами ядра. Например, за работу файловых систем — как сетевых (Network File System — NFS), так и общих межсетевых (Common Internet File System — CIFS), — отвечает драйвер, который перенаправляет запросы серверу, расположенному на другом компьютере.

К сожалению, концептуальные границы нечетко очерчены, поэтому имеется слишком много "особых" случаев. К примеру, файлы устройств позволяют программам взаимодействовать с драйверами ядра. Они не являются файлами данных, но обрабатываются файловой системой, а их характеристики записываются на диск.

Еще одним усложняющим (но, в конечном счете, полезным) фактором является то, что ядро поддерживает несколько типов дисковых файловых систем. Среди наиболее перспективных на данный момент файловых систем можно назвать ext3 и ext4, которые во многих дистрибутивах являются базовыми, а также ReiserFS, JFS компании IBM, ZFS компании Sun, VxFS компании Veritas. К этому списку можно добавить и Btrfs — новую свободную файловую систему, все еще разрабатываемую при поддержке компании Oracle.

Есть много реализаций файловых систем с другой семантикой, таких как FAT и NTFS (используются в Microsoft Windows) и 9660 (применяется на компакт-дисках). Linux поддерживает больше файловых систем, чем любая другая разновидность UNIX. Столь широкий выбор упрощает обмен файлами с другими системами.

Организация файловой системы — очень обширная тема, которую мы рассмотрим с нескольких точек зрения. В этой главе рассказывается, где обычно хранятся файлы, и описываются характеристики файлов, а также рассматриваются ключевые команды, позволяющие задавать атрибуты файлов. В главе 8 структура файловой системы описывается на более низком уровне. В главе 18 рассматриваются средства совместного доступа к файлам, столь часто используемые в системе Linux. Средства совместного доступа к файлам в системе Microsoft Windows рассматриваются в главе 30.

При столь большом разнообразии реализаций файловых систем может показаться странным, что данная глава посвящена всего лишь одной файловой системе. Однако на отдельных реализациях можно специально не останавливаться, поскольку в большинстве современных файловых систем либо предприняты попытки обеспечить более эффективное и надежное применение традиционных функциональных возможностей, либо дополнительные функциональные возможности реализованы в качестве отдельного слоя, работающего поверх семантики стандартной файловой системы (некоторые файловые системы реализуют оба направления). Как бы то ни было, слишком много программных пакетов ориентировано на описанную в этой главе модель, чтобы ею можно пренебречь.

[2] Интерфейс прикладных программ (Application Programming Interface — API) — общий термин для описания набора подпрограмм, организованных в виде библиотеки или совокупности библиотек и доступных программистам.

# 6.1. Имена файлов и каталогов

Файловая система — это единая иерархическая структура, которая начинается с каталога / и разветвляется, охватывая произвольное число подкаталогов. Каталог самого верхнего уровня называется *корневым*. Эта моноиерархическая система отличается от используемой в Windows, где применяется понятие пространства имен, основанное на принципе деления диска на разделы.

## Абсолютные и относительные пути

Цепочка имен каталогов, через которые необходимо пройти для доступа к заданному файлу, вместе с именем этого файла образуют путь к файлу. Путь может быть абсолютным (например, **/tmp/foo**) или относительным (например, **book4/filesystem**). Последние интерпретируются начиная с текущего каталога. Возможно, многие считают, что текущий каталог задается интерпретатором команд. На самом деле текущий каталог есть у каждого процесса. (Большинство процессов никогда не изменяет свои рабочие каталоги, и поэтому они просто наследуют текущий каталог процесса, который их запустил.)

Термины *имя файла* и *путь* в той или иной степени являются взаимозаменяемыми (по крайней мере, так они трактуются в данной книге). Соответственно, имя файла бывает полным (абсолютный путь) или сокращенным (относительный путь). Файловое дерево может иметь произвольную глубину, однако каждый компонент имени файла должен состоять не более чем из 255 символов.

Существует также ограничение на длину пути, который вы можете передавать ядру в качестве аргумента системного вызова (4095 байт в Linux и 1023 байт в более старых системах). Для того чтобы получить доступ к файлу, полное имя которого превышает эти ограничения, необходимо с помощью команды **cd** перейти в промежуточный каталог, а затем воспользоваться сокращенным именем.

## Использование пробелов в именах файлов

На имена файлов и каталогов не налагаются никакие ограничения, за исключением того, что длина имени не должна превышать заданный предел и в имя нельзя включать символы косой черты и нулевые символы. В частности, допускаются имена, содержащие пробелы. К сожалению, по давней традиции аргументы командной строки в системе UNIX разделяются пробелами, поэтому старые программы интерпретируют пробелы в именах файлов как признак конца одного имени и начала другого.

Первоначально пробелы в именах файлов применялись в основном в файловых системах, используемых на компьютерах Mac и PC, но теперь они проникли и в культуру UNIX и их можно встретить в некоторых стандартных программных пакетах. По этому поводу не может быть двух мнений: административные сценарии должны быть готовы к обработке пробелов в именах файлов (не говоря уже об апострофах, звездочках и других "опасных" знаках препинания).

В интерпретаторе команд и его сценариях имена с пробелами необходимо просто заключать в двойные кавычки. Например, команда

```
$ less "My excellent file.txt"
```

будет воспринята как попытка передать программе **less** единый аргумент **My excellent file.txt**. Отдельные пробелы можно также предварять символом обратной косой черты. Функция завершения имен файлов, предоставляемая современными интерпретаторами (обычно она связана с клавишей <Tab>), позволяет это делать автоматически.

При написании сценариев полезным инструментом является опция **-print0** команды **find**. В сочетании с опцией **-0** команды **xargs** эта опция позволяет сочетанию команд **find/xargs** работать правильно независимо от наличия пробелов в именах файлов. Например, команда

```
$ find /home -type f -size +1M -print0 | xargs -0 ls -l
```

выводит длинный **ls**-список всех файлов, хранящихся в логическом разделе **/home**, размер которых превышает один мегабайт.


К сожалению, система HP-UX поддерживает команду **find -print0**, а не **xargs -0**. Система AIX не поддерживает ни один из этих вариантов. Но в любой системе вы можете инсталлировать GNU-пакет **findutils**, чтобы получить текущие версии команд **find** и **xargs**. (В качестве альтернативного варианта вместо команды **xargs** можно использовать опцию **-exec** команды **find**, хотя она работает менее эффективно.)

## 6.2. Монтирование и демонтирование файловой системы

Файловое дерево формируется из отдельных частей, называемых файловыми системами, каждая из которых содержит корневой каталог и список его подкаталогов и файлов. Термин "файловая система" имеет, по сути, два значения. С одной стороны, это составная часть файлового дерева, а с другой — все файловое дерево и алгоритмы, с помощью которых ядро управляет им. Как правило, значение термина становится ясным из контекста.

Большинство файловых систем представляет собой разделы диска или логические тома с памятью на дисках, но, как уже было сказано, файловая система может принять облик всего, что подчиняется определенным функциональным правилам, — сетевых файловых систем, компонентов ядра, резидентных дисков и т.д. В Linux и Solaris есть даже оригинальная файловая система, позволяющая монтировать отдельные файлы, как если бы они были физическими устройствами. Это очень удобно для разработки образов файловых систем, поскольку отпадает необходимость в перераспределении дисковой памяти.

В большинстве случаев файловые системы присоединяются к файловому дереву с помощью команды **mount**[3]. Эта команда связывает каталог существующего файлового дерева, называемый *точкой монтирования*, с корневым каталогом новой файловой системы. На время монтирования доступ к прежнему содержимому точки монтирования становится невозможным. Впрочем, в большинстве случаев точка монтирования — это пустой каталог.

Например, команда

```
$ sudo mount /dev/sda4 /users
```

монтирует на устройстве **/dev/hda4** файловую систему **/users**. По окончании монтирования можно с помощью команды **ls /users** просмотреть содержимое файловой системы.

[3] Мы говорим "в большинстве случаев", поскольку в файловой системе ZFS (в Solaris) принят несколько другой подход к монтированию и демонтированию, не говоря уже о других аспектах администрирования файловых систем. Возможно, читатели ожидали придирчивых комментариев относительно непозволительной несовместимости в этой части систем, но структура ZFS представляет собой значительное усовершенствование, и мы с нетерпением ожидаем того дня, когда оно будет взято на вооружение другими системами. Между тем мы считаем своим долгом описывать файловую систему ZFS несколько обособленно. Подробнее см. раздел 8.10.

Список смонтированных пользователями файловых систем хранится в файле **/etc/fstab**, **/etc/vfstab** (Solaris) или **/etc/filesystems** (AIX). Благодаря этому возможны автоматическая проверка (с помощью команды **fsck**) и монтирование (с помощью команды **mount**) файловых систем на этапе начальной загрузки, а также выполнение коротких команд наподобие **mount /usr** (точное местонахождение монтируемой файловой системы ищется в файле **/etc/fstab**). Информация, содержащаяся в этом файле, служит документацией к схеме расположения файловых систем на диске. Подробнее файл **fstab** описан в разделе 8.9.

Файловые системы демонтируются командой **umount**. "Занятую" файловую систему демонтировать невозможно. В ней не должно быть ни открытых файлов, ни выполняющихся процессов с их текущими каталогами. Если демонтируемая файловая система содержит исполняемые программы, они не должны быть запущены.

В системе Linux предусмотрена возможность "ленивого" демонтирования (с помощью команды **umount -l**), которая удаляет файловую систему из иерархии имен, но в действительности не демонтирует ее до тех пор, пока все существующие файловые ссылки не будут закрыты. Полезность этой опции достаточно спорна. Во-первых, нет никакой гарантии, что существующие ссылки когда-либо будут закрыты сами по себе, кроме того, "наполовину демонтированное" состояние файловой системы может нарушать семантическую целостность для использующих ее программ. Они могут выполнять операции чтения и записи с существующими дескрипторами файлов, но не могут открывать новые файлы или выполнять какие-либо другие операции файловой системы.

Команда **umount -f** предназначена для "насильственного" демонтирования занятой файловой системы и поддерживается во всех рассмотренных нами системах. Однако не стоит использовать ее в системах, не использующих NFS. Кроме того, она может не применяться к определенным типам файловых систем (например, для таких, как системы `ext3` или `ext4`, которые ведут системные журналы).

Вместо использования команды **umount -f**, когда оказывается, что файловая система, которую вы пытаетесь демонтировать, занята, запустите команду **fuser**, чтобы узнать, какие процессы работают с файловой системой. Команда **fuser -c** *точка_монтирования* выводит идентификаторы всех процессов, обращающихся к файлам или каталогам указанной файловой системы, а также ряд буквенных кодов, которые отображают природу этой активности.

```
$ fuser -c /usr
/usr:     157tm     315ctom     474tom     5049tom     84tm     496ctom     490tm
      16938c     16902ctm     358ctom     484tm
```

Буквенные коды зависят от конкретной системы. Назначение кодов ниже описано в табл. 6.1, но детали обычно не имеют большого значения; главное для нас — идентификаторы процессов (PID).

**Таблица 6.1. Коды команды `fuser -c`**

| Коды | Значение |
|---|---|
| `f,o` | Процесс открыл файл для чтения или записи |
| `c` | В файловой системе находится текущий каталог процесса |
| `e,t` | Процесс в данный момент выполняет программу |
| `r` | В файловой системе находится корневой каталог процесса (задается с помощью команды **chroot**) |
| `m,s` | Процесс отображает в памяти файл или совместно используемую библиотеку |

Для того чтобы в точности определить, что собой представляют эти процессы, вызовите команду ps, передав ей список идентификаторов, о которых сообщила команда fuser.

```
$ ps -fp "157 315 5049"
 UID   PID PPID C  STIME TTY TIME CMD
 root 5049  490 0 Oct 14   ? 0:00 /usr/bin/X11/xdm
 root  157    1 0 Jun 27   ? 5:26 /usr/sbin/named
   lp  315    1 0 Jun 27   ? 0:00 /usr/lib/lpsched
...
```

Список идентификаторов взят в кавычки, чтобы интерпретатор передал его команде ps как один аргумент.

В системах Linux можно избежать необходимости задания идентификаторов (PID) в команде **ps**, выполнив команду **fuser** с флагом **-v**. Этот вариант команды генерирует более читабельный результат, который включает имя команды.

```
$ fuser -cv /usr
                USER  PID  ACCESS  COMMAND
/usr            root  444  ....m   atd
                root  499 .. ..m   sshd
                root  520  ....m   lpd
                ...
```

Буквенные коды в столбце ACCESS такие же, как в результате выполнения команды **fuser -c**.

Более удачной альтернативой команде **fuser** является утилита **lsof**. Утилита **lsof** — более сложная программа, чем **fuser**, и, соответственно, результаты ее работы более содержательны. Она доступна на сайте people.freebsd.org/~abe и работает во всех наших примерах систем.

В Licистемenux сценарии, предназначенные для поиска конкретной информации об использовании файловых систем процессами, располагают возможностью непосредственного чтения файлов, хранящихся в каталоге **/proc**. Однако команда **lsof -F**, форматирующая вывод команды **lsof** для облегчения синтаксического анализа, — более простое и переносимое решение. Для запроса только той информации, которая действительно необходима, следует использовать дополнительные флаги командной строки.

## 6.3. Организация файловой системы

Файловые системы в UNIX никогда не были хорошо организованы. Одновременно используется много разных, несовместимых соглашений об именовании файлов. Во многих случаях файлы группируются по выполняемым функциям, независимо от того, как часто они изменяются. Это затрудняет обновление операционной системы. Например, каталог **/etc** содержит ряд файлов, которые никогда не меняются, а также полностью локальные файлы. Как определить, какие файлы нельзя трогать при обновлении системы? Это нужно просто знать...

Несмотря на некоторые нововведения, как, например, каталог **/var** (в качестве места хранения системных данных), файлы систем UNIX и Linux все еще недостаточно упорядочены. Впрочем, для всего находится место. Особенно важно не менять стандартную структуру файлового дерева, поскольку программные пакеты и их инсталляционные

утилиты часто ищут те или иные файлы в строго определенных каталогах. А если вы попробуете усовершенствовать стандартную структуру, можете столкнуться с проблемами.

🕮 О конфигурировании ядра рассказывается в главе 13.

Корневая файловая система содержит корневой каталог и минимальный набор файлов и подкаталогов. Файл ядра находится в недрах корневой файловой системы, но не имеет стандартного имени или точного местоположения; в Solaris это даже не один файл, а, скорее, набор компонентов.

Частью корневой файловой системы являются также каталог **/etc** для критических системных файлов и файлов конфигурации, каталоги **/sbin** и **/bin** — для важных утилит и иногда каталог **/tmp** — для временных файлов. Каталог **/dev** — это обычно реальный каталог, который включен в корневую файловую систему, но он (частично или полностью) может перекрываться другими файловыми системами, если ваша система виртуализировала поддержку своих устройств. (Подробнее см. в разделе 13.2.)

Одни системы хранят совместно используемые библиотечные файлы и прочие важные программы (например, препроцессор языка C) в каталоге **/lib**. Другие переместили эти элементы в каталог **/usr/lib**, оставив для каталога **/lib** роль символьной ссылки.

🕮 В разделе 8.6 приводятся аргументы в пользу того, зачем разбивать диск на разделы, и рассказывается, как именно это следует делать.

Огромное значение имеют также каталоги **/usr** и **/var**. В первом хранится большинство стандартных программ и другие полезные компоненты, в частности интерактивная документация и библиотеки. Совсем не обязательно, чтобы каталог **/usr** был отдельной файловой системой, однако для удобства администрирования его, как правило, монтируют именно так. Для того чтобы система могла загрузиться в многопользовательском режиме, необходимы оба каталога — **/usr** и **/var**.

В каталоге **/var** содержатся буферные каталоги, журнальные файлы, учетная информация и прочие компоненты, специфичные для каждого компьютера. Поскольку при возникновении проблем журнальные файлы быстро разрастаются, рекомендуется помещать каталог **/var** в отдельную файловую систему.

Домашние каталоги пользователей чаще всего хранятся в отдельной файловой системе, которая обычно монтируется в корневом каталоге. Отдельные файловые системы можно использовать и для хранения больших информационных массивов, например библиотек исходных кодов программ и баз данных.

Наиболее важные стандартные каталоги перечислены в табл. 6.2.

Во многих системах `man`-страница `hier` (`man`-страница файловой системы в Solaris) содержит общие рекомендации по формированию файловой системы. Но не стоит ожидать, что реальная система во всех отношениях согласует свои действия с "генеральным планом". Для получения полезной информации по данной теме советуем также обратиться к странице Википедии "UNIX directory structure" (Структура каталогов UNIX).

При написании стандарта Filesystem Hierarchy Standard для систем Linux (`pathname.com/fhs`) предпринимается попытка систематизировать и унифицировать стандартные каталоги, а также пояснить их назначение. Это прекрасный информационный ресурс, с которым можно сверяться при возникновении вопросов, куда инсталлировать тот или иной компонент.

Дополнительные соображения, касающиеся иерархии локальных каталогов, приведены в разделе 12.10.

**Таблица 6.2. Стандартные каталоги и их содержимое**

| Каталог | ОС[а] | Содержимое |
|---|---|---|
| /bin | Все | Команды операционной системы ядра[б] |
| /boot | LS | Ядро и файлы для его загрузки |
| /dev | Все | Файлы устройств: дисков, принтеров, псевдотерминалов и т.д. |
| /etc | Все | Важные файлы запуска и конфигурации системы |
| /home | Все | Стандартные домашние каталоги пользователей |
| /kernel | S | Компоненты ядра |
| /lib | Все | Библиотеки, совместно используемые библиотеки и компоненты компилятора языка C |
| /media | LS | Точки монтирования файловых систем на съемных носителях |
| /mnt | LSA | Временные точки монтирования |
| /opt | Все | Программные пакеты необязательных приложений (которые пока не находят широкого применения) |
| /proc | LSA | Информация о всех выполняющихся процессах |
| /root | LS | Домашний каталог суперпользователя (часто просто /) |
| /sbin | Все | Команды, необходимые для обеспечения минимальной работоспособности системы[в] |
| /stand | H | Автономные утилиты, средства диагностики и форматирования дисков |
| /tmp | Все | Временные файлы, которые могут удаляться при перезагрузке |
| /usr | Все | Иерархия дополнительных файлов и программ |
| /usr/bin | Все | Большинство команд и исполняемых файлов |
| /usr/include | Все | Файлы заголовков, предназначенные для компиляции C-программ |
| /usr/lib | Все | Библиотеки и вспомогательные файлы для стандартных программ |
| /usr/lib64 | L | 64-разрядные библиотеки для 64-разрядных дистрибутивов Linux |
| /usr/local | Все | Локальные программы (программы, создаваемые или устанавливаемые локальным пользователем); отражает структуру каталога /usr |
| /usr/sbin | Все | Менее важные команды системного администрирования |
| /usr/share | Все | Элементы, общие для различных систем |
| /usr/share/man | Все | Страницы интерактивной документации |
| /usr/src | LSA | Исходные коды нелокальных программных пакетов (не находит широкого применения) |
| /usr/tmp | Все | Дополнительный каталог для временных файлов, которые могут сохраняться при перезагрузке |
| /var | Все | Системные данные и конфигурационные файлы |
| /var/adm | Все | Разное: журнальные файлы, записи об инсталляции системы, административные компоненты |
| /var/log | LSA | Системные журнальные файлы |
| /var/spool | Все | Буферные каталоги для принтеров, электронной почты и т.д. |
| /var/tmp | Все | Каталог для временного хранения файлов (после перезагрузки файлы не исчезают) |

[а] L = Linux, S = Solaris, H = HP-UX, A = AIX.

[б] В системах HP-UX и AIX каталог **/bin** служит символьной ссылкой на каталог **/usr/bin**.

[в] Отличительная особенность команд в каталоге **/sbin** обычно состоит в том, что они связаны со "статическими" версиями системных библиотек и, следовательно, не имеют много зависимостей от других частей системы.

## 6.4. Типы файлов

В большинстве реализаций файловых систем определены семь типов файлов:

- обычные файлы;
- каталоги;
- файлы байт-ориентированных (символьных) устройств;
- файлы блочно-ориентированных (блочных) устройств;
- локальные сокеты;
- именованные каналы (реализующие принцип обслуживания FIFO (First in First Out, т.е. "первым поступил — первым обслужен");
- символические ссылки.

Даже если разработчики добавляют в файловую систему что-нибудь новое и необычное (например, информацию о процессах в каталог **/proc**), им приходится маскировать это под файлы стандартных типов.

Определить тип существующего файла можно с помощью команды **ls -ld**. Первый символ в строке вывода обозначает тип объекта. В следующем примере выдается информация о каталоге **/usr/include**.

```
$ ls -ld /usr/include
  drwxr-xr-x   27 root   root   4096   Jul 15 20:57   /usr/include
```

Возможные коды для представления различных типов файлов перечислены в табл. 6.3.

**Таблица 6.3.Кодирование типов файлов в листинге команды ls**

| Тип файла | Символ | Создается командой | Удаляется командой |
|---|---|---|---|
| Обычный файл | - | Редакторы, cp и др. | rm |
| Каталог | d | mkdir | rmdir, rm -r |
| Файл символьного устройства | c | mknod | rm |
| Файл блочного устройства | b | mknod | rm |
| Локальный сокет | s | socket(2) | rm |
| Именованный канал | p | mknod | rm |
| Символическая ссылка | l | ln -s | rm |

Как видно из табл. 6.3, команда **rm** является универсальным средством удаления файлов. Но как удалить файл, имя которого, скажем, **-f**? В большинстве файловых систем это абсолютно корректное имя, но команда **rm -f** не сделает то, что нужно, поскольку выражение **-f** будет воспринято как флаг команды. Выйти из положения можно, либо указав более полное имя (например, **./-f**), либо воспользовавшись специальным аргументом **--**, который сообщит команде **rm** о том, что остальная часть командной строки представляет собой имя файла, а не список аргументов: **rm -- -f**.

Похожая проблема возникает с файлами, в именах которых присутствуют управляющие символы, поскольку такие имена трудно или вообще невозможно воспроизвести с помощью клавиатуры. В подобной ситуации нужно воспользоваться метасимволами интерпретатора команд, чтобы задавать не имя целиком, а лишь его шаблон. Указывайте также опцию **-i**, чтобы команда **rm** требовала подтвердить удаление каждого файла. Это позволит не удалить нужные файлы, соответствующие шаблону. Вот как, к примеру, можно удалить файл с оригинальным именем **foo**<Ctrl+D>**bar**.

```
$ ls
foo?bar foose kde-root
$ rm -i foo*
rm: remove 'foo\004bar'? y
rm: remove 'foose'? n
```

Команда **ls** отображает вместо управляющего символа знак вопроса, что иногда сбивает с толку[4]. Если забыть, что символ "?" тоже является подстановочным знаком интерпретатора, и попытаться выполнить команду **rm foo?bar**, можно удалить более одного файла (правда, не в данном примере). Ценность флага **-i** очень велика!

Если имена файлов совсем уж сложно набирать, прибегните к последнему средству.

```
rm -i *
```

Еще одна возможность удаления файлов с причудливыми именами — использование другого интерфейса файловой системы, такого как режим представления каталогов интерпретатора **emacs** или визуального средства, подобного Nautilus.

## Обычные файлы

Обычный файл — это просто последовательность байтов. Файловые системы не налагают ограничения на его структуру. Текстовые документы, файлы данных, программные файлы, библиотеки функций и многое другое — все это хранится в обычных файлах. К их содержимому возможен как последовательный, так и прямой доступ.

## Каталоги

Каталог хранит именованные ссылки на другие файлы. Он создается командой **mkdir** и удаляется (при условии, что он пуст) командой **rmdir**. Непустые каталоги можно удалить командой **rm -r**.

Специальные ссылки "." и ".." обозначают сам каталог и его родительский каталог соответственно. Такие ссылки нельзя удалить. Поскольку корневой каталог находится на вершине иерархии, ссылка ".." в нем эквивалентна ссылке ".".

Имя файла в действительности хранится в родительском каталоге, а не в самом файле. На файл можно ссылаться из нескольких каталогов одновременно и даже из нескольких элементов одного и того же каталога, причем у всех ссылок могут быть разные имена. Это создает иллюзию того, что файл одновременно присутствует в разных каталогах.

Эти дополнительные "жесткие" (фиксированные) ссылки (которые следует отличать от символических, или "мягких") можно считать синонимами для исходных файлов, и с точки зрения файловой системы все ссылки на файл эквивалентны. Файловая система подсчитывает количество ссылок на каждый файл и при удалении файла не освобождает блоки данных до тех пор, пока не будет удалена последняя ссылка на него. Ссылки не могут указывать на файл, находящийся в другой файловой системе.

Жесткие ссылки создаются командой **ln** и удаляются командой **rm**. Синтаксис команды **ln** легко запомнить, поскольку она является "зеркальным отражением" команды cp. Команда **cp oldfile newfile** создает копию файла **oldfile** с именем **newfile**, а команда **ln newfile oldfile** преобразует имя **newfile** в дополнительную ссылку на файл **oldfile**.

[4] Команда ***ls -b*** отображает специальные символы в виде восьмеричных значений, что может пригодиться, если их нужно отдельно идентифицировать. Так символ <Ctrl+A> отображается как "1" (\001 в восьмеричной системе), <Ctrl+B> — как "2" и т.д.

Жесткие ссылки можно создавать не только на файлы, но и на каталоги, но это делается довольно редко. Для того чтобы узнать, сколько ссылок существует на данный файл, используйте команду **ls -l**. (См. пример использования команды **ls** в разделе 6.5.)

Важно понимать, что жесткие ссылки не являются особым типом файлов, просто файловая система позволяет создавать несколько ссылок на один файл. Не только содержимое, но и атрибуты файла, в частности, права доступа и идентификатор владельца, являются общими для всех ссылок.

## Файлы символьных и блочных устройств

🕮 Устройства и драйверы более подробно рассматриваются в главе 13.

Файлы устройств позволяют программам получать доступ к аппаратным средствам и периферийному оборудованию системы. Ядро включает (или загружает) специальные программы (драйверы), которые во всех деталях "знают", как взаимодействовать с каждым из имеющихся устройств, поэтому само ядро может оставаться относительно абстрактным и независимым от оборудования.

Драйверы устройств образуют стандартный коммуникационный интерфейс, который воспринимается пользователем как совокупность обычных файлов. Получив запрос к файлу символьного или блочного устройства, файловая система передает этот запрос соответствующему драйверу. Важно отличать файлы устройств от драйверов этих устройств. Файлы сами по себе не являются драйверами.

Их можно рассматривать как шлюзы, через которые драйвер принимает запросы. Файлы символьных устройств позволяют связанным с ними драйверам выполнять собственную буферизацию ввода-вывода. Файлы блочных устройств обрабатываются драйверами, которые осуществляют ввод-вывод большими порциями, а буферизацию выполняет ядро. В прошлом некоторые типы аппаратных средств могли быть представлены файлами любого типа, но в современных системах такая конфигурация встречается редко.

Файлы устройств характеризуются двумя номерами: старшим и младшим. Старший номер устройства позволяет ядру определить, к какому драйверу относится файл, а младший номер, как правило, идентифицирует конкретное физическое устройство. Например, старший номер устройства 4 в Linux соответствует драйверу последовательного порта. Таким образом, первый последовательный порт (**/dev/tty0**) будет иметь старший номер 4 и младший номер 0.

Драйверы могут интерпретировать переданные им младшие номера устройств как угодно. Например, драйверы накопителей на магнитных лентах с помощью этого номера определяют, необходимо ли перемотать ленту после закрытия файла устройства.

В далеком прошлом **/dev** играл роль общего каталога, а файлы устройств, которые в нем хранились, создавались с помощью команды **mknod** и удалялись командой **rm**. Стандартизировать работу по созданию файлов устройств помогал сценарий с именем **MAKEDEV**.

К сожалению, эта "сырая" система плохо справлялась с безбрежным морем драйверов и типов устройств, которые появились в последние десятиления. Кроме того, она способствовала возникновению разного рода потенциальных конфигурационных нестыковок: например, файлы устройств ссылались на несуществующие устройства, устройства оказывались недоступными, поскольку они не имели файлов устройств, и т.д.

В наши дни в большинстве систем реализована некоторая форма автоматического управления файлами устройств, которая позволяет системе играть более активную роль в конфигурировании собственных файлов устройств. Например, в Solaris каталоги **/dev** и **/devices** полностью виртуализированы. В дистрибутивах Linux каталог **/dev** являет-

ся стандартным, но управлением файлами внутри него занимается демон **udevd**. (Демон **udevd** создает и удаляет файлы устройств в ответ на изменения в оборудовании, о которых сообщает ядро.) Подробнее о методах решения этой задачи в разных системах написано в главе 13.

## Локальные сокеты

Установленные посредством сокетов соединения позволяют процессам взаимодействовать, не подвергаясь влиянию других процессов. В системе UNIX поддерживается несколько видов сокетов, использование которых, как правило, предполагает наличие сети. Локальные сокеты доступны только на локальном компьютере, и обращение к ним осуществляется через специальные объекты файловой системы, а не через сетевые порты. Иногда такие сокеты называют UNIX-сокетами.

🕮 Подробнее о системе Syslog рассказывается в главе 11.

Несмотря на то что другие процессы распознают файлы сокетов как элементы каталога, только процессы, между которыми установлено соответствующее соединение, могут осуществлять над файлом сокета операции чтения и записи. В качестве примеров стандартных средств, использующих локальные сокеты, можно назвать системы X Window и Syslog.

Локальные сокеты создаются с помощью системного вызова **socket**. Когда с обеих сторон соединение закрыто, сокет можно удалить командой **rm** или с помощью системного вызова **unlink**.

## Именованные каналы

Подобно локальным сокетам, именованные каналы обеспечивают взаимодействие двух процессов, выполняемых на одном компьютере. Такие каналы еще называют файлами FIFO (First In, First Out — "первым поступил, первым обслужен"). Они создаются командой **mknod** и удаляются командой **rm**.

Как и в случае локальных сокетов, реальные экземпляры именованных каналов весьма немногочисленны и нечасто встречаются. Они редко требуют административного вмешательства[5].

Именованные каналы и локальные сокеты имеют практически одинаковое назначение, а их обоюдное существование сложилось исторически. Если бы системы UNIX и Linux разрабатывались в наши дни, то об этих средствах взаимодействия вопрос бы не стоял; сейчас их заменили бы сетевые сокеты.

## Символические ссылки

Символическая, или "мягкая", ссылка позволяет вместо имени файла указывать его псевдоним. Столкнувшись при поиске файла с символической ссылкой, ядро извлекает хранящееся в ней имя. Разница между жесткими и символическими ссылками состоит в том, что жесткая ссылка является прямой, т.е. указывает непосредственно на индексный дескриптор файла, тогда как символическая ссылка указывает на файл по имени. Файл, адресуемый символической ссылкой, и сама ссылка представляют собой разные объекты файловой системы.

[5] По словам нашего рецензента, "программа Nagios (см. раздел 21.12) использует именованные каналы, и иногда ей необходима помощь".

Символические ссылки создаются командой **ln -s** и удаляются командой **rm**. Они могут содержать произвольное имя, т.е. разрешается указывать на файлы, хранящиеся в других файловых системах, и даже на несуществующие файлы. Иногда несколько символических ссылок образуют петлю.

Символическая ссылка может хранить как полное, так и сокращенное имя файла. Например, команда

```
$ sudo ln -s archived/secure /var/log/secure
```

посредством относительного пути связывает имя **/var/log/secure** с именем **/var/log/archived/secure**. Как видно из следующего результата выполнения команды **ls**, полученная символическая ссылка будет содержать строку "archived/secure".

```
$ ls -l /var/log/secure
lrwxrwxrwx 1 root root 18 2009-07-05 12:54 /var/log/secure -> archived/secure[6]
```

Каталог **/var/log** можно переместить куда угодно, но символическая ссылка останется корректной.

Часто ошибочно думают, будто первый аргумент команды **ln -s** должен преобразовываться с учетом текущего каталога. На самом деле он не раскрывается командой **ln** (как имя файла), а просто записывается в виде литеральной строки, которая становится объектом символической ссылки.

# 6.5. Атрибуты файлов

В традиционной модели файловой системы UNIX и Linux каждому файлу соответствует набор из девяти битов режима. Они определяют, какие пользователи имеют право читать файл, записывать в него данные или запускать его на выполнение. Вместе с другими тремя битами, которые в основном влияют на работу исполняемых файлов, этот набор образует код, или режим, доступа к файлу.

Двенадцать битов режима хранятся вместе с четырьмя дополнительными битами, определяющими тип файла. Эти четыре бита устанавливаются при создании файла и не подлежат изменению. Биты режима могут изменяться владельцем файла или суперпользователем с помощью команды **chmod** ("change mode" — изменить режим). Просмотр значений этих битов осуществляется с помощью команды **ls -l** (или **ls -ld** в случае каталога). Пример приведен далее в этой главе.

## Биты режима

Девять битов режима определяют, кто и какие операции может выполнять над файлом. Традиционная система UNIX не позволяет устанавливать режим на уровне отдельного пользователя (хотя теперь списки управления доступом поддерживаются во всех основных системах). Подробнее это описано в разделе 6.6. Вместо этого три различных набора битов определяют права доступа, предоставляемые владельцу файла, членам группы, которой принадлежит файл, и прочим пользователям (в таком порядке). Каждый набор состоит из трех битов: бита чтения, записи и выполнения.

[6] Права доступа к файлу, отображаемые командой **ls** для символической ссылки *lrwxrwxrwx*, представляют собой фиктивные значения. Права на создание, удаление или разрешение ссылки управляются содержащим ее каталогом, в то время как права чтения, записи или выполнения для файла, указываемого ссылкой, предоставляются собственными правами целевого файла. Следовательно, символическая ссылка не нуждается (и не обладает) ни в какой собственной информации о правах доступа.

Код доступа удобно записывать в виде восьмеричного числа, так как каждая цифра в нем представляется тремя битами. Три старших бита (в коде доступа им соответствуют восьмеричные значения 400, 200 и 100) служат для управления доступом к файлу его владельца. Вторые три бита (40, 20 и 10) задают доступ для членов группы. Последние три бита (4, 2 и 1) определяют доступ к файлу остальных пользователей. Старший бит каждой триады — это бит чтения, средний — бит записи, младший — бит выполнения.

Каждый пользователь попадает только в одну из категорий прав доступа. В случае неоднозначности выбираются самые строгие права. Например, права владельца файла всегда определяются триадой битов владельца и никогда — битами группы. Возможна ситуация, когда другие пользователи имеют больше прав, чем владелец, но такая конфигурация используется редко.

Для обычного файла бит чтения означает возможность открывать и читать файл. Бит записи позволяет изменять содержимое файла, в том числе полностью очищать его. Правом удаления и переименования файла управляют биты, заданные для его родительского каталога, поскольку именно там хранится имя файла.

Установкой бита выполнения задается разрешение запускать файл. Существует два типа исполняемых файлов: бинарные файлы, которые выполняются непосредственно центральным процессором, и сценарии, обрабатываемые интерпретатором команд или какой-нибудь другой аналогичной программой. В соответствии с принятыми соглашениями, сценарии начинаются со строки примерно такого вида.

```
#!/usr/bin/perl
```

Здесь задается соответствующий интерпретатор. Текстовые исполняемые файлы, в которых не указан конкретный интерпретатор, считаются сценариями **bash** или **sh**[7].

Будучи установленным для каталога, бит выполнения (в этом контексте его часто называют *битом поиска*) разрешает переходить в каталог или проходить через него при доступе к файлу, но без получения списка содержимого каталога. Последнее (т.е. получение списка) становится возможным, если для каталога задана комбинация битов чтения и выполнения. Комбинация битов записи и выполнения позволяет создавать, удалять и переименовывать файлы в данном каталоге.

Традиционную девятибитовую модель режимов способны усложнить и переопределить многие такие расширения, как списки управления доступом (см. раздел 6.6), система SELinux, т.е. Linux с улучшенной безопасностью (см. раздел 22.9), и "бонусные" биты режимов, определяемые отдельными файловыми системами (см. последний подраздел данного раздела). Если окажется, что вы не можете объяснить поведение своей файловой системы, проверьте, оказывает ли на нее влияние один из перечисленных выше факторов.

## Биты setuid и setgid

Биты, которым в коде доступа соответствуют восьмеричные значения 4000 и 2000, — это биты смены идентификатора пользователя (`setuid`) и идентификатора группы (`setgid`). Если эти биты установлены для исполняемых файлов, они позволяют программам получать доступ к файлам и процессам, которые при прочих обстоятельствах недоступны пользователю, выполняющему эти программы. Подробнее этот механизм был описан в разделе 4.1.

[7] Когда файл начинается с символов *#!*, он в первую очередь обрабатывается ядром. Но если интерпретатор команд не указан или указан неправильно, ядро откажется выполнять файл. В этом случае текущий интерпретатор предпримет вторую попытку, попытавшись выполнить сценарий в интерпретаторе *sh*.

Если бит setgid установлен для каталога, то создаваемые в нем файлы будут принимать идентификатор группы каталога, а не группы, в которую входит владелец файла. Это упрощает пользователям, принадлежащим к одной группе, совместный доступ к каталогам. Следует также учитывать, что для исполняемого файла бит setgid имеет другое значение.

Бит setgid можно устанавливать для файлов, не являющихся исполняемыми. Тем самым запрашивается специальный режим блокирования файлов при их открытии. Правда, нам не известны случаи использования этой возможности.

## Дополнительный бит

Бит, которому в коде доступа соответствует восьмеричное значение 1000, называется *дополнительным* (sticky bit). В первых UNIX-системах дополнительный бит запрещал выгрузку программ из памяти. Сегодня он утратил свое значение и, попросту, игнорируется.

Если дополнительный бит установлен для каталога, то файловая система позволит удалять и переименовывать его файлы только владельцу каталога, владельцу файла или суперпользователю. Иметь одно лишь право записи в каталог недостаточно. Такая мера позволяет несколько лучше защитить каталоги вроде **/tmp**.


Системы Solaris и HP-UX менее строги в обработке дополнительных каталогов: вы можете удалить файл в дополнительном каталоге, если у вас есть на него право записи (даже если вы не являетесь его владельцем). В этом действительно есть здравый смысл, но об этом отличительном нюансе не стоит забывать.

## Команда ls: просмотр атрибутов файла

Файловая система хранит для каждого файла около сорока информационных полей; большая часть из них используется самой файловой системой. Администратора, в основном, интересует количество жестких ссылок, владелец, группа, код доступа, время последнего обращения и последней модификации, размер и тип файла. Всю эту информацию можно получить с помощью команды **ls -l** (или команды **ls -ld** в случае каталога; без флага **-d** команда **ls** перечисляет содержимое каталога).

Для каждого файла хранится также время последнего изменения атрибутов. Традиционное название этого поля ("ctime" от "change time" — время изменения) многих вводит в заблуждение, так как они полагают, что это время создания файла. На самом деле здесь зафиксировано время последнего изменения одного из атрибутов файла (владелец, код доступа и так далее), но не его содержимого.

Рассмотрим пример.

```
$ ls -l /bin/gzip
-rwxr-xr-x 3 root root 62100 May 28 2010 /bin/gzip
```

В первом поле задается тип файла и режим доступа к нему. Поскольку первый символ — дефис, значит, перед нами обычный файл (см. табл. 6.3).

Следующие девять символов — это три набора битов режима. Порядок наборов таков: владелец, группа, другие пользователи. В листинге команды **ls** биты режима представляются буквами r, w и x (чтение, запись и выполнение). В показанном примере владелец имеет все права доступа к файлу, а остальные пользователи — только право на чтение и выполнение.

Если бы был установлен бит setuid, то вместо буквы x, обозначающей право владельца на выполнение, стояла бы буква s. В случае бита setgid вместо буквы x для группы тоже была бы указана буква s. Последний бит режима (право выполнения для других пользователей) представляется буквой t, когда для файла задан sticky-бит. Если бит setuid/setgid или sticky-бит установлен, а надлежащий бит выполнения — нет, эти биты представляются символами S и T, что указывает на игнорирование данных атрибутов вследствие ошибки.

Второе поле листинга представляет собой счетчик ссылок на файл. В данном случае здесь стоит цифра 3, свидетельствующая о том, что **/bin/gzip** — одно из трех имен файла (два других — **/bin/gunzip** и **/bin/zcat**). Каждый раз при создании жесткой ссылки на файл этот счетчик увеличивается на единицу. Символические ссылки в счетчике не учитываются.

Любой каталог имеет минимум две жесткие ссылки: из родительского каталога и из специального файла "." внутри самого каталога.

Следующие два поля определяют владельца и группу файла. В данном примере файл принадлежит пользователю root и одноименной группе. В действительности ядро хранит эти данные не в строковом виде, а в виде идентификаторов. Если символьные версии идентификаторов определить невозможно, в этих полях будут отображаться числа. Так происходит, когда запись пользователя или группы была удалена из файла, соответственно, **/etc/passwd** или **/etc/group**. Не исключено также, что возникла ошибка в базе данных NIS или LDAP (описана в главе 19), если она используется.

В следующем поле отображается размер файла в байтах. Рассматриваемый файл имеет размер 62100 байт. Далее указана дата последней модификации файла: 28 мая 2010 г. В последнем поле листинга приведено имя файла: **/bin/gzip**.

Для файла устройства команда ls выдает несколько иную информацию.

```
$ ls -l /dev/tty0
crw-rw---- 1   root   root   4, 0   Jun 11   20:41   /dev/tty0
```

Большинство полей те же, но вместо размера в байтах показаны старший и младший номера устройства. Имя /dev/tty0 относится в данной системе (Red Hat) к первой виртуальной консоли, управляемой драйвером устройства 4 (драйвер терминала).

При поиске жестких ссылок может пригодиться команда ls -i, отображающая для каждого файла номер его индексного дескриптора. Не вдаваясь в детали реализации файловой системы, скажем, что этот номер представляет собой индекс таблицы, в которой перечислены все файлы системы. Именно на индексные дескрипторы ссылаются файловые записи каталогов. Жесткие ссылки, указывающие на один и тот же файл, будут иметь одинаковый номер. Для того чтобы составить представление о "паутине" ссылок, воспользуйтесь командой **ls -li** для определения числа ссылок и номера индексного дескриптора какого-нибудь файла, а затем поищите его "двойников" с помощью команды **find**[8].

Другими важными опциями команды **ls** являются **-a**, отображающая все записи в каталоге (даже те файлы, имена которых начинаются с точки), **-t**, выполняющая сортировку файлов по времени изменения (или **-tr**, осуществляющая сортировку в обратном хронологическом порядке), **-F**, отображающая имена файлов с выделением каталогов и исполняемых файлов, **-R**, выводящая рекурсивный список, и **-h**, которая отображает размеры файлов в удобной для человеческого восприятия форме (например, 8K или 53M).

[8] Выполните команду **find** *точка_монтирования* **-xdev -inum** *индексный_дескриптор* **-print**

## Команда chmod: изменение прав доступа

Код доступа к файлу можно изменить с помощью команды **chmod**. Такое право есть лишь у владельца файла и пользователя `root`. В первых UNIX-системах код доступа задавался в виде восьмеричного числа. В современных версиях поддерживается также система мнемонических обозначений. Первый способ удобнее для системного администратора, но при этом можно задать только абсолютное значение кода доступа. В случае использования мнемонического синтаксиса разрешается сбрасывать и устанавливать отдельные биты режима.

Первым аргументом команды **chmod** является спецификация прав доступа. Второй и последующий аргументы — это имена файлов, права доступа к которым подлежат изменению. При использовании восьмеричной формы записи первая цифра относится к владельцу, вторая — к группе, а третья — к другим пользователям. Если необходимо задать биты `setuid/setgid` или дополнительный бит, следует указывать не три, а четыре восьмеричные цифры: первая цифра в этом случае будет соответствовать трем специальным битам.

В табл. 6.4 показано восемь возможных комбинаций для каждого трехбитового набора, где символы `r`, `w` и `x` обозначают право чтения, записи и выполнения соответственно.

**Таблица 6.4. Коды доступа в команде chmod**

| Восьмеричное число | Двоичное число | Режим доступа | Восьмеричное число | Двоичное число | Режим доступа |
|---|---|---|---|---|---|
| 0 | 000 | `---` | 4 | 100 | `r--` |
| 1 | 001 | `--x` | 5 | 101 | `r-x` |
| 2 | 010 | `-w-` | 6 | 110 | `rw-` |
| 3 | 011 | `-wx` | 7 | 111 | `rwx` |

Например, команда **chmod 711 myprog** предоставляет владельцу все права, а остальным пользователям — только право выполнения[9].

При использовании мнемонического синтаксиса вы объединяете множество исполнителей (`u` — пользователь, `g` — группа или `o` — другой) с оператором (`+` — добавить, `-` — удалить и `=` — присвоить) и набором прав доступа. Более подробное описание мнемонического синтаксиса можно найти на `man`-странице команды `chmod`, но синтаксис всегда лучше изучать на примерах. В табл. 6.5 приведено несколько примеров мнемонических спецификаций.

**Таблица 6.5. Примеры мнемонических спецификаций команды chmod**

| Спецификация | Назначение |
|---|---|
| **u+w** | Владельцу файла дополнительно дается право записи |
| **ug=rw,o=r** | Владельцу и членам группы дается право чтения/записи, остальным пользователям — право чтения |
| **a-x** | У всех пользователей отбирается право выполнения |
| **ug=srx,o=** | Владельцу и членам группы дается право чтения/выполнения, устанавливаются также биты SUID и SGID; остальным пользователям запрещается доступ к файлу |
| **g=u** | Членам группы предоставляются такие же права, что и владельцу |

[9] Если файл `myprog` является сценарием интерпретатора команд, должно быть задано также право чтения для остальных пользователей. Файлы сценариев, запускаемые интерпретатором, открываются и читаются в текстовом виде, а бинарные файлы выполняются непосредственно ядром, поэтому для них не нужно задавать право чтения.

Символ `u` ("*u*ser") обозначает владельца файла, `g` ("*g*roup") — группу, `o` ("*o*thers") — других пользователей, `a` ("*a*ll") — всех пользователей.

В системах Linux и OpenSolaris у существующего файла можно "заимствовать" права доступа. Например, при выполнении команды **`chmod --reference=filea fileb`** код доступа для файла **`fileb`** будет установлен таким же, как у файла **`filea`**.

При наличии опции **`-R`** команда **`chmod`** будет рекурсивно обновлять права доступа ко всем файлам указанного каталога и его подкаталогов. Здесь удобнее всего придерживаться мнемонического синтаксиса, чтобы менялись только те биты, которые заданы явно. Например, команда

```
$ chmod -R g+w mydir
```

добавляет групповое право записи к каталогу **`mydir`** и его содержимому, не затрагивая остальные права.

При установке битов выполнения будьте осторожны с выполнением команды **`chmod -R`**. Ведь применительно к каталогу и файлу бит выполнения интерпретируется по-разному. Поэтому вряд ли реальный результат выполнения команды **`chmod -R a-x`** будет соответствовать ожидаемому вами.

## Команды chown и chgrp: смена владельца и группы

Команды **`chown`** и **`chgrp`** предназначены для изменения владельца файла и группы соответственно. Синтаксис этих команд подобен синтаксису команды **`chmod`**, за исключением того, что первый аргумент содержит нового владельца или группу соответственно.

Для того чтобы изменить группу файла, необходимо либо быть владельцем файла и входить в назначаемую группу, либо быть пользователем `root`. Правила изменения владельца в целом усложнились и зависят от конкретной системы. В большинстве систем предусмотрены средства настройки поведения команды **`chown`** в зависимости от выполняемого процесса.

Как и команда **`chmod`**, команды **`chown`** и **`chgrp`** имеют флаг **`-R`**, который задает смену владельца или группы не только самого каталога, но и всех его подкаталогов и файлов. Например, последовательность команд

```
$ sudo chown -R matt ~matt/restore
$ sudo chgrp -R staff ~matt/restore
```

можно применить для восстановления из резервной копии владельца и группы файлов для пользователя `matt`. Если вы устанавливаете домашний (исходный) каталог пользователя, не следует пытаться выполнять команду `chown` для файлов, имена которых начинаются с точки.

```
$ sudo chown -R matt ~matt/.*
```

Дело в том, что указанному шаблону соответствует также файл **`~matt/..`**, поэтому будет изменен владелец родительского каталога и, возможно, домашних каталогов других пользователей.

Команда **`chown`** может изменить владельца файла и группу одновременно с помощью такого синтаксиса.

**`chown`** *пользователь*`:`*группа файл* ...

Например:

```
$ sudo chown -R matt:staff ~matt/restore
```

В системах Linux и Solaris этот синтаксис позволяет опустить либо элемент *пользователь*, либо элемент *группа*, что делает команду **chgrp** ненужной. Если вы включите в синтаксис команды двоеточие, но не укажете элемент *группа*, в команде **chown** будет использована стандартная группа пользователя.

## Команда umask: задание стандартных прав доступа

Встроенная команда **umask** позволяет менять стандартный режим доступа к создаваемым файлам. Каждый процесс имеет собственный атрибут **umask**; встроенная в интерпретатор команда **umask** устанавливает собственное значение **umask**, которое затем наследуется выполняемыми вами командами.

Значение **umask** задается в виде трехзначного восьмеричного числа, которое соответствует аннулируемым правам доступа. При создании файла код доступа к нему устанавливается равным разнице между величиной, которую запрашивает создающая файл программа, и значением **umask**. В табл. 6.6 перечислены возможные комбинации битов режима для команды `umask`.

**Таблица 6.6. Схема кодирования значения umask**

| Восьмеричное число | Двоичное число | Режим доступа | Восьмеричное число | Двоичное число | Режим доступа |
|---|---|---|---|---|---|
| 0 | 000 | rwx | 4 | 100 | -wx |
| 1 | 001 | rw- | 5 | 101 | -w- |
| 2 | 010 | r-x | 6 | 110 | --x |
| 3 | 011 | r-- | 7 | 111 | --- |

Например, команда **umask 027** предоставляет все права владельцу файла, запрещает читать файл членам группы и не дает никаких прав другим пользователям. Стандартное значение **umask** равно, как правило, 022, т.е. право модификации файла предоставляется только его владельцу.

О стандартных конфигурационных сценариях речь идет в главе 7.

Нельзя заставить пользователей придерживаться конкретного значения **umask**, так как они в любой момент могут изменить его. Можно, однако, задать стандартное значение **umask** в тех копиях файла **.profile**, которые предоставляются новым пользователям системы.

## Дополнительные флаги в системе Linux

В файловых системах `ext2`, `ext3` и `ext4` определен ряд вспомогательных флагов, устанавливая которые можно запрашивать особые режимы обработки файлов (здесь важно подчеркнуть слово "запрашивать", поскольку многие флаги еще не реализованы). Например, один флаг делает файл доступным только для дополнения, а другой — недоступным для изменения и удаления.

В связи с тем что эти флаги являются специфической особенностью файловых систем серии `ext*`, в Linux для их просмотра и изменения предназначены специальные

команды `lsattr` и `chattr`. В табл. 6.7 перечислены действительно реализованные флаги (в настоящее время в интерактивной документации описаны только около половины этих флагов).

**Таблица 6.7. Дополнительные флаги файловых систем `ext2` и `ext3`**

| Флаг | Назначение |
|---|---|
| `A` | Никогда не обновлять время доступа (в целях повышения производительности) |
| `a` | Разрешить запись в файл только в режиме добавления (может устанавливаться лишь суперпользователем) |
| `D` | Включить режим синхронной записи изменений каталога |
| `d` | Не создавать резервные копии (команда `dump` будет игнорировать такой файл) |
| `i` | Сделать файл неизменяемым и неудаляемым (может устанавливаться лишь суперпользователем) |
| `S` | Включить режим синхронной записи изменений (буферизация отсутствует) |

Не вполне ясно, имеют ли остальные флаги, за исключением `d`, практическую ценность. Флаги `a` и `i` изначально задумывались как дополнительное средство защиты системы от хакеров и злонамеренных программ. К сожалению, они лишь мешают работе полезных программ, отпугивая только тех хакеров, которые не знают о существовании команды `chattr -ia`. Исследования показали, что эти флаги чаще используются самими хакерами, чем для защиты от них.

Особого внимания заслуживают флаги `S` и `D`. Поскольку при внесении изменений они заставляют немедленно записывать на диск все страницы памяти, связанные с файлом или каталогом, может показаться, будто это служит дополнительной защитой данных на случай сбоя. На самом же деле порядок выполнения операций, связанных с синхронными обновлениями, весьма необычен и, как стало известно, мешает работе команды `fsck`. В результате процедура восстановления поврежденной файловой системы усложняется, вместо того чтобы стать более надежной. Обычно лучше использовать возможность ведения журнала файловой системы, поддерживаемую `ext3` и `ext4`. Опция `j` может вызвать запись в журнал информации об определенных файлах (правда, за счет некоторого снижения производительности системы).

## 6.6. Списки контроля доступа

Традиционная девятибитовая система контроля доступа (access control system — ACL) "владелец/группа/другие" позволяет решать большинство задач администрирования. Хотя ей присущи явные ограничения, она хорошо согласуется с традициями (кое-кто может сказать "с прошлыми традициями") простоты и предсказуемости UNIX.

Практически все другие операционные системы используют значительно более сложный метод управления доступом к файлам: списки управления доступом или ACL. Каждому файлу или каталогу может соответствовать список ACL, в котором перечислены правила установки прав доступа к данному файлу или каталогу. Каждое правило в списке ACL называется *записью управления доступом* (access control entry — ACE).

В общем, запись контроля доступа идентифицирует пользователя или группу, к которой она применяется, и определяет набор привилегий, которыми наделяются эти пользователи. Списки ACL не имеют фиксированной длины и могут содержать определения прав множества пользователей или групп. Большинство операционных систем ограни-

чивает длину отдельного списка ACL, но это ограничение может быть довольно слабым (обычно 32 записи), которое достигается не часто.

Более сложные системы позволяют администраторам указывать частичные наборы прав или отрицательные права, а некоторые из систем поддерживают наследование, которое позволяет определять права доступа для передачи их вновь создаваемым элементам файловых систем.

Конечно, системы ACL предоставляют больше возможностей, чем традиционная модель UNIX, но они и сложнее ее на порядок как для администраторов, так и для разработчиков программ. Используйте их с большой осторожностью. Они не только сложнее в использовании, но и могут порождать проблемы при использовании в сочетании с "не имеющими понятия" об ACL-списках системами резервного копирования, узлами NFS и даже такими простыми программами, как текстовые редакторы.

Списки ACL тяготеют к беспорядку, и поэтому со временем они становятся непригодными для управления.

## Краткий обзор развития UNIX-списков ACL

В следующих разделах рассмотрим различные системы ACL, которые поддерживаются в UNIX и Linux, а также наборы команд, предназначенные для управления ими. Но сначала необходимо ответить на вопрос: "Как эти списки ACL превратились в катастрофу?"

Обычно объект рассматривается как история политики, денег и ветвлений. В данном случае базовое понимание истории помогает наложить на реальность некоторую структуру.

Подкомитет POSIX впервые начал работать над ACL для UNIX в средине 1990-х годов. В первом приближении модель POSIX ACL просто расширила традиционную UNIX-систему прав доступа (`rwx`), чтобы удовлетворить потребности в привилегиях различных пользователей и групп.

К сожалению, проект POSIX не стал официальным стандартом, и рабочая группа была расформирована в 1998 году. Тем не менее некоторые компании реализовали системы POSIX ACL "на свой вкус". Нашлись и такие компании, которые создали собственные системы ACL. Но поскольку в этой области лидер четко не обозначился, все реализации выглядели по-разному.

Тем временем для систем UNIX и Linux стало практически нормой использовать файловые системы совместно с Windows, которая имеет собственные ACL-соглашения. Здесь напряжение в нашей истории возрастает, поскольку Windows создает множество функций, которых нет ни в традиционной модели UNIX, ни в ее эквиваленте POSIX ACL. Кроме того, списки ACL в системе Windows семантически более сложные; например, они позволяют использовать отрицательные права (записи с "отказом" в правах) и предусматривают усложненную схему наследования.

🕮 Подробнее об NFS написано в главе 18.

Архитекторы-разработчики версии 4 протокола NFS — стандартного UNIX-протокола по совместному использованию файлов — стремились включить в свое детище системы ACL как элемент первостепенной важности. Из-за "разногласий" между UNIX и Windows и несовместимости между реализациями UNIX ACL стало очевидно, что на стороне соединения NFSv4 могут оказаться системы совершенно разных типов. Все они могут понимать "язык" NFSv4 ACL, POSIX ACL, Windows ACL или совсем не понимать системы ACL. Стандарт NFSv4 должен иметь возможность взаимодействовать с различными мирами без особых сюрпризов и проблем с безопасностью.

В таких условиях неудивительно, что системы NFSv4 ACL являются, по сути, объединением всех ранее существовавших систем. Они представляют собой строгое супермножество систем POSIX ACL, поэтому любая система POSIX ACL может быть представлена как NFSv4 ACL без потери информации. В то же время системы NFSv4 ACL обеспечивают все биты прав доступа, которые предусмотрены в системах Windows, а также имеют большинство семантических свойств Windows.

## Реализация списков ACL

Теоретически ответственность за поддержку и функционирование систем ACL можно было бы переложить на ряд других компонентов операционной системы. Системы ACL могли бы быть реализованы ядром от имени всех файловых систем, отдельными файловыми системами или, может быть, такими высокоуровневыми программами, как серверы NFS и CIFS.

На практике только файловые системы могут реализовать списки ACL ясно, надежно и с приемлемой производительностью. Следовательно, поддержка ACL зависит от операционной системы (ОС) и файловой системы. Файловая система, которая поддерживает списки ACL в одной ОС, может не поддерживать их в другой или может предусматривать несколько другую реализацию, поддерживаемую другими командами.

В UNIX стандартные системные вызовы, которые обеспечивают управление файлами (**open**, **read**, **unlink** и пр.), не создают никаких возможностей для поддержки ACL. Тем не менее они продолжают прекрасно работать в системах со списками ACL благодаря тому, что базовые файловые системы выполняют собственные действия по контролю прав доступа. Операции, которые не разрешены релевантной системой ACL, просто не выполняются и возвращают общий код ошибки в виде "отсутствия прав доступа".

ACL-ориентированные программы для чтения или установки файловых списков ACL используют отдельный системный вызов или библиотечную утилиту. Если в операционную систему добавляется ACL-поддержка, то обычно модернизируются такие распространенные утилиты, как **ls** и **cp**, чтобы обеспечить хотя бы минимальную поддержку ACL (например, с помощью команды **cp -p**, которая должна хранить списки ACL, если таковые имеются). Кроме того, система должна добавлять новые команды или командные расширения, которые бы разрешили пользователям читать и устанавливать списки ACL из командной строки. К сожалению, эти команды не стандартизированы в операционных системах.

Поскольку реализации списков ACL зависят от конкретной файловой системы и операционные системы поддерживают несколько реализаций файловых систем, во многих системах предусматривается поддержка нескольких типов списков ACL. Даже отдельно взятая файловая система может предложить несколько вариантов ACL, как в системе JFS2 (IBM). Если возможно использование нескольких систем ACL, команды по управлению ими могут быть одинаковыми или различными — все зависит от конкретной системы.

## Системная поддержка списков ACL

Рассмотрим особенности поддержки списков ACL в разных системах UNIX и Linux, поскольку трудно описать эту область общими словами.

- На момент написания этого раздела (2010 г.) POSIX-ориентированные ACL-системы играют ведущую роль по реализации и использованию, но NFSv4 ACL прогрессируют особенно быстро и, по всей вероятности, станут стандартом де-факто.

В настоящее время только системы ZFS (компания Sun) и JFS2 (IBM) обладают собственной поддержкой NFS4v4 ACL.

В Linux ACL-списки в POSIX-стиле поддерживаются файловыми системами ReiserFS, XFS, JFS, Btrfs и семейством `ext*`. Обычно они отключены по умолчанию, но с помощью команды **`mount`** с опцией **`-o acl`** их можно включить. Управлять записями POSIX ACL позволяют команды **`getfacl`** и **`setfacl`**.

Системы Solaris поддерживают POSIX ACL на основе более старой файловой системы UFS и NFSv4 ACL на базе системы ZFS. Версии Solaris-команд **`ls`** и **`chmod`** модифицированы для отображения и редактирования обоих типов списков ACL[10]. В системе Solaris предусмотрены команды **`setfacl`** и **`getfacl`**, которые отчасти подобны используемым в дистрибутивах Linux, но в действительности они служат только для совместимости и работают лишь для POSIX ACL.

В операционной системе HP-UX разработана собственная подсистема ACL для собственной файловой системы HFS (High-performance File System — высокопроизводительная файловая система). Когда компания HP приняла VxFS (Veritas) в качестве своей главной файловой системы, она также включила поддержку ACL в стиле POSIX[11]. К сожалению, эти две системы ACL управляются различными наборами команд. Система HFS в настоящее время не имеет большого практического значения, но команды HFS ACL остаются в силе для совместимости. В этой книге системы HFS ACL не рассматриваются.

В AIX файловая система JFS2 поддерживает собственную систему ACL, известную как AIXC. Как и AIX 5.3.0, система JFS2 также поддерживает списки ACL в стиле NFSv4. В AIX используются одинаковые команды (**`aclget`**, **`aclput`** и **`acledit`**) для управления обоими типами списков ACL, а для упрощения перехода из одного формата в другой используется утилита **`aclconvert`**. В этой книге системы AIXC не рассматриваются.

## Обзор POSIX ACL

Списки POSIX ACL поддерживаются во многих файловых системах Linux и в порте файловой системы VxFS (компания HP-UX), именуемой JFS. Они также присутствуют в системе Solaris только для файловой системы UFS.

Списки POSIX ACL Linux представляют собой практически простое расширение стандартной 9-битовой UNIX-модели прав доступа. Система поддерживает только права чтения, записи и выполнения. Такие дополнительные атрибуты, как `setuid`- и дополнительные биты, обрабатываются исключительно традиционными битами определения режима.

Списки ACL позволяют устанавливать биты `rwx` независимо для любой комбинации пользователей и групп. Возможный вид отдельных записей в ACL приведен в табл. 6.8.

[10] Убедитесь, что в вашей переменной среды *PATH* путь **`/bin`** указан перед путем **`/usr/gnu/bin`**, что позволит вам получить именно Solaris-версии команд ***`ls`*** и ***`chown`***, а не GNU-версии.

[11] Для того чтобы дезориентировать своих клиентов и держать их в покорном неведении, компания HP использует стратегию "похищения" имен существующих файловых систем, применив их к запатентованным продуктам. Например, файловая система HFS компании HP была названа так специально, чтобы ее можно было легко спутать с файловой системой Hierarchical File System компании Apple, также именуемой HFS. Компания HP для своего порта VxFS выбирает название "JFS", чтобы пользователям было трудно отличить его от имени JFS, которое выбрала компания IBM для собственной файловой системы.

**Таблица 6.8. Записи, которые могут встретиться в списке управления доступом**

| Формат | Пример | Определяет права доступа для |
|---|---|---|
| `user::`*`права_доступа`* | `user::rw-` | владельца файла |
| `user:`*`имя_пользователя`*`:`*`права_доступа`* | `user:trent:rw-` | конкретного пользователя |
| `group::`*`права_доступа`* | `group::r-x` | группы, которая владеет файлом |
| `group:`*`имя_группы`*`:`*`права_доступа`* | `group:staff:rw-` | конкретной группы |
| `other::`*`права_доступа`* | `other::---` | всех других пользователей |
| `mask::`*`права_доступа`* | `mask::rwx` | для всех, кроме владельца и других пользователей[a] |

[a] Иногда маски не совсем понятны; пояснения к ним даны далее в этом разделе.

Пользователей и группы можно задавать именами или идентификаторами UID/GID. Точное число записей, которое может содержать список ACL, зависит от реализации файловой системы и колеблется от 25 в XFS до практически неограниченного значения в файловых системах ReiserFS и JFS. Файловые системы `ext*` допускают использование до 32 записей, что, вероятно, вполне оправдано с точки зрения обеспечения возможности управления списком.

### *Взаимодействие между традиционными режимами и списками ACL*

При использовании ACL файлы сохраняют свои исходные биты режима, но при этом автоматически поддерживается целостность атрибутов, и два набора прав доступа не могут вступить в конфликт. Ниже приведен пример (используется синтаксис Linux-команд) автоматического обновления записей ACL в ответ на изменения, выполненные "старой" командой **chmod**.

```
$ touch /tmp/example
$ ls -l /tmp/example
-rw-rw-r--  1 garth  garth    0 Jun 14 15:57 /tmp/example
$ getfacl /tmp/example
getfacl: Removing leading '/' from absolute path names
# file: tmp/example
# owner: garth
# group: garth
user::rw-
group::rw-
other::r--
$ chmod 640 /tmp/example
$ getfacl --omit-header /tmp/example
user::rw-
group::r--
other::---
```

Эта принудительная целостность атрибутов позволяет более старым программам, которые даже не подозревают о существовании ACL, достаточно успешно работать в использующей эти списки среде. Однако существует и определенная проблема. Несмотря на то что ACL-запись `group::` в приведенном примере вроде и отслеживает средний набор традиционных битов режима, это не всегда так.

Для того чтобы понять, в чем здесь дело, предположите, что унаследованная программа очищает биты разрешения записи во всех трех наборах прав традиционного ре-

жима (например, **chmod ugo-w** *файл*). Понятно, что целью выполнения этой команды является запрет выполнения записи в файле кем-либо. Но что произойдет, если результирующий список ACL выглядит следующим образом?

```
user::r--
group::r--
group:staff:rw-
other::r--
```

С точки зрения унаследованной программы файл выглядит неизменяемым, хотя в действительности он доступен для записи для любого члена группы `staff`. Подобная ситуация нежелательна. Для снижения вероятности недоразумений приняты следующие правила.

- По определению записи `user::` и `other::` списка ACL идентичны битам режима "владелец" и "другие" традиционного режима файла. Изменение режима ведет к изменению соответствующих записей ACL и наоборот.
- Во всех случаях эффективными правами доступа, предоставляемыми владельцу файла и пользователям, которые не указаны как-либо иначе, являются те, которые указаны в соответствующих записях `user::` и `other::` списка ACL.
- Если файл не имеет явно определенного списка ACL или имеет ACL, который состоит только из одной записи `user::`, одной записи `group::` и одной записи `other::`, эти записи идентичны трем наборам традиционных битов режима. Именно такой случай продемонстрирован в приведенном выше примере применения команды **getfacl**. (Подобный ACL называют минимальным, и он не требует действительной реализации в виде логически отдельного ACL.)
- В более сложных списках ACL традиционные биты режима соответствуют специальной записи ACL, называемой "маской", а не записи `group::`. Маска ограничивает доступ так, что ACL может предоставлять права всем названым пользователям, всем названым группам и группе, заданной по умолчанию.

Иначе говоря, маска определяет верхний предел прав доступа, которые система ACL может присваивать отдельным группам и пользователям. Концептуально эта маска аналогична команде **umask**, за исключением того, что маска ACL действует всегда и указывает предоставленные, а не отобранные права. Записи ACL для названых пользователей, названых групп и группы, заданной по умолчанию, могут содержать права доступа, отсутствующие в маске, но ядро будет просто их игнорировать.

В результате традиционные биты режима не в состоянии уменьшить права, глобально предоставленные списком ACL. Более того, удаление бита из части группы традиционного режима ведет к удалению соответствующего бита в маске ACL и, следовательно, к лишению этих прав всех пользователей, кроме владельца файла и тех, кто относится к категории "другие".

При расширении списка ACL, приведенного в предыдущем примере, для включения в него записей конкретного пользователя и группы команда **setfacl** автоматически определяет соответствующую маску.

```
$ ls -l /tmp/example
-rw-r----- 1 garth  garth 0 Jun 14 15:57 /tmp/example
$ setfacl -m user::r,user:trent:rw,group:admin:rw /tmp/example
$ ls -l /tmp/example
-r--rw----+ 1 garth  garth 0 Jun 14 15:57 /tmp/example
$ getfacl --omit-header /tmp/example
```

```
user::r--
user:trent:rw-
group::r--
group:admin:rw-
mask::rw-
other::---
```

Как видите, Linux-версия команды **setfacl** генерирует маску, которая позволяет вступить в действие всем правам, предоставленным в списке ACL. Для определения маски вручную нужно включить ее в список записей ACL, переданный команде **setfacl**, либо использовать опцию **-n**, чтобы помешать ее повторному генерированию командой **setfacl**. (В системе Solaris команда **setfacl** по умолчанию не пересчитывает запись с маской; поэтому для ее повторного генерирования используйте флаг **-r**.)

Обратите внимание на то, что в результате выполнения второй команды **ls -l** (после команды **setfacl**) в конце режима файла выводится знак "+", означающий, что с ним теперь связан реальный список ACL. Первая команда **ls -l** не выводит знак "+", поскольку на данный момент список ACL является "минимальным". Другими словами, он полностью описан 9-битовым режимом, и поэтому нет необходимости хранить его отдельно.

Следует иметь в виду, что применение традиционной команды **chmod** для манипулирования правами доступа группы в файле, связанном с ACL, сказывается только на маске. Продолжим рассмотрение предыдущего примера.

```
$ chmod 770 /tmp/example
$ ls -l /tmp/example
-rwxrwx---+ 1 garth  staff    0 Jun 14 15:57 /tmp/example
$ getfacl --omit-header /tmp/example
user::rwx
user:trent:rw-
group::r--
group:admin:rw-
mask::rwx
other::---
```

В данном случае результат выполнения команды **ls** вводит в заблуждение. Несмотря на обширные права, предоставленные группе, ни один из ее членов в действительности не обладает правами выполнения файла на одном лишь основании принадлежности к группе. Для того чтобы предоставить это право, придется отредактировать список ACL вручную.

### Определение прав доступа

При попытке получения процессом доступа к файлу эффективный UID сравнивается с UID владельца файла. Если они совпадают, права доступа определяются правами `user::` в списке ACL. В противном случае, при наличии соответствия записи конкретного пользователя в ACL, права определяются этой записью и маской ACL. При отсутствии подходящей записи конкретного пользователя файловая система пытается найти подходящую запись группы, предоставляющую запрошенные права. Эти записи также обрабатываются в сочетании с маской ACL. При отсутствии подходящей записи права доступа определяются записью `other::`.

### Наследование списка ACL

Кроме типов записей, перечисленных в табл. 6.8, списки ACL каталогов могут содержать записи, определенные "по умолчанию", которые передаются спискам ACL новых

файлов и подкаталогов внутри них. Подкаталоги получают свои записи как в форме записей активного ACL, так и в форме записей, заданных по умолчанию. Следовательно, исходные заданные по умолчанию записи могут со временем распространяться по нескольким уровням иерархии каталогов.

Связь между родительским и дочерними ACL не сохраняется при копировании заданных по умолчанию записей. Изменения в заданных по умолчанию записях родительского каталога не отражаются в списках ACL существующих подкаталогов.

## Списки NFSv4 ACL

В этом разделе рассмотрим характеристики списков NFSv4 ACL и синтаксис Solaris-команд, используемых для их установки и просмотра. Система AIX также поддерживает списки NFSv4 ACL, но для работы с ними использует другие команды (**aclget**, **aclput**, **acledit** и пр.). Не останавливаясь на деталях синтаксиса конкретного набора команд, сосредоточимся на теоретической основе, не зависящей от системы. Понимая основные принципы, нетрудно будет освоить соответствующие команды в интересующей вас системе.

С точки зрения структуры, списки NFSv4 ACL аналогичны спискам ACL в системе Windows. Основное различие между ними состоит в спецификации категории, к которой относится запись контроля прав доступа.

В списках ACL обеих систем запись хранится в виде строки. Для списков Windows ACL эта строка обычно содержит идентификатор защиты Windows (Windows security identifier — SID), в то время как NFSv4-строка, как правило, имеет такой формат: `user:`*имя_пользователя* или `group:`*имя_группы*. Она также может иметь один из специальных маркеров: `owner@`, `group@` или `everyone@`. В действительности эти "специальные" записи можно считать самыми популярными, поскольку они связаны с битами режима доступа, установленными для каждого файла.

Такие системы, как Samba, которые разделяют файлы между системами UNIX и Windows, должны обеспечивать определенный способ преобразования данных между Windows и NFSv4.

Модель прав доступа систем Windows и NFSv4 отличается большей детализацией по сравнению с традиционной UNIX-моделью "чтение-запись-выполнение". Рассмотрим ее основные отличительные черты.

- В системе NFSv4 разрешение на создание файлов в каталоге отличается от разрешения создавать подкаталоги.
- В системе NFSv4 предусмотрен отдельный бит разрешения на "добавление".
- В системе NFSv4 предусмотрены отдельные биты разрешения на чтение и запись для данных, файловых атрибутов, расширенных атрибутов и ACL.
- В системе NFSv4 реализовано управление возможностью пользователей менять владельца файла посредством стандартной системы ACL. В традиционной модели UNIX возможность менять владельца файла обычно имеет только суперпользователь.

В табл. 6.9 приведены различные разрешения (права доступа), которые могут назначаться в системе NFSv4, а также однобуквенные коды, используемые для их представления, и имена, отображаемые и принимаемые Solaris-командами **ls** и **chmod**.

**Таблица 6.9. Права доступа к файлам в системе NFSv4**

| Код | Имя | Права доступа |
|---|---|---|
| r | read_data<br>list_directory | Чтение данных (для файла) или получение содержимого каталога (для каталога) |
| w | write_data<br>add_file | Запись данных (для файла) или создание файла (для каталога) |
| p | append_data<br>add_subdirectory | Добавление данных (для файла) или создание подкаталога (для каталога) |
| R | read_xattr | Чтение именованных ("расширенных") атрибутов |
| W | write_xattr | Запись именованных ("расширенных") атрибутов |
| x | execute | Выполнение файла как программы |
| D | delete_child | Удаление дочернего объекта в каталоге |
| a | read_attributes | Чтение нерасширенных атрибутов |
| A | write_attributes | Запись нерасширенных атрибутов |
| d | delete | Удаление |
| c | read_acl | Чтение списка управления доступом |
| C | write_acl | Запись списка управления доступом |
| o | write_owner | Изменение владельца |
| s | synchronize | Синхронное завершение записи |

Некоторые права доступа имеют несколько имен, поскольку они представлены одинаковым значением флага, но интерпретируются по-разному для файлов и каталогов. Этот вид перегрузки, вероятно, вам знаком из традиционных UNIX-систем управления доступом. (Например, код x в традиционной системе означает право на выполнение обычного файла и "транзитное" разрешение применительно к каталогу.)

Несмотря на то что в NFSv4 модель прав доступа достаточно детализирована, отдельные разрешения должны быть очевидны, т.е. не должны требовать дополнительных разъяснений. Разрешение "synchronize" позволяет клиенту указать, что модификации файла должны быть синхронизированы, т.е. вызовы, связанные с записью данных, не должны завершиться до тех пор, пока данные не будут реально сохранены на диске.

Расширенный атрибут — это именованная часть данных, которая сохраняется вместе с файлом. Большинство современных файловых систем поддерживает такие атрибуты, хотя они пока не нашли широкого применения. В настоящее время расширенные атрибуты, в основном, используются для сохранения самих списков ACL. Однако модель прав доступа в NFSv4 предусматривает обработку списков ACL отдельно от других расширенных атрибутов.

### *Объекты NFSv4, для которых можно задавать права доступа*

В дополнение к обычным спецификаторам `user:`*`имя_пользователя`* или `group:`*`имя_группы`*, в системе NFSv4 определено еще несколько объектов, которые могут быть указаны в списке ACL. Самыми важными из них являются `owner@`, `group@` и `everyone@`, которые связаны с традиционными категориями в 9-битовой модели прав доступа.

В системе NFSv4 спецификация (RFC3530) определяет еще несколько таких специальных объектов, как `dialup@` и `batch@`. С точки зрения системы UNIX все они немного странные. Насколько нам известно, их назначение — способствовать совместимости с Windows.

Система NFSv4 отличается от POSIX по ряду признаков. Прежде всего, тем, что NFSv4 не имеет стандартного объекта, используемого по умолчанию для управления ACL-наследованием. Вместо этого любая отдельная запись управления доступом (ACE) может быть помечена как передающаяся по наследству (см. ниже раздел "Наследование списков ACL"). Кроме того, система NFSv4 не использует маску для согласования прав доступа, заданных в режиме файла с помощью списка ACL. Режим, необходимый для согласования с параметрами, задаваемыми для `owner@`, `group@` и `everyone@`, а также файловые системы, которые реализуют списки NFSv4 ACLs, должны сохранять эту совместимость при обновлении режима либо списка ACL.

### Определение прав доступа

В системе POSIX ACL файловая система пытается сопоставить идентификатор пользователя с одной, но самой важной записью управления доступом. Эта запись (ACE) затем предоставляет полный набор управляющих прав для доступа к файлу.

Система NFSv4 отличается тем, что ACE может задавать только частичный набор прав. Каждая запись NFSv4 ACE либо разрешает, либо запрещает доступ. Запись действует больше как маска, чем официальная спецификация всех возможных прав[12]. К любой конкретной ситуации можно применить несколько записей ACE.

Принимая решение о том, разрешать ли выполнение конкретной операции, файловая система считывает список ACL и обрабатывает записи ACE до тех пор, пока все затребованные права не будут разрешены или какое-то затребованное право не будет отклонено. При этом рассматриваются только те записи ACE, строки которых соответствуют текущему идентификатору пользователя.

Файловая система может "пройти" список NFSv4 ACL до конца, не получив определенного ответа на запрос прав доступа. В этом случае стандарт NFSv4 считает полученный результат неопределенным, однако большинство существующих реализаций выберут доступ с отказом, во-первых, потому, что такое соглашение используется в системе Windows, и, во-вторых, потому, что это — единственный вариант, который имеет смысл.

### Наследование списка ACL

Подобно спискам POSIX ACL, списки NFSv4 ACL позволяют созданным объектам наследовать записи управления доступом от включающего их каталога. Однако система NFSv4 при небольшом выигрыше в возможностях гораздо более запутана по сравнению с POSIX. И вот почему.

- Любая запись может быть отмечена как наследуемая. Признаки наследования для создаваемых подкаталогов (`dir_inherit` или `d`) и создаваемых файлов (`file_inherit` или `f`) устанавливаются по отдельности.
- Можно применять различные записи управления доступом для новых файлов и новых каталогов путем создания отдельных записей управления доступом в родительском каталоге с соответствующими признаками. Можно также применить одну запись ACE ко всем новым дочерним записям (любого типа) за счет включения обоих флагов `d` и `f`.

[12] Помимо записей "разрешения" и "отказа", спецификация NFSv4 также позволяет использовать записи "аудита" и "аварийного сигнала", которые не оказывают влияния на вычисления прав, но являются потенциально полезными для контроля регистрации и безопасности. Точное значение этих записей зависит от конкретной реализации.

- С точки зрения определения прав доступа, записи управления доступом оказывают одинаковое влияние на родительский (исходный) каталог независимо от того, наследуемый он или нет. Если вы хотите применить запись к дочернему, но не к родительскому каталогу, включите для соответствующей записи ACE флаг `inherit_only` (`i`).
- Новые подкаталоги обычно наследуют две копии каждой записи ACE: с отключенными флагами наследования (применяется к самому подкаталогу) и с включенным флагом `inherit_only` (устанавливает новый подкаталог для распространения своих передаваемых по наследству записей ACE). Вы можете подавить создание этой второй записи ACE включением флага `no_propagate` (`n`) для копии записи ACE родительского каталога. Конечный результат состоит в том, что запись ACE распространяется только на прямых потомков исходного каталога.
- Не путайте распространение записей управления доступом с настоящим наследованием. Установка связанного с наследованием флага для записи ACE просто означает, что данная запись ACE будет скопирована в новые записи. Она не создает никаких постоянных отношений между родителем и его потомком. Если вы позже измените записи ACE для родительского каталога, дочерние каталоги при этом не будут обновлены.

Различные флаги наследования приведены в табл. 6.10.

**Таблица 6.10. Флаги наследования в системе NFSv4 ACE**

| Код | Имя | Назначение |
|---|---|---|
| f | file_inherit | Распространение записи ACE на создаваемые файлы |
| d | dir_inherit | Распространение записи ACE на создаваемые подкаталоги |
| i | inherit_only | Распространение, но не применение к текущему каталогу |
| n | no_propagate | Распространение на новые подкаталоги, но не на их потомков |

### *Использование списков NFSv4 ACL в системе Solaris*

В системе Solaris поддержка списков ACL интегрирована в команды `ls` и `chmod`, что является простым расширением обычных функций этих команд. Таким удачным способом поддерживаются как POSIX-, так и NFSv4-списки ACL, но здесь мы демонстрируем только примеры NFSv4-списков. Конкретные детали чтения или установки списков ACL зависят от используемой файловой системы.

При выполнении команды **`ls -v`** отображается ACL-информация для объектов файловой системы. Как и в случае использования флага **`-l`**, вы должны включить флаг `-d`, если хотите получить список ACL для каталога; в противном случае команда **`ls -v`** отобразит список ACL каждого потомка каталога. Рассмотрим следующий простой (!) пример.

```
solaris$ mkdir /var/tmp/example
solaris$  ls -dv /var/tmp/example
drwxr-xr-x 2 garth staff 2 Jan 11 07:19 /var/tmp/example
   0:owner@::deny
   1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
       /append_data/write_xattr/execute/write_attributes/write_acl
       /write_owner:allow
   2:group@:add_file/write_data/add_subdirectory/append_data:deny
```

```
3:group@:list_directory/read_data/execute:allow
4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
    /write_attributes/write_acl/write_owner:deny
5:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
    /read_acl/synchronize:allow
```

Может показаться, что создаваемый каталог получает сложный список ACL, но на самом деле это не так: список ACL — всего лишь 9-битовый код режима, отображаемый в первой строке результата, преобразуемого в формат списка ACL. Совсем не обязательно для файловой системы хранить реальный список ACL, поскольку список ACL и режим доступа эквивалентны. (Такие списки ACL получили название "тривиальных".) Если бы каталог имел реальный список ACL, команда `ls`, чтобы обозначить наличие списка ACL, отображала бы биты режима со знаком "+" в конце (например, `drwxr-xr-x+`).

Каждое нумерованное выражение представляет одну запись управления доступом. Вот как выглядит ее формат.

*индекс*`:`*объект*`:`*права_доступа*`:`*флаги_наследования*`:`*тип*

Числа поля *индекс* добавляются командой `ls` для порядка и не являются частью реального списка ACL. Их можно использовать в последующих командах `chmod` для идентификации специальной записи ACE, подлежащей замене или удалению.

Поле *объект* может занимать одно из таких ключевых слов, как `owner@`, `group@`, `everyone@`, или элемент в такой форме: `user:`*имя_пользователя* либо `group:`*имя_группы*.

Поле *тип* в записи ACE задает один из двух возможных вариантов: `allow`, т.е. "разрешить", или `deny`, т.е. "отказать". Теоретически возможно использование вариантов `alarm` (аварийный сигнал) и `audit` (аудит), но в ZFS эти возможности не реализованы.

Оба поля *права_доступа* и *флаги_наследования* представляют собой списки опций, разделенные слешами. Удивительно, но команда `ls` опускает поле *флаги_наследования* (и одно из двоеточий), если все эти флаги отключены, но это не касается поля *права_доступа*.

Еще больше добавляет "тумана" тот факт, что команда **`ls`** отображает сразу несколько имен для таких битов прав доступа, как **`r`** (чтение данных/содержимого каталога), **`w`** (запись данных/добавление файла) и **`p`** (добавление данных /подкаталога), как если бы они были отдельными элементами прав доступа. В действительности они передают файловые и каталоговые интерпретации одних и тех же битов и всегда представляют "объединенное" наличие или отсутствие конкретной привилегии.

Эти индивидуальные особенности (с учетом использования в качестве разделителя символа двоеточия в рамках поля *объект*) усложняют сценариям анализ результата выполнения команды **`ls -v`**. Если вам необходимо организовать программную обработку списков ACL, поищите сначала библиотеку, например, такую, как модуль Solaris::ACL Perl из архива документации и программного обеспечения CPAN (Comprehensive Perl Archive Network), который существенно упростит этот процесс. В крайнем случае можно использовать результат выполнения команды **`ls -V`** (см. ее описание ниже), поскольку этот формат более подходит для анализа.

С помощью команды **`ls -V`** можно получить табличное представление записей списка ACL. В этом случае права доступа отображаются их однобуквенными кодами, как показано в табл. 6.9. Для каждой записи управления доступом отображены все возможные биты. "Отключенные" права представлены прочерками (подобно тому, как команда **`ls`** отображает традиционный режим файла).

```
solaris$  ls -dV /var/tmp/example
drwxr-xr-x 2 garth staff 2 Jan 11 07:19 /var/tmp/example
           owner@: --------------:-------:deny
           owner@: rwxp---A-W-Co-:-------:allow
           group@: -w-p----------:-------:deny
           group@: r-x-----------:-------:allow
        everyone@: -w-p---A-W-Co-:-------:deny
        everyone@: r-x---a-R-c--s:-------:allow
```

### Взаимодействие между списками ACL и традиционными режимами

Рассмотрим некоторые аспекты преобразования режимов в списки ACL. Во-первых, обратите внимание на то, что записи `group@` и `everyone@` в приведенном выше примере отличаются друг от друга, несмотря на то что соответствующие части в коде режима (`r-x`) совпадают. И это не потому, что правила преобразования различны для категорий `group@` и `everyone@`; все дело в том, что определенные права доступа невозможно реально экстраполировать из традиционного режима.

Эти "неустановленные" биты прав доступа принимают стандартные значения благодаря дополнениям только к записям ACE категории `everyone@`. Права доступа `write_xattr`, `write_attributes`, `write_acl` и `write_owner` всегда означают запрещенный доступ, а права `read_xattr`, `read_attributes`, `read_acl` и `synchronize` всегда разрешены. Если вы "выведите" эти права из набора `everyone@`, то увидите, что оставшиеся записи ACE для категории `everyone@` такие же, как для категории `group@`.

Безусловно, эти "совместимые" права применяются только к тривиальным спискам ACL. Редактируя напрямую список ACL, вы можете установить биты в любой комбинации.

Режим и список ACL должны оставаться совместимыми, поэтому при корректировании одного из этих вариантов другой обновляется автоматически. Файловая система ZFS успешно определяет соответствующий режим для заданного списка ACL, но ее алгоритм для генерирования и обновления списков ACL в ответ на изменения режимов остается рудиментарным. Результаты с функциональной точки зрения нельзя назвать некорректными, но они зачастую излишне детализированы, нечитабельны и с трудом поддаются интерпретации. В частности, для категорий `owner@`, `group@` и `everyone@` система может сгенерировать несколько наборов записей (и притом противоречивых), которые будут зависеть от порядка вычисления.

Возьмите себе за правило никогда не трогать режим файла или каталога после применения списка ACL. В самом худшем случае с помощью команды `chmod A- `*файл* удалите список ACL и начните все заново.

### Модифицирование списков NFSv4 ACL в системе Solaris

Поскольку файловая система ZFS усиливает соответствие между режимом файла и ее списком ACL, все файлы имеют по крайней мере тривиальный список ACL (виртуальный или нет). Следовательно, изменения списка ACL всегда являются обновлениями. Изменения списка ACL выполняйте с помощью команды `chmod`. Ее основной синтаксис такой же, как всегда.

**chmod [-R]** *acl_операция файл ...*

В табл. 6.11 приведены различные типы ACL-операций, "понимаемых" командой `chmod`. К сожалению, не существует ACL-аналога для инкрементного варианта команды `chmod`, символического синтаксиса для управления традиционными режимами. Невозможно добавить или удалить из записи ACE индивидуальные права доступа — вам придется заменить запись полностью.

**Таблица 6.11. ACL-операции, принимаемые Solaris-командой chmod**

| Операция | Функция |
|---|---|
| **A-** | Заменяет весь список ACL его тривиальной версией, "взятой" из режима |
| **A***индекс-* | Удаляет одну запись управления доступом в заданной позиции |
| **A-***ace* | Удаляет заданную запись управления доступом в любой позиции |
| **A***индекс=ace*[,*ace*...] | Заменяет целиком одну или несколько записей управления доступом |
| **A+***ace* | Добавляет запись управления доступом в начало списка ACL |
| A*индекс+ace*[,*ace*...] | Добавляет записи управления доступом перед записью, отмеченной заданным значением *индекс* |

Числа поля *индекс* (см. табл. 6.11) показаны выше на примере команды **ls -v**. Это числа, которыми по порядку, начиная с нуля, нумеруются записи управления доступом. Вы можете закодировать поля *ace* либо именами, либо однобуквенными кодами прав доступа. Например, команда

```
solaris$ chmod A+user:ben:C:allow /var/tmp/example
```

предоставляет пользователю ben право редактировать список ACL для каталога **/var/tmp/example**.

Помните, что определение доступа — это итеративный процесс, который выполняется по всему списку ACL, поэтому пользователь ben сохраняет все права, которые он имел при использовании предыдущей версии списка ACL. Новая запись управления доступом помещается в начало списка ACL (с нулевым индексом), поэтому команда

```
solaris$ chmod A0- /var/tmp/example
```

удаляет только что добавленную запись ACE и возвращает список ACL в исходное состояние.

# 6.7. Упражнения

6.1. Что такое атрибут **umask**? Задайте такое значение атрибута **umask**, которое запрещало бы членам группы и рядовым пользователям доступ к создаваемым файлам.

6.2. В чем разница между жесткими и символическими ссылками? Когда предпочтительнее использовать ссылки того или иного типа?

6.3. ★ Какие действия необходимо выполнить в вашей системе для автоматического монтирования NTFS-раздела с локального жесткого диска? Какой будет наиболее подходящая точка монтирования для такого раздела в соответствии с соглашениями, действующими в вашей системе и используемыми на вашем узле?

6.4. ★ При инсталляции новой системы важно разбить жесткий диск на разделы таким образом, чтобы для каждой файловой системы (**/var**, **/usr** и так далее) было достаточно свободного места. Предположим, в дистрибутиве Foobar Linux используются следующие установки:

| | |
|---|---|
| **/** | 2 Гбайт; |
| **/var** | 100 Мбайт; |
| **/boot** | 100 Мбайт; |
| < раздел подкачки> | 2 Гбайт; |
| **/usr** | оставшееся свободное место. |

6.5. Какие проблемы при такой конфигурации могут возникнуть на интенсивно эксплуатируемом сервере?

6.6. ★ Почему рекомендуется помещать некоторые разделы (в частности, **/var**, **/home** и раздел подкачки) на отдельные диски, чтобы отделить их от файлов данных и программ? А как насчет каталога **/tmp**? Приведите аргументы по каждой из перечисленных файловых систем.

6.7. ★ Напишите сценарий, который находит все жесткие ссылки в системе.

6.8. ★ Какими командами реализуются перечисленные ниже действия?

а) предоставьте владельцу файла README право чтения/записи, а остальным пользователям — только право чтения

б) установите для файла бит setuid, не меняя текущие права доступа (и даже не просматривая их)

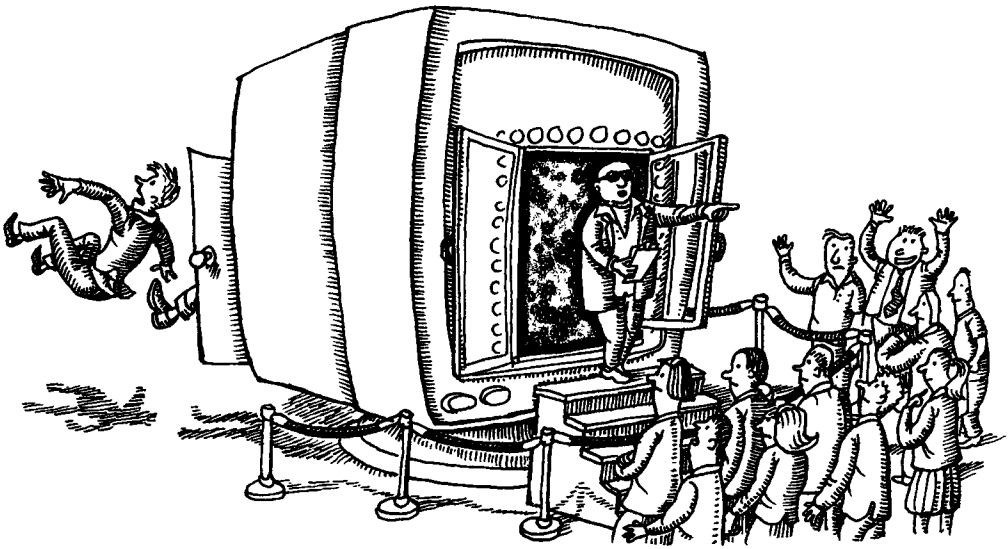
в) отобразите содержимое текущего каталога, отсортировав список по дате последней модификации, причем файлы, которые редактировались последними, должны стоять в конце списка

г) поменяйте группу файла **shared** с "user" на "friends"

6.9. ★ По соглашению каталог **/tmp** доступен для всех пользователей, которых заботит создание в нем файлов. Что мешает одному пользователю читать временные файлы другого пользователя или удалять их? Что должно удержать рассерженного пользователя от заполнения каталога **/tmp** ненужными файлами? Каковы были бы последствия такой атаки?

# Глава 7

# Добавление новых пользователей



Подключение новых пользователей и удаление старых — рутинная задача в большинстве систем. Выполнение этих задач не представляет особой сложности, но достаточно трудоемко. Большинство администраторов создают различные средства автоматизации этого процесса, а затем поручают выполнение реальной работы своему заместителю или оператору.

В наши дни мы являемся свидетелями возвращения эпохи централизованных серверов с сотнями учетных записей, которые теперь используются наряду с распределенными серверами, предназначенными для обслуживания всего нескольких пользователей. Администраторы должны глубоко понимать подсистему учета пользователей, чтобы уметь настраивать работу сетевых служб и надлежащим образом конфигурировать локальные учетные записи. Зачастую управление учетными записями в серверах является всего лишь небольшой частью общей задачи по инициализации учетных записей для всего предприятия.

Современным производственным средам необходимо иметь не просто средство добавления пользователей на заданные компьютеры, а инструмент для управления пользователями и их бесчисленными учетными записями и паролями в вычислительной среде, т.е. систему управления учетной информацией (Identity Management System). Широко используются такие службы управления каталогами, как Active Directory (Microsoft), OpenLDAP и Fedora Directory Server, поэтому мы подробно рассмотрим, как эти системы влияют на решение задач управления учетными записями. (Как обычно, "близорукие" инструменты Microsoft плохо взаимодействуют с другими без помощи службы Active Directory. Увы!)

Потребности некоторых узлов могут расширить возможности даже таких систем. Мы не намерены подробно останавливаться на коммерческих системах управления учетной информацией, но все же отметим несколько заслуживающих внимания вариантов. К ним, вероятно, стоит присмотреться получше, особенно в случае очень большого узла, в котором требуется соответствие определенным законам, например, HIPAA или Сарбейнза-Оксли (в США). См. раздел 7.11.

Правильное управление учетными записями является залогом безопасности системы. Редко используемые учетные записи становятся главными мишенями атак злоумышленников, как и те записи, пароли к которым легко подобрать. Даже если для подключения и удаления пользователей применяются автоматизированные системные утилиты, важно понимать, какие при этом происходят изменения в системе. Поэтому мы начинаем наш разговор об управлении учетными записями с рассмотрения неструктурированных файлов, которые необходимо модифицировать для добавления пользователей на компьютер.

Затем рассмотрим автоматизированные средства, поставляемые со всеми описываемыми здесь примерами операционных систем, а также файлы конфигурации, управляющие их поведением. Вызывает некоторое недоумение то, что во всех наших примерах систем инструменты управления пользователями называются **useradd**, **userdel** и **usermod**, хотя эти утилиты не одинаковы. (Кроме того, в системе AIX соответствие имен достигается за счет заключения в оболочку ее "родных" программ **mkuser**, **rmuser** и **chuser** с помощью специальных сценариев.)

Стандартная команда **useradd** работает вполне хорошо и удовлетворяет потребностям большинства организаций. К сожалению, этого нельзя сказать о команде **userdel**.

Кроме того, для добавления и удаления пользователей во многих системах используются простые GUI-инструменты, хотя эти средства обычно не реализуют пакетный режим или усовершенствованную локализацию. Они достаточно просты, и мы считаем нецелесообразным подробно их описывать, но предоставим ссылки на документацию по каждой системе.

В этой главе сосредоточимся на добавлении и удалении пользователей. Многие темы, связанные с управлением учетными записями, описаны в других главах (здесь вы найдете соответствующие ссылки).

- Подключаемые модули аутентификации (Pluggable authentication modules — PAM) — набор библиотек, предназначенных для создания сильных паролей; описан в главе 22 (см. раздел 22.5).
- Хранилища паролей описаны в главе 4 (см. раздел 4.3).
- О таких службах управления каталогами, как NIS и OpenLDAP, рассказывается в главе 19 (см. раздел 19.3). Отдельные комментарии по использованию службы Active Directory можно найти в главе 30 (см. раздел 30.9).
- Наконец, вопросы стратегии и методики управления являются основными темами главы 32.

В следующих трех разделах речь пойдет об основных файлах, используемых в системе управления пользователями.

## 7.1. Файл /etc/passwd

Файл **/etc/passwd** содержит список пользователей, которые известны системе. Его можно расширить или заменить службой управления каталогами, поэтому его можно считать полным и заслуживающим доверия только в автономных системах.

В процессе регистрации пользователя система обращается к файлу **/etc/passwd** в поисках идентификатора пользователя и его домашнего каталога. Каждая строка файла описывает одного пользователя и содержит семь следующих полей, разделенных двоеточиями:

- регистрационное имя;
- шифрованный пароль или "заполнитель" пароля (вопросы применения шифрованных паролей описаны далее в этом разделе);
- идентификатор пользователя;
- идентификатор группы по умолчанию;
- поле GECOS (полное имя, номер офиса, рабочий и домашний телефоны);
- домашний каталог;
- регистрационная оболочка.

Вот примеры правильно составленных записей файла **/etc/passwd**.

```
root:x:0:0:The System,,x6096, :/:/bin/sh
jl:!:100:0:Jim Lane,ECOT8-3,,:/staff/jl:/bin/sh
dotty:x:101:20: :/home/dotty:/bin/tcsh
```

Шифрованные пароли занимают второе поле, но их уже нельзя считать безопасными: современные быстрые компьютеры позволяют взламывать такие пароли за считанные минуты. Все версии UNIX и Linux в настоящее время скрывают шифрованные пароли, помещая их в отдельный файл, недоступный для чтения. В системах Linux, Solaris и HP-UX файл **passwd** в поле шифрованного пароля содержит элемент `x`, а в системе AIX — элемент `!` или `*` (в системах AIX заполнитель "`*`" блокирует учетную запись).

В системах Linux, Solaris и HP-UX зашифрованные пароли хранятся в файле **/etc/shadow**, а в системе AIX — в файле **/etc/security/passwd**. Форматы этих файлов различны.

🕮 Подробнее о файле **nsswitch.conf** рассказывается в разделе 19.5.

Если пользовательские учетные записи подлежат совместному использованию посредством таких служб управления каталогами, как NIS или LDAP, вы можете увидеть специальные записи в файле **passwd**, которые начинаются с символа "+" или "-". Эти записи сообщают системе, как интегрировать данные службы управления каталогами в содержимое файла **/etc/passwd**. Эта интеграция также может быть реализована в файле **/etc/nsswitch.conf** (в системе AIX это файл **/etc/nscontrol.conf**).

В следующих разделах рассмотрим поля файла **/etc/passwd** более детально.

## Регистрационное имя

🕮 Подробнее о СУБД NIS можно прочитать в разделе 19.3.

Регистрационные имена (называемые также *именами пользователей*) должны быть уникальными и, в зависимости от системы, могут иметь ограничения на длину и набор символов. Правила построения регистрационных имен для разных систем приведены в табл. 7.1. Пользовательские имена не могут содержать двоеточия и символы новой строки, поскольку эти символы применяются в качестве разделителей полей и записей соответственно. Если у вас используется база данных NIS или NIS+, длина имени ограничена восемью символами независимо от операционной системы.

Таблица 7.1. Правила формирования регистрационных имен

| Система | Длина | Набор символов | Первый символ | Специальные правила |
|---|---|---|---|---|
| Linux | 32[а] | a-z0-9_- | a-z_ | Некоторые дистрибутивы не столь строгие |
| Solaris | 8[б] | A-Za-z0-9+._- | A-Za-z | Должна быть хотя бы одна строчная буква |
| HP-UX | 8 | A-Za-z0-9_ | A-Za-z | |
| AIX | 8[в] | POSIX; не используются пробелы, кавычки и # , = / ? \ | Не используются -@~ | Все буквы не должны быть прописными; не используется пароль "default" или "ALL" |

[а] Несмотря на то что система Linux позволяет использовать 32 символа, унаследованное программное обеспечение (например, программы **top** и **rsh**) принимает имена, состоящие только из 8 (или еще меньшего числа) символов.

[б] Это число должно увеличиться.

[в] В систему AIX 5.3 (и в более поздние версии) могут быть внесены изменения; см. ниже.

Изначально в системах UNIX допустимый диапазон ограничивался алфавитно-цифровыми символами, а длина имени не должна была превышать 8 символов. Поскольку правила в различных системах, как правило, разные, рекомендуется придерживаться наиболее строгих ограничений. Это дает возможность избежать конфликтов со старыми программами, а также использовать одно и то же регистрационное имя на любом компьютере. Считается повсеместно приемлемой комбинация из восьми (или меньше) строчных букв, цифр и символов подчеркивания.

Регистрационные имена могут включать как строчные, так и прописные буквы. Однако документ RFC822 призывает не обращать внимания на регистр букв в адресах электронной почты. Смешение регистров в именах не приводит к возникновению проблем, но имена, состоящие из строчных букв, считаются традиционными, к тому же их проще вводить. Проблемы обязательно возникнут при использовании почтовой службы, если, например, такие регистрационные имена, как `john` и `John`, будут принадлежать различным людям.

Следует выбирать такие регистрационные имена, которые легко запомнить, поэтому имя, состоящее из случайной последовательности букв, является не самым удачным вариантом. Избегайте также кличек и прозвищ, даже если они допустимы в вашей организации: имена типа DarkLord или QTPie часто включаются в адреса электронной почты (например, перед частью адреса `@hotmail.com`), так что не забывайте о самоуважении.

Поскольку регистрационные имена часто используются в адресах электронной почты, полезно определить стандартную процедуру их формирования. Пользователям должна быть предоставлена возможность делать обоснованные предположения о регистрационных именах друг друга. Имена, фамилии, инициалы и сочетания этих элементов — вот приемлемые варианты для схем формирования имен.

Любая жестко заданная схема в конечном счете приводит к появлению дубликатов или слишком длинных имен, поэтому иногда придется делать исключения. Выберите стандартный способ разрешения конфликтов, добавив, например, в конец имени цифру. В случае появления длинного имени можно использовать предоставляемую почтовой системой функцию определения псевдонимов, чтобы задать две версии одного и того же имени, по крайней мере, для электронной почты.

В крупных организациях в адресах электронной почты часто используются полные имена (например, `John.Q.Public@mysite.com`), благодаря чему регистрационные имена скрываются от внешнего мира. Это прекрасная идея, но, чтобы облегчить работу

администраторам, необходимо установить четкое и понятное соответствие между регистрационными именами и реальными именами пользователей.

Регистрационные имена должны быть уникальными в двух отношениях. Во-первых, необходимо, чтобы имя пользователя на всех компьютерах было одинаковым. Это удобно как самому пользователю, так и администратору.

Во-вторых, регистрационное имя всегда должно относиться к одному и тому же лицу. Некоторые команды (например, **ssh**) можно настроить так, чтобы они аутентифицировали удаленных пользователей по регистрационным именам. Даже если адреса `scott@boulder.colorado.edu` и `scott@refuge.colorado.edu` относятся к разным пользователям, то в случае неправильной настройки учетных записей (со слабыми средствами защиты) оба пользователя могут при определенных обстоятельствах получить доступ к файлам друг друга, не вводя пароль.

Опыт также показывает, что дублирующиеся имена сбивают с толку пользователей почтовых систем. Почтовая система различает, кому именно принадлежат имена, однако пользователи часто посылают письма не по тому адресу.

🕮 Почтовые псевдонимы подробно рассматриваются в разделе 20.5.

Если в организации имеется глобальный файл почтовых псевдонимов, то все новые регистрационные имена должны отличаться от псевдонимов, присутствующих в этом файле. В противном случае почта будет доставляться не новому пользователю, а пользователю с соответствующим псевдонимом.


Система AIX позволяет изменить максимальную длину регистрационного имени с помощью команды **chdev**. Релевантное устройство в этом случае называется **sys0**. Для получения списка стандартных атрибутов устройства выполните команду **lsattr -D -l sys0**. Среди них есть атрибут `max_logname`, который "отвечает" за максимальную длину регистрационных имен. Следующая команда позволит вам получить информацию только об одном этом атрибуте.

```
aix$ lsattr -El sys0 -a max_logname
max_logname 9 Maximum login name length at boot time True
```

Для того чтобы изменить значение максимальной длины имени, используйте следующие команды. Изменения вступят в силу после следующей перезагрузки системы[1].

```
aix$ sudo su -
aix# chdev -l sys0 -a max_logname=16
```

Стандартная максимальная длина имени равна девяти символам, но AIX-спецификация длины определяется размером буфера и поэтому должна учитывать наличие нулевого символа в качестве признака завершения строки. Следовательно, реальный стандартный максимум составляет восемь символов, а наша команда `chdev` устанавливает его равным 15 символам.

Система AIX поддерживает мультибайтовые символы (для азиатских языков, например), но рекомендует не использовать их. В качестве альтернативного варианта для создания имен файлов предлагается использовать переносимый POSIX-набор символов.

[1] Сначала мы не смогли реализовать эти действия из-за использования программы *sudo* (см. раздел 4.3), ведь переменные среды, установленные командой *sudo*, обычно отличаются от описанных выше. Поэтому сначала необходимо выполнить команду *sudo su -*, а затем изменить значение атрибута. В новых версиях программы *sudo* (1.70 или более поздние) предусмотрен флаг *-i* для корректного выхода из такой ситуации.

## Зашифрованный пароль

Современные системы помещают в файл **/etc/passwd** некий заполнитель для зашифрованного пароля, а затем во время первой регистрации предлагают пользователю ввести реальный пароль. Они также поддерживают ряд схем шифрования (помимо стандартного UNIX-алгоритма **crypt**). Зашифрованный пароль необходим для идентификации формы шифрования, используемой для его генерирования. Наши примеры систем поддерживают разнообразные алгоритмы шифрования: традиционный **crypt** (построенный на базе DES), MD5, Blowfish и итеративная версия MD5, унаследованная от проекта веб-сервера Apache.

Длина пароля — это еще один важный атрибут, который часто определяется алгоритмом шифрования. В табл. 7.2 приведены стандартные максимальные и минимальные значения длины паролей, а также системы шифрования, доступные в наших примерах систем. В некоторых системах разрешается вводить пароли произвольной длины, которые затем "молча" усекаются до действующего максимума длины (показанного в таблице).

**Таблица 7.2. Алгоритмы шифрования паролей и предельные значения длины паролей**

| Система | Минимум | Максимум | Алгоритм | Место установки |
|---|---|---|---|---|
| Linux | 5 | 8 | crypt, MD5, Blowfish[a] | **/etc/login.defs** |
| Solaris | 6 | 8[б] | crypt, MD5, Blowfish, SHA256 | **/etc/security/policy.conf** **/etc/security/crypt.conf** |
| HP-UX | 6[в] | 8 | crypt | **/usr/include/limits.h**[г] |
| AIX | 0 | 8 | crypt, MD5 (BSD), Apache | Параметр для команды **passwd** |

[a] Blowfish — стандартный алгоритм в системах SUSE и openSUSE; большинство других систем использует алгоритм MD5.
[б] Максимальная длина зависит от выбранного алгоритма.
[в] Суперпользователь может установить пароль пользователя любой длины.
[г] Этот файл содержит много конструкций **#ifdef** и поэтому его трудно прочитать и понять.

Если вы решили не пользоваться системными средствами добавления пользователей и для создания новой учетной записи вручную редактировать файл **/etc/passwd** (конечно же, с помощью редактора **vipw** — см. раздел 7.4), поставьте звездочку (*) или символ "x" в поле зашифрованного пароля. Звездочка воспрепятствует несанкционированному использованию учетной записи до установки реального пароля. *Никогда не оставляйте это поле пустым*, иначе безопасность системы окажется под серьезной угрозой, поскольку для доступа к такой учетной записи пароль не требуется.

С криптографической точки зрения, алгоритм MD5 лишь немногим более устойчив, чем предыдущий стандарт DES (используемый алгоритмом **crypt**), зато в нем допускаются пароли произвольной длины. Чем длиннее пароли, тем они надежнее. Алгоритм MD5 не лишен некоторых криптографических недостатков, но вряд ли какая-нибудь защита может устоять против подбора пароля методом грубой силы. Сильными алгоритмами считаются SHA256 и Blowfish. Некоторые советы по выбору паролей можно найти в разделе 22.4.

Зашифрованные пароли имеют постоянную длину (34 символа в случае MD5 и 13 символов в случае DES) независимо от длины исходного пароля. Пароли шифруются с добавлением случайной "примеси", чтобы одному исходному паролю соответствовало много зашифрованных форм. Таким образом, факт выбора пользователями одинаковых паролей не может быть выявлен путем просмотра зашифрованных паролей.

Пароли MD5 легко распознать, так как они всегда начинаются с последовательности "$1$" или "$md5$"[2]. Пароли Blowfish начинаются с последовательности "$2a$", а пароли SHA256 — с последовательности "$5$".

В системе SUSE для новых паролей в качестве стандарта принят алгортм шифрования Blowfish (использует префикс "$2a$").

В настоящее время система OpenSolaris перешла на стандарт SHA256 (с префиксом "$5$"), хотя предыдущие версии этой системы использовали по умолчанию алгоритм MD5.

## Идентификатор пользователя

Идентификатор пользователя (UID) — это, как правило, 32-битовое целое число без знака, которое идентифицирует пользователя в системе. Регистрационные имена используются для удобства пользователей, а идентификаторы UID применяются (внутренне) программными средствами и файловой системой. Но для обеспечения совместимости со старыми системами рекомендуем, чтобы значение самого старшего идентификатора по возможности не превышало 32767 (наибольшее 16-битовое целое число со знаком).

🕮 Описание учетной записи суперпользователя можно найти в разделе 4.1.

По определению пользователь **root** имеет идентификатор (UID) 0. В большинстве систем есть также псевдопользователи **bin** и **daemon**, что позволяет им быть владельцами команд и файлов конфигурации. Как правило, такие псевдоимена помещаются в начало файла **/etc/passwd**, и им назначаются маленькие идентификаторы, а также фиктивный интерпретатор команд (например, **/bin/false**). Это не позволит кому-либо зарегистрироваться под такими псевдоименами. Чтобы зарезервировать побольше номеров для неперсонифицированных пользователей, рекомендуем присваивать реальным пользователям идентификаторы, начиная с номера 500 или выше. (Желаемый диапазон для новых идентификаторов можно указать в файлах конфигурации для утилиты **useradd**.)

🕮 Подробнее о "пустой" учетной записи (nobody account) написано в конце раздела 18.2.

Предусмотрен еще один специальный идентификатор, именуемый псевдопользователем (nobody), которому обычно назначаются такие значения, как –1 или –2 (эти значения, представленные в виде целых без знака для установки поля UID, оказываются максимально большими идентификаторами). Регистрационное имя псевдопользователя используется в случае, если суперпользователь на одном компьютере пытается получить доступ к файлам, смонтированным (посредством системы NFS) с другого компьютера, который не доверяет первому.

Нежелательно создавать более одной учетной записи с идентификатором 0. Может показаться удобным иметь несколько суперпользовательских записей с разными интерпретаторами команд и паролями, но в действительности это лишь создает дополнительные бреши в системе защиты и приводит к ненужным трудностям. Если нескольким пользователям необходимо быть администраторами, пусть они применяют утилиту **sudo**.

[2] Последовательность "$1$" представляет собой своего рода маркер для алгоритма BSD MD5; компания Sun использует собственную реализацию MD5 с маркировочной последовательностью "$md5$".

Избегайте повторного использования идентификаторов, даже идентификаторов тех пользователей, которые уволились из организации и учетные записи которых были удалены. Такая мера предосторожности позволит избежать путаницы, если файлы впоследствии будут восстанавливаться из резервных копий, где пользователи могут быть идентифицированы по номерам, а не по регистрационным именам.

Идентификаторы должны быть уникальными в пределах всей организации, т.е. конкретный идентификатор должен соответствовать одному и тому же регистрационному имени и физическому лицу на каждом компьютере. Если уникальность идентификаторов нарушена, то в такой системе, как NFS, появятся проблемы безопасности; кроме того, это может привести в замешательство пользователей, переходящих из одной рабочей группы в другую.

Трудно соблюдать уникальность идентификаторов, когда группы компьютеров находятся в административном подчинении разных лиц и даже организаций. Это проблема как техническая, так и концептуальная. Лучшим ее решением является создание центральной базы данных или сервера каталогов, содержащих для каждого пользователя уникальную запись. Проще всего назначить каждой группе в пределах организации отдельный диапазон идентификаторов и позволить распоряжаться им по своему усмотрению. В результате решится вопрос с уникальностью пользовательских идентификаторов, но останется проблема уникальности регистрационных имен.

Популярным средством управления идентификаторами пользователей становится также база данных LDAP. Она кратко описана в этой главе (см. раздел 7.11), а более подробно — в главе 19.

## Идентификатор группы по умолчанию

Как и идентификатор пользователя (UID), идентификатор группы (GID) является 32-битовым целым числом. Идентификатор 0 зарезервирован для группы с именем `root` или `system`. Как и в случае идентификаторов пользователей, системы используют несколько предопределенных групп для собственных нужд, так сказать для служебного пользования. Увы, в этом вопросе согласованности среди коллег-производителей систем не наблюдается. Например, группа `bin` имеет идентификатор GID, равный 1, в системах Red Hat и SUSE и GID, равный 2, — в системах Ubuntu, Solaris, HP-UX и AIX.

В те далекие времена, когда компьютеры были еще дорогим удовольствием, группы были полезны для выполнения вычислений, но так, чтобы использованное вами дисковое пространство, секунды центрального процессора и минуты процесса регистрации оплачивал соответствующий отдел. В наши дни группы используются, в основном, для организации совместного доступа к файлам.

О каталогах с установленным битом `setgid` рассказывалось в разделе 6.5.

Группы определяются в файле **`/etc/group`**, а поле идентификатора группы в файле **`/etc/passwd`** задает стандартный ("эффективный") идентификатор на момент регистрации пользователя в системе. Этот идентификатор не играет особой роли при определении прав доступа; он используется лишь при создании новых файлов и каталогов. Новые файлы обычно включаются в эффективную группу своего владельца, но если вы хотите разделять файлы с другими участниками проектной группы, не забудьте вручную изменить для этих файлов владельца группы.

При этом следует иметь в виду, что если у каталогов установлен бит `setgid` (02000) или файловая система смонтирована с опцией **`grpid`**, новые файлы включаются в группу родительского каталога.

## Поле GECOS

Это поле, в основном, используется для хранения персональной информации о каждом пользователе. Оно не имеет четко определенного синтаксиса. В принципе структура поля GECOS может быть произвольной, но команда **finger** интерпретирует разделенные запятыми элементы поля в следующем порядке:

- полное имя (часто используется только это поле);
- номер офиса и здания;
- рабочий телефон;
- домашний телефон.

О базе данных LDAP можно почитать в разделе 19.3.

С помощью команды **chfn** можно изменять информацию, содержащуюся в поле GECOS[3]. Эта команда используется для ведения и обновления списка телефонных номеров, но ею часто злоупотребляют: например, пользователь может изменить информацию так, что она станет нецензурной или некорректной. Некоторые системы можно сконфигурировать, установив определенные ограничения, а именно указать, какие поля может модифицировать команда **chfn**. Администраторы сетей большинства университетских городков совсем отключают команду **chfn**. Во многих системах команда **chfn** "понимает" только файл **/etc/passwd**, поэтому, если для управления регистрационной информацией вы используете базу данных LDAP или какую-либо иную службу управления каталогами, команда **chfn** может не работать вовсе.


В системе AIX для команды **chfn** предусмотрен флаг `-R` *модуль*, который позволяет загрузить заданный модуль для выполнения реального обновления. Доступные модули находятся в каталоге **/usr/lib/security**. Среди них находится модуль, отвечающий за функционирование службы LDAP.

## Домашний каталог

Войдя в систему, пользователь попадает в свой домашний каталог. Следует иметь в виду, что если домашние каталоги смонтированы вне файловой системы, то в случае проблем с сервером или с самой сетью они могут оказаться недоступными. Если на момент регистрации этот каталог отсутствует, выводится сообщение вроде "no home directory" (домашний каталог отсутствует)[4] и пользовательские данные помещаются в каталог /. Если в файле **/etc/login.defs** системы Linux в качестве значения поля `DEFAULT_HOME`, в котором задается домашний каталог по умолчанию, указано значение `no`, продолжение регистрации пользователя будет невозможно.

## Регистрационная оболочка

В качестве регистрационной оболочки, как правило, задается интерпретатор команд, например **/bin/sh** или **/bin/csh**, но в принципе это может быть любая программа.

[3] Исключение составляет система Solaris, где команды **chfn** нет. Суперпользователь может изменить информацию о пользователе с помощью команды **passwd -g**.

[4] Это сообщение появляется при регистрации в системе через консоль или терминал, но не через экранный диспетчер, такой как **xdm**, **gdm** или **kdm**. В последнем случае пользователь вообще будет немедленно выведен из системы, поскольку программа-диспетчер не сможет осуществить запись в нужный каталог (например, **~/.gnome**).

По умолчанию для систем UNIX используется интерпретатор **sh**, для систем Linux и Solaris — интерпретатор **bash** (GNU-версия **sh**), а для AIX — интерпретатор **ksh** (*K*orn *sh*ell). В Linux имена **sh** и **csh** являются всего лишь ссылками на **bash** и **tcsh** (усовершенствованный C-интерпретатор с командами редактирования) соответственно.

В некоторых системах пользователи могут менять интерпретатор с помощью команды **chsh**, но, подобно **chfn**, эта команда может не работать, если для управления регистрационной информацией вы используете базу данных LDAP или какую-либо иную службу управления каталогами. Если вы используете файл **/etc/passwd**, системный администратор всегда может заменить интерпретатор пользователя, отредактировав файл **passwd** с помощью программы **vipw**.

Система Linux поддерживает команду **chsh** и варианты замены интерпретаторов, перечисленные в файле **/etc/shells**. В системе SUSE этот список является обязательным для соблюдения, а в Red Hat в случае выбора незарегистрированного интерпретатора будет лишь выдано предупреждение. При добавлении записей в файл **/etc/shells** убедитесь в том, что они содержат полные пути, поскольку команда chsh и другие программы работают только с полностью заданными именами файлов.


В системах AIX пользователи могут заменить свои интерпретаторы с помощью команды **chsh** и длинного списка вариантов замены[5]. Файл **/etc/security/login.cfg** содержит список проверенных интерпретаторов. Файл **/etc/shells** включает лишь подмножество этого списка и используется только FTP-демоном **in.ftpd**. Многие из интерпретаторов, указанных в длинном списке, представляют собой жесткие ссылки на одни и те же двоичные файлы. Например, **sh**, **ksh**, **rksh**, **psh** и **tsh** (в каталогах **/bin** и **/usr/bin**) — это, по сути, одна и та же программа, которая изменяет свое поведение в зависимости от своего имени. Как и в случае с командой **chfn**, команда **chsh** принимает флаг **-R** *модуль* для предоставления базы данных LDAP и других служб управления каталогами.


В системе Solaris только суперпользователь может заменить интерпретатор пользователя (с помощью команды **passwd -e**). Список разрешенных интерпретаторов содержится в файле **/etc/shells** (который не существует по умолчанию, несмотря на наличие его man-страницы).

## 7.2. Файлы /etc/shadow и /etc/security/passwd

Файл скрытых (теневых) паролей доступен для чтения только суперпользователю и предназначен для хранения зашифрованных паролей подальше от любопытных глаз. В нем также содержится учетная информация, которая отсутствует в исходном файле **/etc/passwd**. В настоящее время механизм скрытых паролей используется по умолчанию практически во всех системах.

В IBM-системах файл, в котором хранятся зашифрованные пароли, называется **/etc/security/passwd**, в других системах для этой цели используется файл **/etc**

[5] Используйте команду ***chsec*** для замены файлов в каталоге ***/etc/security***, вместо того чтобы редактировать их напрямую.

**/shadow**. Эти файлы различаются по формату. Сначала рассмотрим содержимое файла **/etc/shadow**.

Файл **shadow** не включает в себя файл **passwd**, и последний не генерируется автоматически при изменении файла **shadow**. Оба файла необходимо хранить независимо друг от друга (или использовать команду наподобие **useradd** для автоматического управления файлами). Как и **/etc/passwd**, файл **/etc/shadow** содержит одну строку для каждого пользователя. Каждая строка состоит из 9 полей, разделенных двоеточиями:

- регистрационное имя;
- зашифрованный пароль;
- дата последнего изменения пароля;
- минимальное число дней между изменениями пароля;
- максимальное число дней между изменениями пароля;
- количество дней до истечения срока действия пароля, когда выдается предупреждение;
- количество дней по истечении срока действия пароля, когда учетная запись аннулируется (в Linux); количество дней до автоматического аннулирования учетной записи (в Solaris/HP-UX);
- срок действия учетной записи;
- зарезервированное поле, которое в настоящее время всегда пустое (за исключением Solaris).

Лишь первые два поля обязательно должны быть заполнены. Поля дат в файле **/etc/shadow** задаются в виде числа дней (а не секунд), прошедших с 1-го января 1970 года, что не является стандартным способом вычисления времени в системах UNIX и Linux. При этом вы можете преобразовать секунды в дни с помощью такой команды.

```
solaris$ expr `date +%s` / 86400[6]
```

Типичная запись выглядит так.

```
millert:$md5$em5J8hL$a$iQ3pXe0sakdRaRFyy7Ppj. :14469:0:180:14: : :
```

Вот более подробное описание каждого поля.

- Регистрационное имя берется из файла **/etc/passwd**. Оно связывает записи файлов **passwd** и **shadow**.
- Зашифрованный пароль идентичен тому, который ранее хранился в файле **/etc/passwd**; отображается фиктивный пароль Solaris MD5.
- Третье поле содержит дату последнего изменения пользователем своего пароля.
- Это поле обычно заполняется командой **passwd**.
- В четвертом поле задано количество дней, спустя которые пользователь сможет снова изменить пароль. Это не позволяет пользователям немедленно возвращаться к привычным паролям после их обязательного изменения. Такая возможность кажется нам опасной, если в течение указанного времени будет обнаружено нарушение безопасности системы. Поэтому мы рекомендуем устанавливать данное поле равным нулю.
- В пятом поле задано максимальное число дней между двумя изменениями пароля. С помощью этой установки администраторы заставляют пользователей менять

[6] В сутках содержится 86 400 секунд: 60 × *60* × *24*.

свои пароли (подробнее механизм устаревания паролей описан в разделе 22.4). В системе Linux максимальное время жизни пароля определяется суммой значений данного и седьмого (льготный период) полей.

- В шестом поле задано количество дней, оставшихся до момента устаревания пароля, когда программа **login** должна предупреждать пользователя о необходимости сменить пароль.
- Системы Solaris и HP-UX отличаются от системы Linux их интерпретацией седьмого поля. В системе Linux седьмое поле определяет, через сколько дней после устаревания пароля отменяется учетная запись.

  В системах Solaris/HP-UX ситуация такова. Если пользователь не зарегистрировался в течение определенного числа дней, заданных в седьмом поле, учетная запись блокируется. Неиспользуемые учетные записи — любимая мишень хакеров, и здесь предусмотрена возможность объявить такие учетные записи устарелыми ("off the market"). Однако она действует только в том случае, если пользователь "прописан" в файле **/var/adm/lastlog**; те же пользователи, которые ни разу не зарегистрировались, автоматически не блокируются. Следовательно, эта функция реально не работает в сетевой среде, поскольку каждый узел имеет собственный файл **lastlog**.
- В восьмом поле задана дата отмены учетной записи. По прошествии этой даты пользователь не сможет зарегистрироваться в системе, пока администратор не сбросит значение поля. Если поле оставлено пустым, учетная запись всегда будет активной.

  В системе Linux, чтобы установить поле даты истечения срока, можно использовать команду **usermod** (даты здесь должны быть в формате гггг-мм-чч). В системе Solaris команда **usermod** также вычисляет дни, прошедшие с 1-го января 1970 года. Даты здесь принимаются примерно в 30 форматах, которые заданы в файле **/etc/datemsk**, но, увы, не в Linux-формате гггг-мм-чч.
- В системах Linux и HP-UX девятое поле зарезервировано на будущее, но в системе Solaris последние четыре бита используются для подсчета неудачных попыток регистрации.

Теперь, когда назначение полей понятно, вернемся к рассмотренному выше примеру.

```
millert:$md5$em5J8hL$a$iQ3pXe0sakdRaRFyy7Ppj. :14469:0:180:14: : :
```

В этой записи говорится о том, что пользователь `millert` последний раз менял свой пароль 13 августа 2009 года. Следующий раз пароль должен быть изменен через 180 дней. За две недели до этого пользователь начнет получать предупреждения о необходимости сменить пароль. Учетная запись не имеет срока действия.

В системах Solaris, HP-UX и Linux с помощью утилиты **pwconv** можно согласовать содержимое файлов **shadow** и **passwd**. В Linux большинство параметров файла **shadow** берется из стандартных установок файла **/etc/login.defs**.

Суперпользователь в системе Solaris может использовать команду **passwd -f** *имя_пользователя*, чтобы заставить пользователя изменить его (или ее) пароль во время следующей регистрации. Это средство полезно в том случае, если вы регулярно запускаете утилиту `crack` для выявления неудачных (ненадежных) паролей. (В Linux тот же самый флаг **-f** позволяет пользователям изменить информацию о себе.)

В системе AIX не используется термин "теневые пароли", но на практике их идея успешно реализована. Зашифрованные AIX-пароли хранятся в файле **/etc/security/passwd**, причем их формат совершенно отличается от формата файла **/etc/passwd**. Рассмотрим пример из исходной инсталляции системы AIX, где в качестве стандартного алгоритма генерирования паролей используется **crypt**[7].

```
trent:
    password = u10.OaYxRx4qI
    lastupdate = 1224876639
    flags = ADMCHG

evi:
    password = PiIr2qOPabZ.Q
    lastupdate = 1235785246
    flags =
```

Формат должен быть самоочевиден. Здесь записи разделяются одной или несколькими пустыми строками. Этот же формат используется для большинства AIX-файлов конфигурации в каталоге **/etc/security** при использовании имени пользователя, общего для любого управляемого или регистрируемого объекта.

В системе AIX предусмотрено множество возможностей для управления всеми аспектами процесса регистрации (включая генерирование паролей). В некоторых случаях результат зависит от конкретного пользователя и порта (при управлении портами TTY, на которых может зарегистрироваться данный пользователь). Более подробную информацию можно получить, просмотрев комментарии в файлах **/etc/security/login.cfg** и **/etc/security/user**. Удобно также использовать команду `pwdadm`, которая позволяет заставить пользователя изменить его пароль при следующей регистрации.

## 7.3. Файл /etc/group

Файл **/etc/group** содержит имена UNIX-групп и списки членов каждой группы. Вот как выглядит фрагмент файла **group** из системы AIX.

```
system: ! :0:root,pconsole,esaadmin
staff: ! :1: ipsec,esaadmin,trent,ben,garth,evi
bin: ! :2:root,bin
sys:!:3:root,bin,sys
adm: ! :4:bin,adm
nobody: ! :4294967294:nobody,lpd
```

Каждая строка представляет одну группу и содержит следующие четыре поля:

- имя группы;
- зашифрованный пароль или заполнитель;
- идентификатор группы;
- список членов, разделенный запятыми (пробелов быть не должно).

Как и в файле **/etc/passwd**, поля разделяются двоеточиями. Длина имени группы не должна превышать 8 символов из соображений совместимости, хотя во многих системах такого ограничения нет. Несмотря на наличие поля пароля (с помощью кото-

[7] Для новых версий дистрибутивов AIX первым в списке задач первоочередной важности необходимо поставить переход к более сильному алгоритму шифрования.

рого пользователи могут присоединиться к группе, выполнив команду **newgrp**) пароль задается редко[8]. Пароль можно установить с помощью команды **gpasswd**; а его зашифрованная форма сохранится в файле **/etc/gshadow**. Групповые пароли используются довольно редко.

Имена групп и их идентификаторы должны быть одинаковыми на всех компьютерах, получающих совместный доступ к файлам через систему NFS. Этого трудно достичь в гетерогенной среде, поскольку в разных операционных системах используются разные идентификаторы для одних и тех же групп.

Мы считаем, что по умолчанию пользователь не должен регистрироваться как член системной группы. В некоторых системах групповое право собственности используется вместе с битами прав доступа для управления характером выполнения команд. Противоречия в использовании групповых идентификаторов в различных системах создают проблемы при обслуживании узлов, а именно при обновлении и инсталляции программных продуктов.

Если в файле **/etc/passwd** пользователь объявлен членом определенной группы, а в файле **/etc/group** — нет, пользователь все равно включается в группу. На этапе регистрации членство в группе проверяется по обоим файлам, но лучше все же согласовывать их содержимое.

В некоторых системах ограничивается количество групп, к которым может принадлежать пользователь. В качестве предельного значения обычно используется число 8, но в Solaris оно сейчас равно 16, в HP-UX — 20, а в AIX и Linux ограничений нет вообще.

Для того чтобы избежать конфликтов, связанных с идентификаторами стандартных групп, рекомендуем выбирать идентификаторы локальных групп начиная со значения 500.

Согласно традиции UNIX, новые пользователи добавляются в группу, название которой отражает их категорию, например "students" или "finance". Однако следует учитывать, что подобная традиция повышает вероятность доступа пользователей к файлам друг друга из-за неаккуратной установки прав. Чтобы этого избежать, рекомендуем создавать для каждого пользователя уникальную группу. Лучше всего выбирать одинаковое имя для пользователя и группы. Можно также выбрать одинаковый номер пользователя и группы.

Утилита **useradd** всех выбранных в качестве примеров дистрибутивов Linux, за исключением SUSE, по умолчанию помещает пользователей в персональные группы. Системы семейства UNIX по умолчанию помещают всех новых пользователей в одну и ту же группу, но их утилиты **useradd** также можно сконфигурировать на поддержку персональных групп.

В такую персональную группу должен входить лишь сам пользователь. Если нужно позволить другим пользователям совместно работать с файлами посредством группового владения, создайте для этих целей отдельную группу. Идея персональных групп состоит не в том, чтобы запретить механизм группового доступа как таковой, а в том, чтобы создать более строгую стандартную группу для каждого пользователя и предотвратить случайный коллективный доступ к файлам. Этой цели можно также достичь с помощью команды **umask** (см. раздел 6.5).

В системах Linux, Solaris и HP-UX поддерживаются команды, которые создают, модифицируют и удаляют группы: **groupadd**, **groupmod**, **groupdel**. В системе AIX вам для

[8] Для того чтобы установить групповой пароль в системе Solaris, необходимо выполнить команду ***passwd***, а также команды вырезки и вставки пароля в файл ***/etc/group***. Здесь не используется файл ***/etc/gshadow*** или его эквивалент.

этого придется модифицировать файл **/etc/group** с помощью текстового редактора. Но для проверки синтаксиса в этом файле предусмотрена команда **grpck**.

## 7.4. Подключение пользователей: основные действия

Прежде чем создавать учетную запись для нового пользователя в правительственном учреждении, учебном заведении или в крупной коммерческой компании, крайне важно заставить его подписать соглашение о правилах работы в системе. (Как?! У вас нет такого соглашения? Немедленно прочтите материал в разделе 32.7, чтобы узнать, для чего оно нужно и как его составлять.)

У пользователей нет веских причин подписывать такое соглашение, поэтому в интересах администратора убедить их сделать это, чтобы иметь рычаги влияния. После того как учетная запись создана, добиться подписи будет трудно, так что лучше получить ее заранее.

Процесс подключения нового пользователя состоит из ряда этапов, определяемых системными требованиями; два связаны с формированием пользовательской среды, а еще несколько этапов может понадобиться для целей системного администрирования.

- Обязательные этапы:
  - добиться, чтобы пользователь подписал соглашение о правилах работы в системе;
  - отредактировать файлы **passwd** и **shadow** с целью создания учетной записи пользователя;
  - добавить запись нового пользователя в файл **/etc/group** (в действительности это не необходимо, а желательно);
  - задать пароль;
  - создать для нового пользователя домашний каталог, назначив его владельца с помощью команды **chown** и задав режим доступа с помощью команды **chmod**;
  - сконфигурировать роли и права доступа (если вы используете RBAC; см. чуть ниже).
- Пользовательские этапы:
  - скопировать в домашний каталог пользователя стандартные конфигурационные сценарии;
  - создать каталог электронной почты и настроить почтовые псевдонимы.
- Административные этапы:
  - проверить правильность создания учетной записи;
  - добавить контактную информацию о пользователе, а также сведения о статусе учетной записи в административную базу данных.

Для выполнения этого списка действий необходимо иметь сценарий или утилиту, и, к счастью, во всех наших примерах систем предусмотрена таковая в форме команды **useradd**.

Все описываемые операции по добавлению пользователей необходимо выполнять с правами суперпользователя. В системе AIX вы должны иметь привилегии UserAdmin. Именно в этом случае наступает звездный час для использования утилиты **sudo**. Об утилите **sudo** рассказывалось в разделе 4.3.

## Редактирование файлов passwd и group

Если вам придется добавлять пользователя вручную, используйте команду **vipw**, чтобы отредактировать файлы **passwd** и **shadow**. По умолчанию выбирается редактор **vi**, но эту установку можно изменить, задав новое значение переменной среды EDITOR. Более важно то, что команда **vipw** блокирует редактируемый файл, позволяя только одному пользователю редактировать его. Это не допускает никаких конфликтов между действиями разных пользователей.

В системах Solaris и Red Hat команда **vipw** (после редактирования файла **passwd**) автоматически интересуется желанием пользователя редактировать файл **shadow**. В системах SUSE и Ubuntu для этого используйте команду **vipw -s**.

В системах HP-UX и AIX рекомендуется не редактировать файл **password** вручную — с помощью команды **vipw** или без нее (в системе AIX соответствующая утилита даже не инсталлирована). Здесь лучше использовать утилиту **useradd** или специальные инструменты системных администраторов **smh** и SMIT соответственно. Утилита **useradd** рассматривается в разделе 7.5.

Если новый пользователь должен стать членом нескольких групп, а не просто одной группы, заданной по умолчанию в файле **passwd**, вам необходимо отредактировать файл **/etc/group** и добавить регистрационное имя пользователя во все дополнительные группы.

## Задание пароля

🕮 Правила выбора удачных паролей приведены в разделе 4.3.

Никогда не оставляйте без пароля новую учетную запись или запись с доступом к интерпретатору команд. Определенная сложность работы с паролями может быть связана с файлами конфигурации (см. разделы в конце этой главы, чтобы узнать, какие именно файлы и переменные применяются в ваших операционных системах). Установите пароль для нового пользователя с помощью следующей команды.

```
$ sudo пароль новое_имя_пользователя
```

Вам будет предложено ввести реальный пароль. В некоторых автоматизированных системах при добавлении новых пользователей необязательно вводить начальный пароль. Задание пароля в таких случаях происходит при первой регистрации. Несмотря на удобство, этот вариант приводит к образованию огромной дыры в системе безопасности; тот, кто может угадать новые регистрационные имена (или узнать их в файле **/etc/passwd**), может выполнить "пиратские" действия по отношению к учетным записям, прежде чем соответствующие пользователи получат возможность зарегистрироваться.

## Создание домашнего каталога пользователя и инсталляция конфигурационных файлов

Вы можете создать домашний каталог пользователя с помощью простой команды **mkdir**. Вам также понадобится установить права владения на новый каталог и права доступа к нему, но лучше всего сделать это после инсталляции локальных конфигурационных файлов.

Имена таких файлов традиционно начинаются с точки и заканчиваются суффиксом **rc** (сокращение от "run command" — команда запуска) — "отголосок" операционной системы CTSS. Предшествующая точка заставляет команду **ls** не включать эти файлы

в листинги каталогов, если только не указана опция **-a**. Мы рекомендуем предоставлять стандартные файлы запуска для всех интерпретаторов команд, которые популярны в ваших системах, чтобы пользователи по умолчанию продолжали работать в корректной среде даже при смене командной оболочки. Наиболее часто встречающиеся конфигурационные файлы перечислены в табл. 7.3.

**Таблица 7.3. Распространенные конфигурационные файлы**

| Команда | Имя файла | Применение |
|---|---|---|
| **sh** | **.profile** | Установка пути поиска, типа терминала и среды |
| **basha** | **.bashrc** | Установка типа терминала (если это необходимо)<br>Установка опций программ **biff** и **mesg** |
| | **.bash_profile** | Установка переменных среды<br>Установка псевдонимов команд<br>Установка пути поиска<br>Установка атрибута **umask** для управления правами доступа<br>Установка переменной CDPATH для поиска имен файлов<br>Установка переменных PS1 (строка приглашения) и HISTCONTROL |
| **csh/tcsh** | **.login** | Аналог файла **.cshrc** для интерпретатора **csh** |
| | **.cshrc** | Аналог файла **.login** для интерпретатора **csh** |
| **vi/vim** | **.exrc/.vimrc** | Установка опций редакторов **vi/vim** |
| **emacs** | **.emacs** | Установка опций редактора **emacs** и функциональная привязка его клавиш |
| **mail/mailx** | **.mailrc** | Задание персональных почтовых псевдонимов<br>Установка параметров оригинального почтового UNIX-клиента |
| **GNOME** | **.gconf** | Среда GNOME: конфигурация среды пользователя посредством команды **gconf** |
| | **.gconfpath** | Путь для дополнительного варианта конфигурации среды пользователя посредством команды **gconf** |
| **KDE** | **.kde/** | Среда KDE: каталог файлов конфигурации |

[a] Интерпретатор **bash** также читает файл **.profile** или **/etc/profile** в эмуляции оболочки команд **sh**.

Образцы файлов запуска традиционно хранятся в каталоге **/etc/skel** (Linux, Solaris, HP-UX) или **/etc** (все системы). Система AIX, как всегда отличаясь от других, использует для этого каталог **/etc/security**. Если вы будете редактировать примеры конфигурационных файлов, каталог **/usr/local/etc/skel** — самое удачное место для хранения модифицированных копий. Система Linux также хранит важные детали файлов запуска в каталоге **/etc/profile.d**, в который "заглядывают" интерпретаторы команд, чтобы отыскать указатели на параметры вывода результатов команды **ls** (чтобы сделать их нечитабельными на темном фоне) или путь к исполняемым Kerberos-файлам.

В зависимости от интерпретатора, в каталоге **/etc** могут содержаться системные конфигурационные файлы, обрабатываемые раньше пользовательских. Например, интерпретатор **bash** читает файл **/etc/profile** до начала обработки файла **~/.profile** и **~/.bash_profile**. В эти файлы стоит помещать стандартные установки, необходимые для функционирования вашего узла, но следует иметь в виду, что пользователи могут переопределять свои установки в собственных конфигурационных файлах. Информацию о других интерпретаторах можно найти в документации (см. соответствующие **man**-страницы).

Не забудьте задать разумное значение **umask** (рекомендуем 077, 027 или 022, в зависимости от "дружелюбности" среды и размеров организации). Если вы не используете индивидуальные группы, рекомендуем установить для **umask** значение 077, поскольку оно предоставляет владельцу полный доступ, а для группы и вообще для всех остальных — никакого доступа. Подробнее о **umask** см. раздел 6.5.

Конфигурационные файлы и каталоги, перечисленные для настольных сред GNOME и KDE, представляют собой лишь вершину айсберга. Утилита **gconf** предназначена для хранения предпочтений в выборе приложений для программ под управлением GNOME (подобно тому, как это реализовано в системном реестре Windows).

## Установка прав доступа и прав собственности

После установки домашнего каталога переходите к пользователю и убедитесь, что он имеет соответствующие права доступа. Команда

```
$ sudo chown -R новый_пользователь:новая_группа ~новый_пользователь
```

должна надлежащим образом установить права собственности. Обратите внимание на невозможность использования команды

```
$ sudo chown новый_пользователь:новая_группа ~новый_пользователь/.*
```

применительно к файлам, имена которых начинаются с точки, поскольку *новый_пользователь* станет владельцем не только собственных файлов, но и родительского каталога "`..`" (например, `/home`). Это распространенная и опасная ошибка.

## Назначение каталога для электронной почты

Пользователи зачастую предпочитают получать почту на одном компьютере. Эта схема часто реализуется с помощью записи в файле глобальных псевдонимов **/etc/mail/aliases** или установки параметра `userDB` программы **sendmail** на центральном почтовом сервере. Об электронной почте можно прочитать в главе 20.

## Конфигурирование ролей и административных привилегий

Управление доступом на основе ролей (role-based access control — RBAC) позволяет "подогнать" системные права к отдельным пользователям и обеспечивается на многих наших примерах систем. Политика RBAC не является традиционной частью модели управления доступом UNIX или Linux, но если на вашем узле она используется, то конфигурация ролей должна быть частью процесса добавления пользователей. Модель RBAC подробно описана в разделе 4.2.

🕮 Подробнее о SOX и GLBA написано в главе 32.

Принятые в США законы Сарбейнза-Оксли (Sarbanes-Oxley Act) и Грэма-Лич-Блайли (Gramm-Leach-Bliley Act), предназначенные, в частности, для защиты информации заказчика, полученной или используемой финансовыми организациями США, от кражи, неавторизованного доступа или злоупотребления, усложнили многие аспекты системного администрирования в корпоративной сфере, включая управление пользователями. Поэтому использование ролей может быть вашим единственным жизнеспособным вариантом, позволяющим выполнить некоторые требования, устанавливаемые законами SOX/GLBA.

## Заключительные действия

Для того чтобы удостовериться, что новая учетная запись сконфигурирована надлежащим образом, завершите сеанс работы (т.е. выйдите из системы), а затем снова войдите в систему под именем нового пользователя и выполните следующие команды.

```
$ pwd /* для проверки домашнего каталога */
$ ls -la /* для проверки владельца/группы файлов конфигурации */
```

Вам необходимо уведомить новых пользователей об их регистрационных именах и начальных паролях. Многие узлы отправляют эту информацию по электронной почте, но из соображений безопасности это не приветствуется. Делайте это при личном контакте или по телефону. Но если вам приходится добавлять сразу 500 новичков на университетские компьютеры CS-1, переложите эту задачу на преподавателей! Именно в это время стоит напомнить пользователям о пользе ознакомления с дополнительной документацией на предмет местных обычаев, если таковые имеют место.

🕮 Подробнее о заключении контрактов с пользователями можно прочитать в разделе 32.10.

Если на вашем узле требуется, чтобы пользователи подписывали соответствующее соглашение, убедитесь в реализации этого действия до создания их учетных записей. Такая проверка предотвратит возможные упущения и усилит правовую основу санкций, которые, возможно, вам придется позже применить.

Напомните новым пользователям о необходимости немедленного изменения их паролей. При желании это можно обеспечить путем установки действия паролей в течение некоторого короткого времени. Еще один вариант состоит в создании сценария, который бы проверял новых пользователей и удостоверялся в том, что их зашифрованные пароли в файле **shadow** были изменены[9]. Если вы знаете пользователей лично, то в такой среде относительно несложно проследить, кто использует систему и с какой целью. Но если в подчинении администратора большой и часто меняющийся штат пользователей, необходим более формальный способ контроля учетных записей. Ведение базы данных с контактной информацией и сведениями о статусе учетных записей поможет, в случае необходимости, легко выяснить, кто есть кто и зачем тому или иному пользователю понадобилась учетная запись.

# 7.5. Добавление пользователей с помощью программы useradd

Во всех системах программа `useradd` реализует одну и ту же базовую процедуру, описанную выше. Но ее можно сконфигурировать под конкретную среду. Поскольку в каждой системе имеются свои представления о том, что именно следует настраивать, где следует реализовать настройки и каково должно быть стандартное поведение, мы описываем эти подробности в соответствующих разделах.

В табл. 7.4 приведены команды и файлы конфигурации, имеющие отношение к управлению пользователями. Все рассматриваемые нами системы имеют "джентльменские" наборы команд для управления пользователями (как правило, к ним относят-

[9] Поскольку один и тот же пароль может иметь множество зашифрованных представлений, этот метод проверяет только сам факт изменения пользователем своего пароля, а не то, что пароль реально был заменен другим паролем.

ся команды **useradd**, **usermod** и **userdel**). Поскольку все эти команды сконфигурированы одинаково, рассмотрим только команду `useradd` и дополним ее запись другими командами, в зависимости от конкретной системы.

Обратите внимание на то, что, несмотря на одинаковое имя (**useradd**), сами утилиты реализованы в разных системах по-разному.

**Таблица 7.4. Команды и файлы конфигурации для управления пользователями**

| Система | Команды | Конфигурационные файлы | Комментарии |
|---|---|---|---|
| Ubuntu | **useradd** | **/etc/login.defs** | |
| | **adduser** | **/etc/default/useradd** | |
| | | **/etc/adduser.conf** | Более дружественная Perl-версия |
| SUSE | **useradd** | **/etc/login.defs** | |
| | | **/etc/default/useradd** | |
| | | **/etc/default/passwd** | |
| | | **/usr/sbin/useradd.local** | Для локальных настроек |
| | | **/usr/sbin/userdel.local** | Для локальных настроек |
| | | **/usr/sbin/userdel-pre.local** | Для локальных настроек |
| | | **/usr/sbin/userdel-post.local** | Для локальных настроек |
| Red Hat | **useradd** | **/etc/login.defs** | |
| | | **/etc/default/useradd** | |
| Solaris | **useradd** | **/etc/default/{login,passwd}** | |
| | | **/etc/security/policy.conf** | |
| HP-UX | **useradd** | **/etc/default/useradd** | GUI-инструмент, также именуемый **sam** |
| | | **/etc/default/security** | |
| | **smh** | | |
| AIX | **useradd** | **/etc/security/user** | |
| | | **/etc/security/login.cfg** | |
| | | **/etc/security/mkuser.defaulta** | |
| | **mkuser** | | Вызывается **useradd** |
| | **chuser** | | Вызывается **usermod** |
| | **rmuser** | | Вызывается **userdel** |
| | **SMIT** | | GUI-инструмент |

[a] В старых системах AIX этот файл находится в каталоге **/usr/lib/security**.

## Команда useradd в системе Ubuntu

В системе Ubuntu предусмотрено два способа добавления пользователей: **adduser** и **useradd**. Команда **adduser** представляет собой Perl-оболочку для команды **useradd**, что увеличивает ее "коэффициент полезного действия", позволяя создавать домашние каталоги, копировать файлы конфигурации и пр.

Команда **adduser** конфигурируется в файле **/etc/adduser.conf** по таким направлениям:

- правила размещения домашних каталогов: по группе, по имени пользователя и пр.;
- установки прав доступа для новых домашних каталогов;
- диапазоны индивидуальных и групповых идентификаторов (UID и GID) для системных и общих пользователей;
- возможность создания индивидуальной группы для каждого пользователя;
- дисковые квоты (к сожалению, с использованием только типа **Boolean**);

- сопоставление имен пользователей и имен групп с заданными регулярными выражениями.

Другие типы параметров команды **useradd** (такие, как правила создания паролей) устанавливаются как параметры, передаваемые модулю PAM, который выполняет регулярную аутентификацию паролей. О подключаемых модулях аутентификации (Pluggable Authentication Modules — PAM) можно прочитать в разделе 22.5. У команды **adduser** есть "близнец" **addgroup** и "кузины" **deluser** и **delgroup**.

## Команда useradd в системе SUSE

В системе SUSE с помощью команды **useradd** по умолчанию не создается домашний каталог для нового пользователя и не копируются конфигурационные файлы. Эти дополнения придется "заказать" с помощью флага **-m**. (Файлы запуска находятся в каталоге **/etc/skel**.) Команда **useradd** не создает и почтовый файл спулинга для новых пользователей.

Кроме того, SUSE-команда **useradd** не создает пользовательские группы; по умолчанию идентификатор группы (GID) в файле **passwd** задается с помощью переменной `GROUP` в файле **/etc/default/useradd**. Новые пользователи также добавляются в группы, задаваемые переменной `GROUPS`. По умолчанию это группы `video` и `dialout`, они предоставляют доступ к системному буферу кадров и программам, обеспечивающим работу с сетью по коммутируемым телефонным каналам (**pppd**).

Для того чтобы компенсировать эти недостатки, SUSE-команда **useradd** вызывает **bash**-сценарий **/usr/sbin/useradd.local**, в который можно добавить нужные вам настройки.

Файл **/etc/login.defs** в SUSE управляет следующими аспектами:

- разрешать ли регистрацию, если домашний каталог пользователя не существует;
- степень устойчивости (задержка и блокировка) для неудачных попыток регистрации;
- местоположение "сообщения дня" и файлов **ttytype** (ввод с терминала);
- ограничения на использование **chsh** и **chfn**;
- контроль за сроками действия паролей;
- диапазоны системных, пользовательских и групповых идентификаторов;
- правила формирования допустимых имен пользователей и групп;
- пользовательские значения **umask** (по умолчанию оно равно 022);
- локальные сценарии, которые "монтируются" в команды **useradd** и **userdel**;

Различные параметры и их значения достаточно полно описаны в `man`-странице для файла `login.defs` и комментариях в самом файле.

## Команда useradd в системе Red Hat

Программа **useradd** в системе Red Hat Enterprise Linux принимает параметры конфигурации из файла **/etc/login.defs**, в котором хранятся результаты решения таких задач, как контроль устаревших паролей, алгоритмы шифрования, почтовые файлы спулинга и диапазоны идентификаторов UID/GID. Назначение различных параметров разъясняется с помощью комментариев, содержащихся в этом файле.

С помощью команды **useradd -D** можно отобразить значения, которые используются командой **useradd** для регистрации новых пользователей по умолчанию. Эти стандартные значения устанавливаются в файле **/etc/default/useradd**. В системе Red Hat новые пользователи включаются в собственные индивидуальные группы. В файле **passwd** символ "x" используется в качестве заполнителя пароля, а в файле **shadow** символ "!!" работает как код, который запрещает регистрацию и требует, чтобы системный администратор установил пароль для нового пользователя. По умолчанию используется алгоритм шифрования MD5. Домашний каталог нового пользователя заполняется конфигурационными файлами, "взятыми" из каталога **/etc/skel**.

## Команда useradd в системе Solaris

В системе Solaris одна часть стандартных параметров, связанных с регистрационными именами и паролями, хранится в каталогах **/etc/default/login** и **/etc/default/passwd**; другая — встроена в саму команду **useradd**. Команда **useradd -D** позволяет отобразить стандартные значения некоторых параметров. Используя дополнительные флаги, эти стандартные значения можно восстановить.

Формат файлов **default/login** и **default/passwd** подобен формату Linux-файла **login.defs** в том, что пустые строки и строки, начинающиеся с символа "#", игнорируются, а каждая строка, не являющаяся комментарием, назначает переменной некоторое значение. Однако синтаксические строки

```
NAME=значение
```

и

```
NAME <пробельный символ> значение
```

не эквивалентны.

Файл **/etc/default/passwd** позволяет управлять следующими параметрами:

- минимальная длина пароля;
- срок действия пароля;
- необходимая степень сложности пароля;
- контроль взломанных паролей.

Файл **/etc/default/login** позволяет управлять следующими аспектами:

- часовой пояс;
- предельные значения размеров файлов, которые может создавать пользователь;
- может ли суперпользователь регистрироваться только на консоли;
- для каждого ли пользователя требуется пароль;
- обработка неудачных попыток регистрации;
- начальный путь поиска пользователя;
- стандартное значение **umask** для пользователя (по умолчанию устанавливается равным 022);
- протоколировать ли действия суперпользователя и неудачные попытки регистрации посредством утилиты **syslog**.

Файлы **/etc/security/policy.conf** и **/etc/security/crypt.conf** определяют алгоритмы шифрования, которые можно использовать для паролей.

## Команда useradd в системе HP-UX

По умолчанию HP-UX-команда **useradd** не позволяет создавать домашние каталоги или включать пользователя в индивидуальную группу. Однако с помощью опции **-m** домашние каталоги можно создавать, а также заполнять их файлами запуска из каталога **/etc/skel**.

Параметры конфигурации для команды **useradd** устанавливаются в файлах **/etc/default/useradd** и **/etc/default/security**, причем файл **useradd** принимает параметры в стиле Linux (NAME <пробел> *значение*), а файл **security** — в стиле Solaris (NAME=*значение*). Синтаксис должен быть ясен по другим записям в каждом файле, но если вы примените неправильный формат, переменная, которую вы попытаетесь установить, будет молча проигнорирована. Никакие сообщения о синтаксических ошибках при этом не генерируются.

Файл **/etc/default/useradd** управляет следующими опциями:

- стандартная группа и командная оболочка;
- корень дерева домашнего каталога;
- истечение срока действия учетной записи;
- создавать ли домашние каталоги;
- разрешать ли дублированные идентификаторы пользователя.

Файл **/etc/default/security** содержит дополнительные параметры конфигурации, и некоторые из них имеют отношение к управлению пользователями:

- разрешать ли регистрационные имена с опущенным домашним каталогом;
- разрешать ли нулевые (пустые) пароли;
- минимальная длина пароля;
- обработка неудачных попыток регистрации;
- обработка неактивных учетных записей;
- стандартное значение **umask** для новых пользователей (по умолчанию 027).

По именам переменных в этом файле несложно понять их назначение.

## Команда useradd в системе AIX

В системе AIX программа **useradd** представляет собой всего лишь **ksh**-оболочку для команды **mkuser**. Подобным образом команда **usermod** вызывает программу **chuser**, а команда **userdel** — программу **rmuser**. Для этих команд имеются соответствующие man-страницы, которые можно найти по их исходным именам.

Файлы конфигурации, управляющие многочисленными аспектами регистрационных имен и паролей, хранятся в каталоге **/etc/security**. Существует три файла: **login.cfg**, **user** и **mkuser.default**. В первых двух файлах символ "*" используется в качестве символа комментария; а третий файл комментариев не имеет. Эти файлы организованы с использованием следующей формы.

```
метка:
атрибут = значение
следующий_атрибут = значение
следующая_метка:
атрибут = значение
```

Например, в файле **/etc/security/user** элементы *метка* представляют собой имена пользователей (или слово **default**), а список возможных атрибутов приведен в табл. 7.5.

**Таблица 7.5. Опции управления учетными записями пользователей в файле /etc/security/user (AIX)**

| Опция | Тип | Значение |
|---|---|---|
| account_locked | Булев | Блокирует регистрационное имя, если равен **true** |
| admin | Булев | Предоставляет права администратора, если равен **true** |
| auth1 | Список методов | Основной метод аутентификации |
| auth2 | Список методов | Второстепенный метод аутентификации |
| dictionlist | Имена файлов | Словари, которые должны включать пароли |
| expires | Дата | Дата истечения срока действия учетной записи пользователя |
| histexpire | Недели | Период, когда пользователь не может повторно использовать пароль |
| histsize | Число | Количество предыдущих значений пароля, которое не может быть повторно использовано |
| ogin | Булев | Можно зарегистрировать? Подходит для таких регистрационных имен, bin |
| oginretries | Число | Количество попыток регистрации до блокирования учетной записи |
| ogintimes | Временной диапазон | Временные границы, когда пользователь может зарегистрироваться |
| maxexpired | Недели | Льготный период для недействительных паролей |
| maxrepeats | Число | Количество раз, которое символ может появиться в пароле |
| minage, maxage | Недели | Минимальный и максимальный "возраст" пароля |
| minalpha | Число | Минимальное количество буквенных символов в пароле |
| mindiff | Символы | Количество символов старого пароля, разрешенных в новом пароле |
| minlen | Число | Минимальная длина пароля (не устанавливайте его равным 0) |
| minother | Число | Минимальное количество небуквенных символов в пароле |
| pwdchecks | Имена файлов | Функции, вызываемые для проверки безопасности паролей |
| pwdwarntime | Дни | Льготный период для предупреждения пользователя об изменении пароля |
| rlogin | Булев | Может ли использовать протокол **rlogin** или **telnet** для данной учетной записи? |
| su | Булев | Могут ли другие пользователи использовать **su** для данной учетной записи? |
| ttys | Список устройств | Терминалы, с которых данный пользователь может регистрироваться |
| umask | Восьмеричный | Стандартные права доступа для файлов, созданных пользователем |

Ну и ну, что за список! В комментариях файла указываются значения по умолчанию, которые вполне корректны для инсталляции с низким уровнем безопасности. Мы рекомендуем некоторые параметры изменить:

- значение umask с 022 на 077;

- значение `loginretries` с 0 (неограниченное) на небольшое целое, скажем, число 5;
- значение `minlen` с 0 (нет пароля) на хотя бы число 6 или 7;
- значение `expires` с 0 (никогда) на год (только в случае, если у вас есть инструмент периодического обновления дат истечения срока для действительных пользователей).

🕮 Подробнее о PAM рассказывается в разделе 22.5.

К сожалению, это лишь один из файлов конфигурации, который управляет процессом регистрации нового пользователя. Файл **`/etc/security/login.cfg`** содержит параметры управления дефектными регистрационными именами (время задержки между приглашениями ввести имя и пароль, количество дефектных регистрационных имен, разрешаемых до блокирования учетной записи, длительность блокирования учетной записи, время ее восстановления и прочее), значения моментов времени, когда регистрационные имена разрешаются, вид запроса на ввод пароля пользователя, список действующих интерпретаторов команд, максимально допустимое количество одновременных регистраций, длительность времени ожидания пароля пользователя, тип авторизации регистрационного имени (т.е. где бы вы задали PAM[10], если бы должны были его использовать) и алгоритм шифрования паролей (по умолчанию `crypt`). Кроме того, чтобы запутать вас еще больше, некоторые параметры задаются в двух файлах, иногда с одинаковыми именами (например, `logintimes`), а иногда — с разными (`loginretries` и `logindisable`). Наверное, чтобы жизнь медом не казалась!

Команда **`useradd`** в системе AIX не имеет опции **`-D`**, которая отображает стандартные значения для новых пользователей. Она помещает новых пользователей в одну группу и не создает домашние каталоги, если не был указан флаг **`-m`** (в этом случае выполняется копирование данных в файл **`.profile`** из каталога **`/etc/security`**).

## Пример использования команды useradd

Для того чтобы создать в системе Linux новую учетную запись "hilbert" стандартными системными средствами, можно просто выполнить следующую команду.

```
$ sudo useradd hilbert
```

При выполнении этой команды будет создана следующая запись в файле **`/etc/passwd`**.

```
hilbert:x:1005:20: :/home/hilbert:/bin/sh
```

Команда **`useradd`** блокирует учетную запись, помещая символ "x" в поле пароля. Чтобы сделать учетную запись используемой, ей необходимо назначить реальный пароль.

Рассмотрим более реалистичный пример. Мы хотим, чтобы основной группой пользователя `hilbert` была `faculty` и чтобы он был добавлен в группу `famous`. Для этого переопределим стандартное местоположение домашнего каталога и интерпретатор, а также велим команде **`useradd`** создать домашний каталог (если таковой еще не создан).

```
$ sudo useradd -c "David Hilbert" -d /home/math/hilbert -g faculty -G
        famous -m -s /bin/tcsh hilbert
```

Эта команда создает следующую запись **`passwd`**.

```
hilbert:x:1005:30:David Hilbert:/home/math/hilbert:/bin/tcsh
```

[10] PAM — подключаемые модули аутентификации — относительно недавнее дополнение к системе AIX, которое должно быть полностью функциональным в версии 5.3 и более поздних.

Назначенный идентификатор (UID) оказывается больше наибольшего идентификатора в системе, и соответствующая запись имеет следующий вид.

```
hilbert: ! :14322:0:99999:7:0: :
```

В файлах **passwd** и **shadow** используются разные символы-заполнители пароля, которые зависят от конкретной операционной системы. Команда **useradd** также добавит пользователя `hilbert` в соответствующую группу (в файле **/etc/group**), создаст каталог **/home/math/hilbert** и заполнит его файлами из каталога **/etc/skel**.

## 7.6. Добавление пользователей "пакетом" с помощью команды NEWUSERS (Linux)

При выполнении Linux-команды **newusers** одновременно создается несколько учетных записей из содержимого подготовленного текстового файла. Это не вполне формальный способ, но им удобно пользоваться в случае, если нужно добавить много пользователей сразу, например при создании учетных записей для учебной группы. Команда **newusers** в качестве аргумента получает входной файл, состоящий из строк, подобно файлу **/etc/passwd**, за исключением того, что его поле пароля содержит начальный пароль, заданный открытым текстом. Поэтому хорошо подумайте о защите такого файла.

Команда **newusers** принимает связанные со сроком действия пароля параметры, установленные в файле **/etc/login.defs**, но не копирует стандартные конфигурационные файлы, как это делает команда **useradd**. Единственный файл, который предоставляется в этом случае, — файл **.xauth**.

В университетских системах действительно важно применять "пакетный" сценарий **adduser**, который использует список студентов из данных регистрации для генерирования входной информации для команды **newusers**. Для этого необходимо иметь имена пользователей, подготовленные в соответствии с локальными правилами при гарантии их уникальности (на локальном уровне), а также метод случайного генерирования паролей при увеличивающихся значениях идентификаторов пользователей и групповых идентификаторов для каждого студента. Возможно, вам лучше написать собственную оболочку для команды **useradd** на языке Perl или Python, чем пытаться приспособить свои нужды к тому, что предлагает команда **newusers**.

## 7.7. Удаление пользователей

Когда пользователь покидает организацию, его учетная запись и файлы должны быть удалены из системы. Эта процедура включает удаление всех ссылок на регистрационное имя, которые были введены вручную или с помощью команды **useradd**. При ручном удалении учетной записи последовательность операций будет примерно такой:

- удалить записи пользователя из локальных баз данных и телефонных списков;
- удалить пользовательские псевдонимы из файла **aliases** либо организовать перенаправление поступающих ему сообщений;
- удалить пользовательские задания из **crontab**-файла, из очереди команды **at**, а также из заданий на печать;
- уничтожить пользовательские процессы, которые еще выполняются;

- удалить записи пользователя из файлов **passwd**, **shadow**[11], **group** и **gshadow**;
- удалить домашний каталог пользователя;
- удалить почтовый каталог пользователя;
- удалить записи в совместно используемых календарях, системах бронирования номеров и пр.
- удалить или преобразовать права владения любыми списками рассылки, используемыми удаленным пользователем.

Перед тем как удалять домашний каталог пользователя, необходимо переместить из него в другие каталоги все файлы, которые нужны остальным пользователям. Поскольку не всегда можно с уверенностью сказать, какие файлы понадобятся, а какие — нет, лучше предварительно скопировать пользовательские домашний и почтовый каталоги на какой-то носитель.

После удаления учетной записи пользователя убедитесь, что в системе не осталось файлов с его идентификатором. Чтобы узнать точный путь к этим файлам, воспользуйтесь командой **find** с аргументом **-nouser**. Поскольку команда **find** может вести поиск на сетевых серверах, проверяйте файловые системы по отдельности, задавая опцию **-xdev**.

```
$ sudo find  filesystem  -xdev -nouser
```

Прекращение выполнения процессов удаленного пользователя может оказаться проблематичным в распределенной среде. Совместно используемые календари и системы бронирования номеров могут иметь еще работающие элементы (сгенерированные более несуществующим пользователем), которые внезапно "осиротели" и должны быть удалены. Возможно, в вашей системе найдутся и другие места, подлежащие "зачистке". Имеет смысл составить собственный список, может быть, в форме соответствующего сценария.

Если в организации за пользователями закреплены отдельные рабочие станции, эффективнее всего переинсталлировать систему, прежде чем предоставлять ее новому пользователю. Но не забудьте сбросить на системный жесткий диск локальные файлы, которые могут понадобиться в будущем.

Каждая система из наших примеров имеет команду **userdel**, которая автоматизирует процесс удаления пользователя. Если она не отвечает вашим высоким требованиям, можете дополнить ее функциональные возможности предоставлением ей расширенного списка мест хранения данных о пользователях.

В системе Ubuntu команда **deluser** представляет собой Perl-сценарий, который вызывает обычную команду **userdel**, аннулируя все деяния команды **adduser**. При этом вызывается сценарий **deluser.local**, если таковой существует, чтобы реализовать несложную операцию локализации. Файл конфигурации **/etc/deluser.conf** позволяет установить следующие опции:

- удалять ли домашний и почтовый каталоги пользователя;
- резервировать ли файлы пользователя и куда поместить резервные копии;
- удалять ли все файлы, которыми владел пользователь;
- удалять ли группу, если в данный момент она не имеет членов.

[11] В системе AIX */etc/security/{passwd,group}*.

Система SUSE поддерживает набор префиксных и постфиксных сценариев, а также сценарий **userdel.local**, который помогает команде **userdel** (и вам), вооружая соответствующие программные инструменты знаниями о ваших местных особенностях. Сконфигурируйте соответствующие параметры в файле **/etc/login.defs**.

В системе Red Hat предусмотрен сценарий `userdel.local`, но не предусмотрены префиксные и постфиксные сценарии для автоматизации операций по резервированию файлов пользователя, подлежащего удалению.

В системах Solaris и AIX есть дополнительные "тайники", в которых утаивается информация о пользователях. В основном, это файлы, обеспечивающие управление ролями и классами авторизации. Следовательно, команды `userdel` этих систем имеют более длинный "список действий" по удалению всех ссылок на удаляемого пользователя.

Например, в дополнение к файлам **/etc/passwd** и **/etc/group**, Solaris-команда **userdel** обновляет файлы **/etc/shadow**, **/etc/project** и **/etc/user_attr**. В системе AIX команда **userdel** обновляет содержимое следующих файлов в каталоге **/etc/security**: **user**, **user.roles**, **lastlog**, **environ**, **audit/config**, **limits**, **passwd** и **group**. От Solaris не следует ожидать образцового исполнения: ее команда **userdel** оставляет тестовое регистрационное имя с профилем, сконфигурированным в базе данных **user_attr**, которую необходимо "привести в порядок".

В системе HP-UX команда **userdel** лишена "интеллекта", т.е. вполне заурядно ликвидирует изменения, внесенные командой **useradd**. Она обновляет содержимое только файлов **passwd**, **shadow** и **group**.

## 7.7. Отключение учетной записи

Иногда нужно временно отключить учетную запись пользователя. Проще всего сделать это, поставив звездочку (*) или какой-то другой символ в поле зашифрованного пароля в файле **/etc/security/passwd** (AIX) или **/etc/shadow**. Эта мера препятствует большинству типов доступа, управляемых паролем, поскольку дешифровка больше не может привести к получению какого-либо приемлемого пароля. Однако такие команды, как **ssh**, которые не всегда проверяют системные пароли, могут продолжать работать.

Во всех дистрибутивах Linux, рассматриваемых нами в качестве примеров, команды **usermod -L** *пользователь* и **usermod -U** *пользователь* позволяют легко соответственно блокировать и разблокировать пароли. Флаг **-L** просто помещает символ "!" перед зашифрованным паролем в файле **/etc/shadow**, а флаг **-U** удаляет его.

Суперпользователь в системе Solaris может заблокировать учетную запись с помощью команды **passwd -l** *login_имя*, заставить пользователя изменить пароль с помощью флага **-f** или разблокировать учетную запись с помощью флага **-u**. При блокировке учетной записи в поле пароля (в файле **/etc/shadow**) добавляются символы `*LK*`. Они также представляют собой значение, устанавливаемое командой `useradd` для новых пользователей.

Система HP-UX поддерживает только зашифрованные с помощью **crypt** пароли. Символ "*" никогда не может принадлежать полю пароля, сгенериро-

ванного системой **crypt**, поэтому добавление символа "*" к зашифрованному паролю не позволит пользователю войти в систему.

В системе AIX, если поле пароля в файле **/etc/passwd** содержит символ "*" вместо символа "!", учетная запись блокируется. AIX-команда **pwdadm** может заставить пользователя изменить пароль или заблокировать учетную запись, чтобы только администратор мог изменить пароль.

К сожалению, модификация пароля пользователя просто приводит к блокированию регистрационных имен. При этом пользователь не уведомляется о блокировке пароля и не получает разъяснений, почему его учетная запись больше не работает. Альтернативный способ блокировать регистрационные имена состоит в замене интерпретатора команд пользователя программой, которая выдает сообщение, поясняющее, почему учетная запись отключена, и содержащее инструкции по исправлению ситуации. Затем программа завершается, завершая сеанс регистрации.

Этот метод имеет как преимущества, так и недостатки. Те формы доступа, которые проверяют пароли, но не обращают внимания на интерпретатор команд, не будут заблокированы. Многие демоны, предоставляющие доступ к системе без регистрации (например, ftpd), проверяют, упомянут ли интерпретатор пользователя в файле **/etc/shells**. Если он там не указан, вход в систему будет запрещен (именно это нам и требуется). К сожалению, этот метод нельзя назвать универсальным. Поэтому, если для блокирования учетных записей вы решите модифицировать интерпретатор команд, вам придется провести дополнительные исследования своей системы.

Кроме того, поясняющее сообщение нельзя увидеть, если пользователь пытается зарегистрироваться в системе в оконной среде или посредством эмулятора терминала, который не позволяет увидеть сообщения после выхода из системы.

По умолчанию программа **sendmail** не доставляет почту тем пользователям, чьи интерпретаторы не указаны в файле **/etc/shells**. Как правило, не рекомендуется вмешиваться в доставку почты, даже если получатель не может прочитать ее немедленно. Для того чтобы "обмануть" программу **sendmail**, добавьте в файл **/etc/shells** ложный интерпретатор с именем `/SENDMAIL/ANY/SHELL` (конечно, это может вызвать нежелательные побочные эффекты).

## 7.9. Управление учетными записями системными средствами

Системы HP-UX и AIX предоставляют комплексные инструментарии системного администрирования, в которые заложены "знания", как управлять пользователями, по крайней мере, на базовом уровне. В системе AIX это интерфейс SMIT (System Management Interface Tool), а в HP-UX — интерфейс, который сейчас называется **smh** (System Management Homepage). В более ранних версиях HP-UX этот инструмент носил название **sam** (System Administration Manager). Все эти инструменты используют экраны для добавления пользователей и управления ими, с применением либо оконного (GUI), либо терминального интерфейса на базе библиотеки **curses**. Если вы — начинающий системный администратор или уже опытный специалист, но осваивающий новую операционную систему, эти инструменты — как раз то, с чего вам нужно начать для решения многих рутинных задач системного администрирования.

AIX-инструмент **smitty** отличается удобной функциональностью. Нажав клавишу <F6>, вы увидите команду (с возможными параметрами), предлагаемую для выполнения. Этот инструмент регистрирует все действия и хранит файл сценария с командами, выполняемыми от вашего имени. Он может послужить хорошим подспорьем в изучении особенностей системы AIX. Если вы работаете в системе HP-UX, то, вероятно, вам понравится инструмент **smh**, который оснащен односимвольными клавиатурными командами, принимаемыми его интерфейсом **curses**. Эти команды отображаются на каждой странице меню, и поэтому вы сможете быстро с ними познакомиться.

# 7.10. Уменьшение риска с помощью PAM

Подключаемые модули аутентификации (Pluggable Authentication Modules — PAM) подробно описаны в разделе 22.5. В них сосредоточено управление системными средствами аутентификации посредством стандартных библиотечных утилит, чтобы такие программы, как **login**, **sudo**, **passwd** и **su**, не должны были предоставлять собственный код аутентификации. Модули PAM уменьшают риск, присущий отдельно взятым программным продуктам, позволяют администраторам устанавливать политики безопасности, действующие в рамках всего узла сети, а также определяют простой способ добавления в систему новых методов аутентификации.

Добавление и удаление учетных записей пользователей не предусматривает изменение конфигурации модулей PAM, но используемые для этих целей инструменты должны действовать по правилам PAM и в соответствии с требованиями PAM. Кроме того, многие параметры конфигурации PAM подобны тем, которые используются командами **useradd** или **usermod**. Если вы измените какой-нибудь параметр (как описано ниже в этой главе), а команда **useradd**, казалось бы, не обратит на это внимания, проверьте, не аннулировала системная конфигурация PAM ваше новое значение.

# 7.11. Централизация управления учетными записями

В средних и крупных корпорациях всех типов (академических, частных или государственных) важно использовать определенную форму централизованного управления учетными записями. Каждый пользователь должен иметь в узле уникальное, удобное и безопасное регистрационное имя, идентификатор (UID) и пароль. Администраторам нужна централизованная система, которая бы немедленно обеспечивала повсеместное распространение изменений (например, аннулирование конкретной учетной записи).

Такая централизация может быть достигнута различными способами, большинство из которых (включая систему Active Directory компании Microsoft) в той или иной степени (от бесплатных инсталляций с открытым исходным кодом до дорогих коммерческих систем) использует упрощенный протокол доступа к сетевым каталогам (Lightweight Directory Access Protocol — LDAP).

## Протокол LDAP и служба Active Directory

🕮 Подробнее о протоколе LDAP и его реализациях рассказывается в разделе 19.3.

Протокол LDAP представляет собой обобщенный репозиторий (наподобие базы данных), который может хранить данные, связанные с управлением пользователями, а также другие типы данных. Он использует иерархическую модель "клиент/сервер", которая поддерживает несколько серверов и несколько одновременно работающих клиентов.

Одно из самых больших преимуществ LDAP как общеузлового репозитория для регистрационных данных состоит в том, что он может обеспечить в системах уникальность идентификаторов UID и GID. Кроме того, он хорошо стыкуется с Windows, хотя обратное утверждение справедливо лишь в самой малой степени.

Служба Active Directory использует протоколы LDAP и Kerberos (сетевой протокол аутентификации) и может управлять множеством типов данных, включая данные пользователей. Она ведет себя несколько "эгоистично", навязывая "свою линию" при взаимодействии с UNIX- или Linux-репозиториями LDAP. Если вам нужна единая система аутентификации для узла, который включает Windows-компьютеры, а также UNIX- и Linux-системы, то, возможно, проще всего позволить службе Active Directory использовать ваши UNIX-базы данных LDAP в качестве вспомогательных серверов.

Для реализации такой конфигурации вам понадобится служба Active Directory и Microsoft-службы для UNIX (Services for UNIX) или такая коммерческая интеграционная платформа Active Directory, как Quest Authentication Services (ранее именуемая Vintela Authorization Services). Система Virtual Directory (Sun) может состыковать несколько различных систем авторизации/аутентификации.

Все наши примеры систем поддерживают LDAP как встроенный компонент, хотя иногда лишь на стороне клиента (например, это касается системы HP-UX). Протокол LDAP для выполнения задач аутентификации часто связывается с модулями PAM.

Протокол LDAP — это, по сути, база данных, поэтому хранимая там информация должна быть построена по определенной схеме. Под схемами в данном случае понимаются XML-файлы, имена полей в которых должны отвечать релевантным форматам RFC (Request for Comments — запрос комментариев, т.е. документ из серии пронумерованных информационных документов Интернета), в основном RFC2307 для данных управления пользователями. Подробнее написано в главе 19.

## Системы "единого входа"

Системы с однократной идентификацией пользователей (single sign-on — SSO) уравновешивают удобства пользователя с проблемами безопасности. Их основная идея состоит в том, чтобы пользователь, один раз зарегистрировавшись (в ответ на соответствующее приглашение, в веб-странице или в окне Windows), мог затем переходить, к примеру, из одного раздела в другой без повторной аутентификации. Другими словами, пользователь получает аутентификационный мандат (обычно в неявном виде, чтобы не требовалось больше никакого активного управления), который затем можно использовать для доступа к другим компьютерам и приложениям. Пользователь должен помнить только одну последовательность (а не много), состоящую из регистрационного имени и пароля.

Эта схема позволяет усложнять мандаты (поскольку пользователю не нужно помнить о них и вообще иметь с ними дело), которые теоретически повышают степень сложности. Однако влияние скомпрометированной учетной записи в этом случае усиливается, поскольку одно регистрационное имя предоставляет взломщику доступ сразу к нескольким компьютерам и приложениям. Системы SSO переносят "арену действий" из настольного компьютера (пока вы спокойно регистрировались) в зону особой уязвимости. Кроме того, узким местом системы становится сервер аутентификации. Если он "упадет", вся работа организации остановится.

Несмотря на то что в основе системы SSO лежит простая идея, она предполагает большую внутреннюю сложность, поскольку различные приложения и компьютеры,

к которым пользователь хотел получить доступ, должны понимать процесс аутентификации и SSO-мандаты. В некоторых системах SSO мандатами пользователей управляет протокол Kerberos (подробнее он описан в разделе 22.10).

Существует ряд систем SSO с открытым исходным кодом:

- JOSSO, сервер SSO с открытым исходным кодом, написанный на языке Java;
- служба CAS (Central Authentication Service), созданная Йельским университетом (на языке Java);
- продукт Likewise Open, средство интеграции, которое заставляет службу Microsoft Active Directory хорошо работать с системами Linux и UNIX.

Существуют также коммерческие системы, большинство из которых интегрировано с семействами программных продуктов, предназначенными для управления учетными данными (о них пойдет речь в следующем разделе).

## Системы управления учетными данными

Управление учетными данными (identity management) — это последний писк моды в области управления пользователями. Если говорить проще, то этот термин означает идентификацию пользователей конкретной системы, аутентифицирование их личностей и предоставление привилегий на основе этих аутентифицированных личностей. Стандартизация этой деятельности проводится такими организациями, как World Wide Web Consortium и The Open Group.

Коммерческие системы управления учетными данными сочетают несколько ключевых концепций UNIX в виде графического интерфейса, "приправленного" терминами из сферы маркетинга. Основным элементом для всех этих систем является база-данных, которая содержит информацию, связанную с аутентификацией и авторизацией пользователей, часто хранимую в формате LDAP. Управление реализуется на базе таких понятий, как группы UNIX, а ограниченные административные права усиливаются посредством таких утилит, как `sudo`. Большинство таких систем было разработано с целью урегулирования требований, диктуемых с точки зрения возможности идентификации, использования контрольных записей и слежения за ними.

Создано много коммерческих систем в этой сфере, например: Identity Management (Oracle), Sun Identity Management Suite[12], Courion, Avatier Identity Management Suite (AIMS) и BMC Identity Management Suite. Оценивая системы управления учетными данными, ищите следующие функции:

- генерирование глобальных уникальных идентификаторов пользователей;
- возможность создания, изменения и удаления пользовательских учетных записей в организации на оборудовании и операционных системах всех типов;
- безопасный веб-интерфейс для управления, доступного как изнутри, так и снаружи организации;
- возможность простого отображения имен всех пользователей, которые имеют определенный набор привилегий;
- простой способ узнать все права, предоставленные конкретному пользователю;
- возможность разрешить пользователям изменять (и восстанавливать) собственные пароли при соблюдении правил выбора сильных паролей;

[12] Теперь, когда компания Oracle купила Sun, не ясно, выживет ли эта система как продукт после завершения процесса поглощения.

- возможность для пользователей менять их пароли глобально в одной операции;
- механизм организации действий; например, многоуровневые одобрения до того, как пользователь получит определенные права;
- возможность согласования с базами данных кадров для автоматического удаления доступа для служащих, которые уволены или сокращены;
- регистрация с перестраиваемой конфигурацией всех изменений и действий администратора;
- реконфигурируемые отчеты на основе регистрационных данных (по пользователю, по дате и пр.);
- управление доступом на основе ролей, включая инициализацию учетных записей пользователей с использованием ролевого принципа;
- интерфейс, при использовании которого наем менеджеров может потребовать, чтобы учетные записи предоставлялись в соответствии с ролью;
- исключения для предоставления прав на основе ролей, включая организацию действий для рассмотрения исключений.

Рассмотрите также, как реализована система с точки зрения того, какие в действительности выполняются аутентификации и авторизации. Требуется ли системе повсеместная инсталляция пользовательских агентов или возможно самостоятельное согласование с базовыми системами?

## 7.12. Рекомендуемая литература

*"The Complete Buyer's Guide for Identity Management."// Sun Microsystems white paper*, 2008, `sun.systemnews.com/articles/129/4/sec/20930`.

## 7.13. Упражнения

7.1. Как определить стандартную группу, в которую входит пользователь? Как изменить ее?

7.2. Объясните разницу между следующими значениями **umask**: 077, 027, 022 и 755. Как сделать одно из этих значений стандартной общесистемной установкой для новых пользователей? Можно ли заставить пользователей придерживаться одного и того же значения **umask**?

7.3. Каково назначение файла скрытых паролей?

7.4. ✯ Определите, какую систему аутентификации использует программа `login` в вашей системе. Если это модуль PAM, определите, какие другие программы в системе также используют PAM.

7.5. ✯ Перечислите действия, которые необходимо выполнить для добавления пользователя в систему без помощи команды **useradd**. Какие действия дополнительно необходимы в вашей организации?

7.6. ✯ Разработайте схему выбора пользовательских имен для вашей организации. Какие правила устанавливает данная схема? Как обеспечить уникальность имен? Есть ли недостатки у этой схемы? Как происходит удаление пользователей?

7.7. ★★ Возьмите список имен (например, из телефонного справочника) зуйте его в качестве входной информации для сценария, генерирующег ционные имена в соответствии с разработанной выше схемой. Сколь записей удалось получить до возникновения конфликта? Сколько во фликтов возникло? На основании полученных результатов дайте оценку бора пользовательских имен и предложите улучшения к ней.

7.8. ★★ Напишите сценарий проверки файла **/etc/passwd**, выполняющи ленные ниже действия (для осуществления второго и последнего пун потребоваться доступ с правами суперпользователя):

а) поиск записей с идентификатором пользователя, равным нулю;

б) поиск записей без пароля (требуется файл **/etc/shadow**);

в) поиск группы записей с одинаковыми идентификаторами пользо

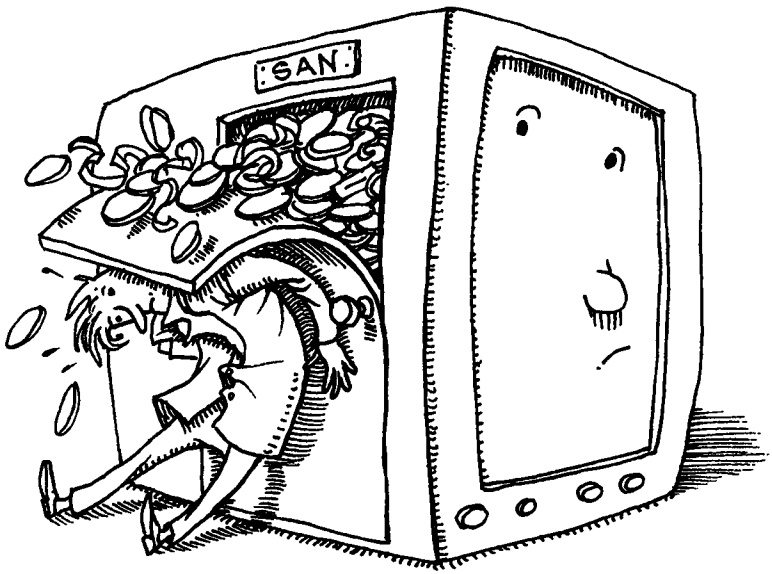
г) поиск записей с одинаковыми регистрационными именами;

д) поиск записей, для которых не установлен срок действия (требуе **etc/shadow**).

7.9. ★★★★★ Напишите модуль PAM для выполнения аутентификации п рирования случайного PIN-кода, отправляемого на мобильный телефо вателя в качестве SMS-сообщения, с последующим приглашением в PIN-код для подтверждения. Инсталлируйте свой модуль и сконфигур в стеке PAM-регистрации для получения двухфакторного процесса а кации.

# Глава 8

# Дисковая память




Дисковая память системы UNIX все больше напоминает фрагменты огромной головоломки, которые можно соединять друг с другом, создавая бесчисленное множество вариантов конфигурации. Что желаете собрать? Истребитель? Самосвал? Современнейший вертолет с воздушными подушками безопасности и приборами ночного видения?

Традиционные жесткие диски остаются доминирующим средством для создания интерактивных хранилищ данных, но при создании высокопроизводительных приложений они все чаще объединяются с флеш-дисками (solid state drive — SSD). Для всего этого аппаратного обеспечения разработаны программные компоненты, играющие роль посредников между устройствами для хранения данных и файловой иерархией, которую видит пользователь. К этим компонентам относятся драйверы устройств, соглашения о разделении дисков, реализации массивов дисков (RAID), менеджеры логических томов, системы виртуализации дисков в сети и реализации самих файловых систем.

В главе обсуждаются административные проблемы и решения, возникающие на каждом из этих уровней. Мы начнем с простых инструкций, позволяющих добавить базовый диск в каждую из рассматриваемых операционных систем. Затем опишем технологии, лежащие в основе современного аппаратного обеспечения для дисковой памяти, и рассмотрим общую архитектуру программного обеспечения для дисковой памяти. Затем мы опишем проблемы, связанные с дисковой памятью, начиная с низкоуровневого форматирования и заканчивая файловой системой. По ходу изложения рассмотрим вопросы, связанные с логическим разделением дисков, системами RAID, менеджерами логических томов и системами для реализации сетей хранения данных (storage area network — SAN).

Несмотря на то что все производители используют стандартизованное дисковое оборудование, в области программного обеспечения наблюдается огромное разнообразие. Соответственно, в главе рассматривается множество особенностей, присущих отдель-

ным производителям. Мы старались достаточно подробно описать каждую систему, чтобы читатели могли хотя бы идентифицировать команды и системы, которые они используют, и найти соответствующую документацию.

# 8.1. Добавление диска

Прежде чем погрузимся в многостраничное повествование об архитектуре и теории дисковых систем, рассмотрим самый обычный сценарий: мы хотим инсталлировать жесткий диск и сделать его доступным посредством файловой системы. В этом сценарии нет ничего особенного: нет никакого массива RAID, все пространство диска находится на одном логическом томе, а тип файловой системы выбирается по умолчанию.

На первом этапе присоединим диск и повторно загрузим систему. Некоторые системы допускают подключение дисков без перезагрузки, но мы не будем рассматривать этот случай. Кроме того, конкретные инструкции могут изменяться в зависимости от операционной системы.

Независимо от операционной системы, в которой вы работаете, очень важно *правильно идентифицировать и форматировать дисковое устройство*. Вновь добавленному диску не всегда соответствует файл устройства с самым большим порядковым номером. В некоторых системах добавление нового диска может изменить имена существующих дисков. Дважды проверьте идентичность нового диска, проверив его производителя, размер и номер модели, прежде чем начнете выполнять потенциально опасные операции.

## Инструкции для системы Linux

Выполните команду **sudo fdisk -l**, чтобы увидеть список системных дисков и идентифицировать новое устройство. Затем запустите любую утилиту для логического разбиения дисков и создания таблицы разделов для данного устройства. Для устройств 2TB и ниже инсталлируйте таблицу разделов MBR для системы Windows. Проще всего это сделать с помощью утилиты **cfdisk**, но для этой цели можно также использовать утилиты **fdisk**, **sfdisk**, **parted** или **gparted**. Более крупные диски требуют использования таблицы разделов GPT, поэтому их следует разбивать на логические разделы с помощью утилиты **parted** или ее интерфейса **gparted** в среде GNOME. Программу **gparted** использовать намного проще, но она, как правило, не инсталлируется по умолчанию.

Поместите все дисковое пространство в один раздел "неформатированного" типа. Не инсталлируйте файловую систему. Запомните логическое имя устройства в новом разбиении, прежде чем запускать утилиту разбиения диска; допустим, что она называется **/dev/sdc1**.

Затем выполните следующую последовательность команд, выбирая подходящие имена для групп тома (*vgname*), логического тома (*volname*) и точки монтирования (*mountpoint*), например homevg, home или **/home**.

```
$ sudo pvcreate  /dev/sdc1                          #Готовимся использовать w/LVM
$ sudo vgcreate vgname /dev/sdc1                    #Создаем группу тома
$ sudo lvcreate -l 100%FREE -n volname vgname #Создаем логический том
$ sudo mkfs -t ext4 /dev/vgname/volname             #Создаем файловую систему
$ sudo mkdir mountpoint                             #Создаем точку монтирования
$ sudo vi /etc/fstab                                #Задаем опции монтирования
```

Откройте файл **/etc/fstab**, скопируйте строку, соответствующую существующей файловой системе, и уточните ее. Монтируемым устройством должен быть диск **/dev** */vgname/volname*. Если ваш существующий файл **fstab** идентифицирует тома по стандарту UUID, замените пункт UUID=*xxx* файлом устройства; для томов LVM идентификация по стандарту UUID не нужна.

В заключение выполните команду **sudo mount** *mountpoint*, чтобы смонтировать файловую систему.

Более подробную информацию о файлах устройств для дисков в системе Linux можно найти в разделе 8.5. Информация о разбиении диска на логические разделы приведена в разделе 8.6, а о менеджере логических томов — в разделе 8.8. Семейство файловых систем ext4 обсуждается в разделе 8.9.

## Инструкции для системы Solaris

Выполните команду **sudo format** и проверьте меню, состоящее из имен известных дисков, чтобы выяснить имя нового устройства. Допустим, что оно называется **c9t0d0**. Нажмите комбинацию клавиш <Ctrl+C>, чтобы отменить выполнение этой команды.

Выполните команду **zpool create** *имя_пула* **c9t0d0**. Выберите простое имя пула, например home или extra. Система ZFS создаст новую файловую систему и смонтирует ее в каталоге */имя_пула*.

Более подробно дисковые устройства описаны в разделе 8.5. Обзор системы ZFS приведен в разделе 8.10.

## Инструкции для системы HP-UX

Выполните команду **sudo ioscan -fNn -Cdisk**, чтобы идентифицировать файлы устройств для нового диска. Допустим, что они называются **/dev/disk/disk4** и **/dev/rdisk/disk4**.

Затем выполните следующую последовательность команд, выбирая подходящие имена для групп тома (*vgname*), логического тома (*volname*) и точки монтирования (*mountpoint*), например homevg, home или **/home**.

```
$ sudo pvcreate  /dev/rdisk/disk4                  #Готовимся использовать w/LVM
$ sudo vgcreate vgname /dev/disk/disk4             #Создаем группу тома
$ vgdisplay vgname
...
Free PE  freespace                                 #Отметьте это значение
...
$ sudo lvcreate -l freespace -n volname vgname #Создаем логический том
$ sudo mkfs /dev/vgname/volname                    #Создаем файловую систему
$ sudo mkdir mounpoint                             #Создаем точку монтирования
$ sudo vi /etc/fstab                               #Задаем опции монтирования
```

Откройте файл **/etc/fstab**, скопируйте строку, соответствующую существующей файловой системе, и уточните ее. Монтируемым устройством должен быть диск **/dev** */vgname/volname*.

В заключение выполните команду **sudo mount** *mountpoint*, чтобы смонтировать файловую систему.

Более подробную информацию о файлах устройств для дисков в системе HP-UX можно найти в разделе 8.5. Информация о разбиении диска на логические разделы приведена в разделе 8.6. Файловая система VxFS обсуждается в разделе 8.9.

## Инструкции для системы AIX

Выполните команду **lsdev -C -c disk**, чтобы увидеть список известных дисков в системе. Затем выполните команду **lspv**, чтобы увидеть, какие диски уже настроены для управления томами. Устройство, которое есть в первом списке, но не во втором, и есть ваш новый диск. Допустим, что он называется **hdisk1**.

Затем выполните следующую последовательность команд, выбирая подходящие имена для групп тома (*vgname*), логического тома (*volname*) и точки монтирования (*mountpoint*), например **homevg**, **home** или **/home**.

```
$ sudo mkvg -y vgname hdisk1                    #Создаем группу тома
$ lsvg vgname                                   #Отмечает значение freespace
...
MAX LVs:  256    FREE PPs:  325 (freespace, Мб)
...
$ sudo crfs -v jfs2 -g vgname -m mountpoint -a size=freespaceM
$ sudo mkdir mounpoint
$ sudo mount mountpoint
```

Более подробную информацию о файлах устройств для дисков в системе AIX можно найти в разделе 8.5. Информация о разбиении диска на логические разделы в системе AIX приведена в разделе 8.6. Файловая система JFS2 обсуждается в разделе 8.9.

# 8.2. Аппаратное обеспечение дисковой памяти

Даже сегодня в мире Интернета есть лишь несколько способов хранения компьютерных данных: жесткие диски, флеш-память, магнитные ленты и оптические устройства. Последние две технологии имеют существенные ограничения, которые не позволяют рассматривать их в качестве средства для создания базовой файловой системы. Однако они по-прежнему широко используются для резервного копирования и организации хранения данных с автоматически устанавливаемыми носителями ("near-line" storage), поскольку в этих ситуациях мгновенный доступ и возможность повторной записи не являются главными факторами.

🕮 Обзор современных технологий, основанных на использовании магнитных лент, приведен в разделе 10.2.

После 40 лет развития технологии жестких дисков разработчики систем наконец-то получили реальную альтернативу — флеш-диски (SSD). Эти устройства предлагают другой набор функциональных возможностей, которые отличаются от возможностей жестких дисков и которые в ближайшие годы окажут сильное влияние на архитектуру баз данных, файловых и операционных систем.

В то же время традиционные жесткие диски продолжают резко увеличивать свою емкость. Двадцать лет назад жесткий диск емкостью 60 Мбайт стоил одну тысячу долларов. В настоящее время существует широкий выбор жестких дисков емкостью 1Тбайт, которые стоят около 80 долл. Это означает, что за те же деньги можно в 200 тысяч раз увеличить емкость дисковой памяти, а объем памяти каждые 1,15 года вдвое превышает ее стоимость. Этот показатель почти в два раза превышает скорость, предсказанную законом Мура. В этот же период последовательная пропускная способность устройств массового производства увеличилась с 500 Кбайт/с до 100 Мбайт/с, что составляет всего

лишь 200-кратное увеличение. В то же время организация прямого доступа к памяти по-прежнему стоит дорого. В общем, "с этих пор по-новому остается все по-старому".

В последние годы появилась третья, гибридная, категория жестких дисков с большими буферами флеш-памяти, которые пока не вышли на уровень массового производства. Пока неизвестно, по каким причинам это не произошло: техническим, производственным или рыночным. Они могут появиться на рынке, но их потенциальное влияние на состояние дел пока остается неясным.

Размеры дисков указываются в гигабайтах, которые составляют миллиарды байтов, в отличие от памяти, объем которой указывается в двоичных гигабайтах, которые равны $2^{30}$ байт (1 073 741 824 байт). Разница составляет около 7%. При оценке и сравнении емкости дисков и устройств памяти об этом следует помнить.

🕮 Более подробная информация о стандартных единицах измерения приведена в разделе 1.8.

Жесткие и флеш-диски довольно похожи друг на друга, поэтому они где-то взаимозаменяемы, по крайней мере на уровне аппаратного обеспечения. Они используют одинаковые интерфейсы аппаратного обеспечения и интерфейсные протоколы. Однако они имеют разные параметры, которые приведены в табл. 8.1.

**Таблица 8.1. Максимальные размеры передаваемых блоков в сетях различных типов**

| Характеристики | Жесткие диски | Флеш-диски |
|---|---|---|
| Объем | Терабайты | Гигабайты |
| Время прямого доступа | 8 мс | 0,25 мс |
| Последовательное считывание | 100 Мбайт/с | 250 Мбайт/с |
| Случайное считывание | 2 Мбайт/с | 250 Мбайт/с |
| Стоимость | 0,10 долл./Гбайт | 3 долл./Гбайт |
| Надежность | Средняя | Неизвестна |
| Ограничения записи | Нет | Да |

В следующих разделах мы рассмотрим эти технологии более подробно.

## Жесткие диски

Обычный жесткий диск состоит из нескольких вращающихся пластин, покрытых магнитным слоем. Считывание и запись данных осуществляются с помощью маленьких скользящих головок, смонтированных на металлическом рычаге, который перемещает их вперед и назад. Головки расположены близко к поверхности пластин, но не прикасаются к ним.

Считывание с пластины осуществляется быстро; оно сводится к механическому перемещению головки в требуемый сектор. В то же время существует два источника задержки.

Рычаг, на котором расположены головки, должен резко перескакивать на соответствующую дорожку. Время, которое требуется для этого, называется задержкой поиска (seek delay). Затем система должна подождать, пока требуемый сектор не окажется под головкой при вращении пластины. Время, которое требуется для этого, называется временем ожидания (rotational latency). Если последовательность операций чтения организована оптимально, то диски могут передавать данные со скоростью до десятков мегабайтов в секунду, но скорость прямого доступа в лучшем случае составляет несколько мегабайтов в секунду.

Совокупность дорожек на разных пластинах, которые находятся на одинаковом расстоянии от оси вращения, называется цилиндром. Данные, расположенные на цилиндре, можно считывать, не перемещая рычаг. Несмотря на то что головки работают чрезвычайно быстро, они перемещаются намного медленнее, чем вращается диск. Следовательно, любой доступ к диску, не требующий перемещения головки в новую позицию, будет выполняться быстрее.

Со временем скорость вращения пластин увеличилась. В настоящее время стандартная скорость вращения на дисках массового производства, ориентированных на высокую скорость доступа, составляет 7 200 оборотов в минуту (RPM — revolutions per minute), а лучшие диски достигают скорости 10 и 15 тысяч оборотов в минуту. Более высокая скорость вращения уменьшает время ожидания и увеличивает пропускную способность при передаче данных, но при этом диски нагреваются.

Жесткие диски часто выходят из строя. В 2007 году подразделение Google Labs, изучив 100 тыс. дисков, удивило техническое сообщество сообщением, что средний ежегодный процент отказов (annual failure rate — AFR) жестких дисков, проработавших более двух лет, превышает 6%, что намного больше показателя, предсказанного производителями на основе экстраполяции результатов краткосрочных тестирований. Поведение дисков можно описать так: несколько месяцев риска "детской смертности", два года безупречной работы, а затем ежегодный процент отказов взлетает до 6–8%. В целом, вероятность пятилетнего "выживания" жестких дисков в исследовании компании Google не превышает 75%.

Интересно, что компания Google не нашла корреляции между процентом отказов и двумя факторами внешней среды, которые ранее считались важными, — температурный режим работы и активность диска. Полный отчет можно найти по адресу: `tinyurl.com/fail-pdf`.

Отказы дисков, как правило, связаны с повреждением либо поверхности пластин (сбойные блоки), либо механических компонентов. Программно-аппаратные средства и интерфейсы аппаратных устройств после сбоя обычно остаются работоспособными, поэтому существует возможность выяснить детали происшедшего сбоя (см. раздел 8.5).

Надежность диска производители обычно измеряют с помощью среднего времени между двумя сбоями (mean time between failures — MTBF), которое измеряется часами. Обычно значение показателя MTBF у промышленного диска составляет 1,2 млн часов. Однако этот показатель носит статистический характер и означает, что конкретный диск проработает 140 лет до первого сбоя.

Показатель MTBF является обратным по отношению к показателю AFR, вычисленному для установившегося режима работы диска, т.е. после включения в систему, но до износа. Показатель MTBF, указываемый производителями и равный 1,2 млн часов, соответствует показателю AFR, равному 0,7% в год. Это значение почти (но не совсем) совпадает с показателем, выявленным компанией Google (1-2%) на протяжении первых двух лет работы их тестовых дисков.

Возможно, показатель MTBF, указываемый производителями, точен, но он получен на основе преднамеренного подбора данных о самом благополучном периоде работы дисков. Следовательно, этот показатель можно считать лишь верхней границей надежности, но его нельзя использовать для прогнозирования фактического процента отказов в долгосрочной перспективе. Основываясь на приведенных выше данных, следует разделить показатель MTBF, объявленный производителями, на 7,5 или перейти к более реалистичным методам выявления частоты отказов за пять лет.

Жесткие диски представляют собой товар. Одна модель может оказаться лучше другой, имея одинаковые скорость вращения, интерфейс аппаратного обеспечения и уровень надежности. В настоящее время существуют специализированные лаборатории, которые проводят сравнительный анализ конкурирующих дисков.

## Флеш-диски

Флеш-диски (SSD) распределяют операции считывания и записи по группам ячеек памяти, каждая из которых по отдельности медленнее, чем современные жесткие диски. Однако, благодаря параллельной работе, диски SSD в целом соответствуют ширине полосы пропускания традиционных дисков и даже превосходят ее. Огромное преимущество флеш-дисков в том, что они продолжают хорошо работать даже тогда, когда происходит прямой доступ к данным для чтения или записи. Эта особенность является очень привлекательной в реальных условиях.

Производители дисковой памяти любят приводить скорость последовательной передачи данных для своей продукции, поскольку эти числа впечатляюще высокие. Однако для традиционных жестких дисков этот показатель практически не связан с пропускной способностью, наблюдаемой при чтении и записи с прямым доступом. Например, скорость последовательной передачи данных высокопроизводительных дисковых устройств компании Western Digital может достигать 120 Мбайт/с, но их скорость прямого доступа составляет примерно 2 Мбайт/с. В противоположность этому, современные диски SSD производства компании Intel обеспечивают скорость передачи данных, равную 30 Мбайт/с в обоих режимах работы.

Тем не менее эта производительность достигается дорогой ценой. Накопители SSD не только дороже в перерасчете на гигабайт, чем жесткие диски, но и порождают новые сложности и неопределенности.

Каждая страница флеш-памяти на диске SSD (в современных дисках ее размер равен четырем двоичным килобайтам) может быть перезаписана ограниченное количество раз (обычно 100 тысяч, в зависимости от использованной технологии). Для того чтобы ограничить износ страницы, программное обеспечение дисков SSD ведет таблицы отображений и распределяет записи по всем страницам накопителя. Эти отображения остаются скрытыми от операционной системы, которая рассматривает накопитель как последовательный ряд блоков. Такой накопитель можно считать виртуальной памятью.

Еще одна сложность заключается в том, что страницы флеш-памяти должны быть очищены, прежде чем использовать их для перезаписи. Очистка представляет собой отдельную операцию, которая выполняется медленнее, чем сама запись. Кроме того, невозможно очистить отдельные страницы — кластеры или смежные страницы (как правило, они составляют 128 или 512 Кибибайт). Если пул заранее очищенных записей исчерпан, то производительность записи на SSD-накопитель может значительно снизиться и устройство должно восстановить страницы на лету, чтобы продолжить запись.

Перестроить буфер очищенных страниц труднее, чем кажется, потому что файловые системы обычно не отмечают и не очищают блоки данных, которые больше не используются. Накопитель не знает, что файловая система считает данный блок свободным; он знает только, что кто-то когда-то записал в него данные для хранения. Для того чтобы SSD-накопитель мог поддерживать работу кеша очищенных страниц (а значит, обеспечивать высокую производительность записи), файловая система должна иметь возможность сообщать SSD-накопителю, что определенные страницы больше не нужны. Этой возможностью обладают только файловые системы ext64 и NTFS в операционной системе

Windows 7. Однако, учитывая огромный интерес к SSD-накопителям, можно не сомневаться, что и другие файловые системы в ближайшем будущем внедрят этот механизм.

Дополнительной сложностью является выравнивание. Размер стандартного блока на диске равен 512 байт, но этот размер слишком мал для того, чтобы файловая система могла эффективно с ним работать.[1] Файловые системы управляют кластерами, размеры которых колеблются от одного до восьми двоичных килобайтов, а слой трансляции отображает кластеры файловой системы в дисковые блоки для чтения и записи.

На жестком диске нет различия между началом и концом кластеров. Однако, поскольку SSD-накопители могут читать и записывать данные только на страницах размером четыре двоичных килобайта (несмотря на эмуляцию традиционных 512-байтовых блоков), границы кластеров и страниц на SSD-накопителе должны совпадать. Нежелательно, чтобы логический кластер размером четыре двоичных килобайта начинался в средине одного физического кластера на SSD-накопителе и заканчивался в средине следующего. В этом случае SSD-накопитель может дважды считать и записать данные стольких физических страниц, сколько помещается в заданном количестве логических кластеров.

Поскольку файловые системы отсчитывают свои кластеры начиная с того места, куда их записал накопитель, проблему выравнивания можно было бы решить путем выравнивания разделов дисков по границам, начинающимся с байта, номер которого представляет степень двойки. Эти разделы больше, чем размер SSD-накопителя или страницы файловой системы (например, 64 двоичных килобайта). К сожалению, схема разбиения диска Windows MBR, которую система Linux унаследовала, не выполняет такое выравнивание автоматически. Проверьте диапазоны блоков, которые вы получили при разбиении диска, и убедитесь, что они выровнены, помня, что сама схема MBR занимает один блок. (Система Windows 7 выравнивает разделы для SSD-накопителей по умолчанию.)

Теоретические лимиты перезаписи флеш-памяти, вероятно, меньше, чем казалось вначале. Исходя из простых арифметических соображений, можно понять, что при потоке данных на скорости 100 Мбайт/с и объеме SSD-накопителя, равном 150 Гбайт, лимит перезаписей будет исчерпан за четыре года непрерывной работы. Однако более общие вопросы долгосрочной надежности SSD-накопителей еще не изучены. Флеш-диски представляют собой еще незрелый продукт и могут преподносить сюрпризы.

Контроллеры, используемые внутри SSD-накопителей, быстро развиваются и в настоящее время сильно различаются по производительности. В результате рынок должен прийти к какой-то одной стандартной архитектуре этих устройств, но на это уйдет еще один или два года. В более короткой перспективе при покупке таких устройств следует проявлять осторожность.

Статья Ананда Шимпи (Anand Shimpi) о технологии SSD, появившаяся в марте 2009 года, представляет собой превосходное описание преимуществ и недостатков этой технологии. Ее можно найти по адресу `tinyurl.com/dexnbt`.

## 8.3. Интерфейсы устройств хранения данных

В настоящее время существует лишь несколько наиболее распространенных стандартов интерфейса. Если система поддерживает несколько разных интерфейсов, следует использовать тот, который наилучшим образом удовлетворяет требования, предъявляемые к скорости, надежности, мобильности и стоимости.

[1] 512-байтовый стандарт для жестких дисков постепенно выходит из употребления; см. `lwn.net/Articles/377895`.

- Интерфейс ATA (Advanced Technology Attachment), ранее известный под названием IDE, был разработан как простой и недорогой интерфейс для персональных компьютеров. Вначале он был назван Integrated Drive Electronics, потому что контроллер устройства находился в той же коробке, что и пластины диска, а для обеспечения взаимодействия между компьютером и дисками применялся протокол относительно высокого уровня. В настоящее время так работают все жесткие диски, но в то время это было новшеством.

  Традиционный параллельный интерфейс ATA (PATA ) связывал диски с материнской платой с помощью ленточного кабеля, имеющего 40 или 80 жил. Этот тип дисков устарел, но все еще установлен на огромном количестве компьютеров. Диски с интерфейсом PATA часто маркируются как IDE, чтобы отличить их от устройств SATA, которые будут описаны ниже, но на самом деле они являются устройствами ATA. Диски PATA имеют умеренную скорость, большую память и невероятно низкую стоимость.
- Последовательный интерфейс ATA (SATA) является наследником интерфейса PATA. Кроме поддержки более высокой скорости передачи данных (в настоящее время она составляет 6 Гбайт), интерфейс SATA упрощает соединение с помощью более аккуратных и длинных кабелей. Интерфейс SATA имеет собственную поддержку замены дисков без перезагрузки системы (hot-swapping) и (необязательно) поддерживает создание очереди команд. Эти две функциональные возможности превратили интерфейс ATA в перспективную альтернативу устройствам SCSI на серверах.
- Несмотря на то что интерфейс SCSI в настоящее время уже не так популярен, как раньше, он остается одним из наиболее широко распространенных дисковых интерфейсов. Он имеет несколько разновидностей, которые все, без исключения, поддерживают работу нескольких дисков на одной шине, а также разные уровни скорости и способы связи. Более подробно устройства SCSI описаны ниже.

  Производители накопителей памяти обычно используют интерфейсы SCSI в самых быстрых и надежных устройствах. Эти устройства стоят дороже, но это объясняется особенностями накопителей, а не интерфейса.
- Fibre Channel — это последовательный интерфейс, популярный в промышленности благодаря широкой полосе пропускания и большому количеству накопителей, которые он может соединять одновременно. Устройства Fibre Channel соединяются с помощью волоконно-оптических или коаксиальных медных кабелей. Скорость их работы колеблется от 1 до 40 Гбайт/с, в зависимости от версии протокола.

  Как правило, топология этих устройств представляет собой кольцо, которое называется Fibre Channel Arbitrated Loops (FC-AL), или фабрику (коммутируемые сети — *Примеч. ред.*) с коммутаторами Fibre Channel. Интерфейс Fibre Channel поддерживает несколько разных протоколов, включая SCSI и даже IP. Устройства идентифицируются по "зашитым" восьмибайтовым идентификаторам ("World Wide Name"), напоминающим адреса Ethernet MAC.
- Последовательные интерфейсы Universal Serial Bus (USB) и FireWire (IEEE1394) стали популярными средствами соединения внешних жестких дисков. В настоящее время скорость их работы составляет 480 Мбайт/с для USB и 800 Мбайт/с для Firewire. Обе системы являются слишком медленными для того, чтобы принимать быстрые потоки данных с диска на полной скорости. Прогнозируемые изменения

обоих стандартов предусматривают более высокую скорость передачи данных (до 5 Гбайт/с для USB 3.0).

В жестких дисках никогда не используются интерфейсы USB или FireWire. Вместо этого в дисковые модули, подсоединенные к этим портам, встраиваются конверторы SATA.

Интерфейсы ATA и SCSI до сих пор доминируют среди накопителей, поэтому только их мы обсудим подробно.

## Интерфейс PATA

Интерфейс PATA (Parallel Advanced Technology Attachment), называемый также IDE, задумывался как простой и недорогой. Его чаще всего можно найти в дешевых персональных компьютерах. Оригинальный интерфейс IDE стал популярен в конце 1980-х годов. Последующие версии протокола, кульминацией которых стала версия ATA-7 (известная также как Ultra ATA/133), включали в себя дополнительно режимы прямого доступа к памяти (DMA), свойства plug-and-play, адресацию логических блоков (LBA), управление электропитанием, возможности самоконтроля и скорость работы шины до 133 Мбайт/с. С момента появления интерфейса ATA-4 стандарт ATA объединился с протоколом ATA Packet Interface (ATAPI), допускавшим присоединение к шине IDE дисководов CD-ROM и лентопротяжных устройств.

Разъем PATA имеет 40 контактов, соединяющих накопитель с интерфейсной платой посредством грубого ленточного кабеля. Стандарты ATA, появившиеся после Ultra DMA/66, использовали кабель с 80 жилами, бо́льшим количеством заземляющих контактов и вследствие этого с более низким уровнем электрических помех. Более изящные кабели, допускающие скручивание ленты в толстую кабельную муфту, позволяют сделать шасси компактнее и улучшают воздухообмен. Кабель питания для накопителей PATA использует массивный четырехконтактный штепсель Molex.

Если кабель и/или накопитель не имеют ключа, то следует убедиться, что контакт 1 на накопителе входит в контакт 1 в гнезде интерфейса. Контакт 1 обычно обозначается маленькой цифрой "1" на одной стороне разъема. Если он не помечен, то его можно найти по следующему правилу: первый контакт ближе всего к разъему питания. Первый контакт на ленточном кабеле обычно маркируется красным цветом. Если вы не видите красной полосы на краю ленточного кабеля, то сделайте так, чтобы первый контакт кабеля был соединен с первым контактом разъема, и пометьте кабель красным маркером.

Большинство персональных компьютеров имеет две шины PATA, к каждой из которых присоединены два накопителя. Если к шине PATA присоединено больше одного накопителя, вы должны назначить один из них главным, а другой — подчиненным. Перемычка "выбора кабеля" на современных накопителях (обычно предусматриваемая по умолчанию) позволяет накопителям самим выбирать, какой из них будет главным, а какой подчиненным. Иногда этот механизм работает неправильно. Тогда необходимо явно назначить главный и подчиненный накопители.

Главный накопитель не имеет никаких преимуществ по производительности. Некоторые из старых накопителей PATA не могут быть подчиненными, поэтому возникают проблемы, которые решаются перестановкой ролей. Если же и это не помогает, то каждый накопитель PATA следует поместить на отдельную шину.

Разрешение конфликтов между главным и подчиненным накопителями на шине PATA может оказаться относительно медленным, поэтому по возможности следует размещать накопители PATA на отдельных шинах.

## Интерфейсы SATA

По мере возрастания скорости передачи данных для накопителей PATA, недостатки этого стандарта становились все более очевидными. Электромагнитная интерференция и другие электрические проблемы снижали надежность работы накопителей на больших скоростях. Для решения этой проблемы были изобретены последовательные интерфейсы ATA и SATA. В настоящее время они являются доминирующими среди устройств для хранения данных.

Интерфейс SATA сгладил многие недостатки интерфейса PATA. Он повысил скорость передачи данных (потенциально до 750 Мбайт/с и 6 Гбайт/с в перспективе) и реализовал отличный механизм для проверки ошибок. Этот стандарт поддерживает замену накопителя без перезагрузки системы, создание очереди команд и многие другие средства для повышения производительности. Интерфейс SATA отменил необходимость назначать главный и подчиненный накопители, поскольку к каждому каналу подключен только один накопитель.

Интерфейс SATA преодолел ограничение, связанное с использованием 18-дюмовых кабелей в интерфейсе PATA, и ввел новые стандарты для кабелей питания и информационных кабелей, состоящих из 7 и 15 жил соответственно.[2] Эти кабели намного гибче и проще в работе, чем их ленточные предшественники, — больше нет необходимости изгибать и скручивать кабели, чтобы подсоединить несколько накопителей на один кабель. Тем не менее похоже, что новый стандарт более чувствителен к качеству материала, чем старые ленточные кабели интерфейса PATA. Мы видели несколько дешевых кабелей SATA, которые при попытке их использования приводили к отказу материнской платы.[3]

Кабели SATA легко входят в разъемы, но так же легко из них выпадают. Существуют кабели с фиксаторами, но они имеют как преимущества, так и недостатки. На материнских платах, в которых разъемы SATA упакованы по шесть или восемь штук, может оказаться трудно высвободить фиксирующие разъемы без помощи тонких клещей.

Интерфейс SATA ввел в действие новый стандарт кабелей под названием eSATA. С точки зрения электрических свойств, эти кабели аналогичны кабелям SATA, но их разъемы немного отличаются. Вы можете добавить порт eSATA в систему, имеющую только внутренние разъемы SATA, инсталлировав недорогой конвертер.

С внешними модулями, содержащими несколько накопителей, имеющих только один порт eSATA, следует быть осторожными — некоторые из них являются интеллектуальными устройствами (RAID) и требуют оригинальные драйверы. (Эти драйверы редко поддерживают системы UNIX или Linux.) Другие модули не имеют сложной логики и содержат встроенный переходник для порта SATA. В принципе они допускают использование системы UNIX, но поскольку не все адаптеры ведущего контроллера SATA поддерживают расширение порта, следует уделить особое внимание вопросам совместимости. Модули с несколькими портами eSATA — по одному на накопитель — всегда являются безопасными.

## Параллельный интерфейс SCSI

Интерфейс SCSI (Small Computer System Interface) определяет обобщенный конвейер данных, который можно использовать для любых периферических устройств. В прошлом он использовался для подсоединения дисков, лентопротяжных устройств,

[2] Это правильно: по некоторым причинам кабель питания сложнее, чем информационный.

[3] В США превосходным поставщиком качественных и дешевых кабелей SATA является сайт monoprice.com.

сканеров и принтеров, но сейчас в большинстве случаев производители этих устройств отказались от интерфейса SCSI в пользу интерфейса USB.

После 1986 года, когда был принят стандарт ANSI под названием SCSI-1, появилось много разновидностей интерфейса SCSI. Традиционный интерфейс SCSI использует от 8 до 16 жил.

К сожалению, в названиях разновидностей параллельного интерфейса SCSI нет никакого смысла. Термины "fast", "wide" и "ultra" вводились для того, чтобы подчеркнуть значительные преимущества интерфейса SCSI, но сейчас эти функциональные возможности стали стандартными, и эти эпитеты исчезли из названий. Несмотря на свое название, интерфейс Ultra SCSI на самом деле обеспечивает скорость передачи данных 20 Мбайт/с, что в настоящее время уже не является мечтой, но за ним последовали Ultra2, Ultra3, Ultra-320 и Ultra-640 SCSI. Для тех, кому интересно, приведем регулярное выражение, соответствующее всевозможным разновидностям параллельного интерфейса SCSI.

(Fast(-Wide)?|Ultra(( Wide)?|2 (Wide )?|3|-320|-640)?) SCSI|SCSI-[1-3]

Использовалось также множество разных разъемов. Они зависели от версии интерфейса SCSI, типа соединения (внутреннее или внешнее) и количества бит пересылаемых данных. На рис. 8.1 показаны некоторые из них. Каждый разъем показан спереди, как будто вы смотрите на них прямо.

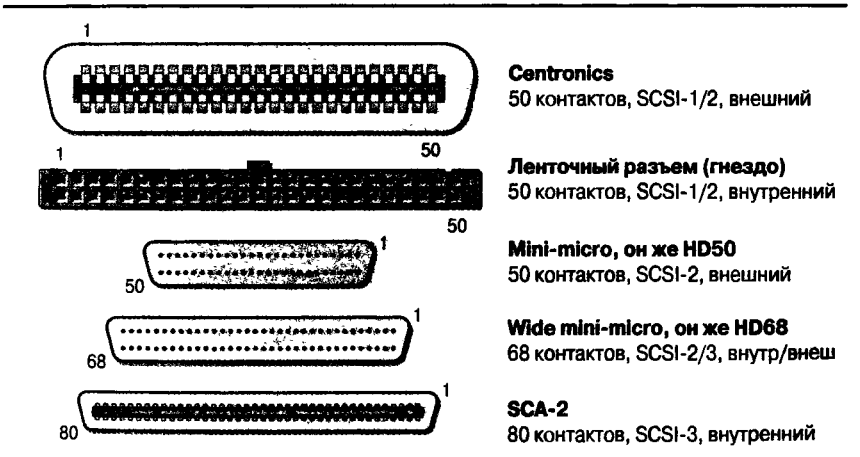



*Рис. 8.1. Параллельные разъемы SCSI (вид спереди, все со штекерами, за исключением указанного отдельно)*

В настоящее время продолжен выпуск только разъема SCA-2, имеющего 80 контактов, включая контакты питания и шины.

Каждый конец шины SCSI должен иметь согласующий резистор (terminating resistor), называемый терминатором. Этот резистор поглощает сигналы, достигающие конца шины, и предотвращает отражение шума обратно в шину. Терминаторы бывают разными — от небольших внешних разъемов, которые вставляются в обычный порт, до аккуратных пакетов резисторов, встроенных в монтажные платы устройства. Большинство современных устройств предусматривает автоматическое отключение согласующего резистора при подключении к разъему кабеля.

Если на вашей шине SCSI возникают проблемы, которые кажутся случайными, сначала проверьте согласующие резисторы на обоих концах шины. Неправильная работа согласующих резисторов является одной из основных проблем в старых системах SCSI, а ошибки, которые при этом возникают, могут быть нераспознанными и перемежающимися.

Параллельные шины SCSI используют гирляндную цепь, так что большинство внешних устройств имеет два порта SCSI.[4] Эти порты идентичны и взаимозаменяемы — любой из них может быть входом. Внутренние устройства SCSI (включая устройства с разъемами SCA-2) присоединяются к ленточному кабелю, поэтому для устройства нужен только один порт.

Каждое устройство имеет SCSI-адрес, или "целевой номер", отличающий его от других устройств, присоединенных к шине. Отсчет целевых номеров начинается с нуля и заканчивается семью или пятнадцатью, в зависимости от вида шины — узкой или широкой. Сам контроллер SCSI также рассматривается как устройство и обычно имеет номер 7. Целевые номера других устройств должны быть уникальными. Типичная ошибка возникает, когда пользователи забывают, что контроллер SCSI имеет свой целевой номер, и присваивают устройству тот же целевой номер. Если вам повезет, то устройство будет иметь внешний манипулятор, с помощью которого можно задать целевой номер. Еще одним распространенным способом установки целевых номеров являются переключатели DIP и перемычки. Если неясно, как установить целевой номер на устройстве, поищите справочное руководство в Интернете.

Интерфейс SCSI поддерживает разновидность частичной адресации, которая использует "номера логических модулей". Каждое целевое устройство может иметь внутри себя несколько логических модулей. Например, можно представить себе массив накопителей с несколькими дисками, но с одним контроллером SCSI. Если устройство SCSI имеет только один логический модуль, то он обычно по умолчанию устанавливается равным нулю.

Использование логических номеров обычно ограничивается большими массивами накопителей. Если вы услышите выражение "номер модуля SCSI", то должны предположить, что на самом деле речь идет о целевом номере, пока не доказано обратное.

Системному администратору, работающему с аппаратным обеспечением SCSI, следует помнить о следующем.

- Не беспокойтесь о точных версиях SCSI, которые поддерживает устройство; смотрите на разъемы. Если два устройства SCSI имеют одинаковые разъемы, значит, они совместимы. Однако это еще не значит, что они могут обеспечить одинаковую скорость работы. Передача данных будет происходить со скоростью, на которой работает более медленное устройство.
- Даже если разъемы отличаются друг от друга, устройства все равно могут быть совмещены с помощью адаптера, если их разъемы имеют одинаковое количество контактов.
- Многие старые рабочие станции имеют внутренние устройства SCSI, такие как лентопротяжные механизмы и дисководы гибких дисков. Перед тем как перезагрузить систему, после добавления нового устройства, проверьте список текущих устройств.

[4] "Гирляндная цепь" — это общепринятое название, но оно может ввести в заблуждение. Параллельный интерфейс SCSI физически представляет собой цепь, а с электротехнической точки зрения — это отдельная шина.

- Добавив новое устройство SCSI, проверьте список устройств, выдаваемый ядром при перезагрузке, чтобы убедиться, что все в порядке. Большинство драйверов SCSI не распознает ситуации, в которых несколько устройств имеют одинаковые SCSI-адреса (это недопустимо). Конфликт SCSI-адресов порождает неправильное поведение системы.
- Если вы наблюдаете ненадежную работу системы, проверьте, не возник ли конфликт целевых номеров, а также проверьте согласующие резисторы на шине.
- Помните, что ваш контроллер SCSI использует один из SCSI-адресов.

## Последовательный интерфейс SCSI

Аналогично устройствам PATA, параллельному интерфейсу SCSI соответствует последовательный интерфейс SCSI (Serial Attached SCSI — SAS), являющийся аналогом интерфейса SATA. С точки зрения аппаратного обеспечения интерфейс SAS улучшает практически все аспекты традиционного параллельного интерфейса SCSI.

- Сцепленные шины устарели. Подобно интерфейсу SATA, интерфейс SAS представляет собой позиционную систему (point-to-point system). Интерфейс SAS позволяет использовать расширители (expanders) для присоединения нескольких устройств к одному порту компьютера. Они напоминают мультипликаторы порта SATA, но в то время как поддержка мультипликаторов иногда реализуется, а иногда нет, расширители поддерживаются всегда.
- Интерфейс SAS не использует терминаторы.
- Целевые номера устройств SCSI больше не используются. Вместо этого каждое устройство SAS имеет назначенное производителем 64-битовое имя World Wide Name (WWN), характерное для оптико-волоконных полей. Они аналогичны адресам Ethernet MAC.
- Количество устройств на шине SCSI (фактически, "домен SAS") больше не ограничено восемью или шестнадцатью. К шине можно присоединять до 16 384 устройств.

Интерфейс SAS в настоящее время функционирует на скорости 3 Гбайт/с, но скорость вскоре планируется довести до 6 Гбайт/с, а в 2012 году — до 12 Гбайт/с.

## Что лучше: SCSI или SATA?

В прошлом интерфейс SCSI был очевидным выбором для серверных приложений. Он обеспечивал максимально широкую полосу пропускания, выборочное выполнение команд (что эквивалентно созданию очередей помеченных команд), меньшую загрузку центрального процессора, более простое обслуживание большого количества накопителей и доступ к самым современным дискам.

После появления интерфейса SATA интерфейс SCSI больше не обеспечивает отдачу от вложенных средств. Устройства SATA сравнимы с эквивалентными дисками SCSI практически в каждой категории (а иногда даже превосходят их). В то же время и устройства SATA, и интерфейсы, и кабели, используемые для их соединения, дешевле и более распространены.

Впрочем, интерфейс SCSI по-прежнему имеет несколько "козырей".

- Производители продолжают использовать разделение SATA/SCSI для стратификации рынка накопителей. Для того чтобы обосновать высокие цены, самые быстро-

действующие и надежные накопители по-прежнему поставляются только с интерфейсами SCSI.

- Интерфейс SATA ограничен глубиной очереди, вмещающей 32 незавершенные операции. Интерфейс SCSI может обслуживать тысячи операций.
- Интерфейс SAS может обслуживать много накопителей (сотни и тысячи) с помощью одного компьютера. Однако следует иметь в виду, что все эти устройства используют один и тот же канал связи с этим компьютером; вы по-прежнему ограничены общей шириной полосы пропускания, составляющей 3 Гбайт/с.

Споры об интерфейсах SAS и SATA могут в конце концов оказаться бессмысленными, поскольку стандарт SAS включает поддержку устройств SATA. Разъемы SAS и SATA достаточно похожи друг на друга, чтобы одна кабельная укладка могла работать с любым типом. На логическом уровне команды интерфейса SATA просто туннелируются через шину SASA.

Это сходство является удивительным техническим достижением, но экономическое обоснование для него менее очевидно. Расходы на инсталляцию интерфейса SAS, в основном, относятся к адаптеру компьютера, кабельной укладке и инфраструктуре; накопители SAS сами по себе не слишком дорогие. Однажды вложив средства в установку интерфейса SAS, вы будете вынуждены придерживаться его до конца. (С другой стороны, возможно, умеренная стоимость накопителей SAS является *результатом* того, что их легко заменить накопителями SATA.)

## 8.4. Программное обеспечение накопителей

Если вам когда-нибудь приходилось подсоединять диск и система Windows спрашивала вас, не желаете ли вы его отформатировать, вы могли поразиться тому, насколько сложным является управление накопителями в системах UNIX и Linux. Почему же оно такое сложное?

Для начала отметим, что на некоторых системах вы можете зарегистрироваться на системном компьютере, присоединить к нему накопитель USB и работать практически так же, как в системе Windows. Вы получите возможность выполнить простую процедуру установки накопителя для хранения персональных данных. Если это все, что вам нужно, то все в порядке.

Как обычно, в этой книге нас интересуют промышленные системы хранения данных: файловые системы, доступ к которым имеет множество пользователей (локальных и удаленных) и которые в то же время являются надежными, высокопроизводительными, допускающими простое резервирование и обновление. Такие системы требуют немного больше осторожности, и системы UNIX и Linux дают для этого достаточно оснований.

Типичный набор компонентов программного обеспечения, обеспечивающих взаимодействие между накопителем и его пользователями, приведен на рис. 8.2. Архитектура, показанная на рис. 8.2, относится к системе Linux, но и другие операционные системы обладают аналогичными возможностями, хотя и не обязательно в точно таком же виде.

Стрелки на рис. 8.2 означают "могут быть построены из". Например, файловая система Linux может быть создана на основе раздела, массива RAID или логического тома. Создание стека модулей, соединяющих каждый накопитель с его окончательным приложением, является одной из задач администратора.

Внимательный читатель заметит, что этот граф имеет цикл, а в реальных конфигурациях циклов не бывает. Система Linux допускает создание стеков из массивов RAID

и логических томов в любом порядке, но ни один из компонентов не может использоваться больше одного раза (хотя с технической точки зрения это возможно).

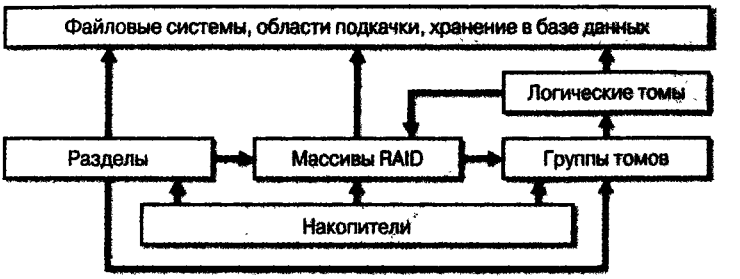



*Рис. 8.2. Уровни управления накопителем*

Рассмотрим фрагменты рис. 8.2.

- **Накопитель** (storage device) — это все, что напоминает диск. Это может быть жесткий диск, флеш-накопитель, диск SSD, внешний массив RAID, реализованный в виде аппаратного обеспечения, и даже сетевая служба, обеспечивающая доступ к удаленному устройству на уровне блока. Конкретный вид оборудования значения не имеет, поскольку устройство допускает прямой доступ, обрабатывает блочный ввод-вывод и представлено файлом устройства.
- **Раздел** (partition) — это фрагмент накопителя, имеющий фиксированный размер. Каждый раздел имеет собственный файл устройства и действует как независимый накопитель. Для обеспечения эффективности разделы создаются с помощью того же драйвера, который используется для управления устройством. Большинство схем разделения занимает несколько блоков в начале накопителя для записи диапазонов блоков, которые занимает каждый раздел.

  Разделение диска постепенно становится исчезающей функциональной возможностью. Системы Linux и Solaris сохранили ее для обеспечения совместимости с дисками, разделенными на разделы с помощью системы Windows. Системы HP-UX и AIX практически отказались от разделения дисков в пользу управления логическими томами, но в системах HP-UX, основанных на микропроцессоре Itanium, разделение дисков остается обязательным.
- **Массив RAID** (избыточный массив недорогих и независимых дисков) объединяет многочисленные накопители в одно виртуальное устройство. В зависимости от настройки этого массива, эта конфигурация может повысить производительность (путем чтения и записи дисков в параллельном режиме) или надежность (путем дублирования и проверки данных с контролем четности на всех дисках) или оба этих показателя.

  Как следует из названия, массив RAID обычно интерпретируется как совокупность обычных дисков, но современные реализации позволяют использовать в качестве компонента этого массива все, что функционирует как диск.
- **Группы томов и логические тома** ассоциируются с менеджерами логических томов (LVM — logical volume manager). Эти системы объединяют физические устройства и формируют пул накопителей, которые называются группами томов.

Администратор может разделить этот пул на логические тома почти так же, как диски разбиваются на разделы. Например, диск объемом 750 Гбайт и диск объемом 250 Гбайт можно объединить в группу томов объемом 1 Тбайт, а затем разбить на два логических тома по 500 Гбайт. По крайней мере, один том будет содержать блоки данных, относящиеся к обоим жестким дискам.

Поскольку менеджер логических дисков добавляет уровень косвенной адресации между логическими и физическими блоками, он может зафиксировать логическое состояние тома, просто сделав копии таблицы отображений. Следовательно, менеджеры логических томов часто обеспечивают возможность "мгновенных копий". Затем том связывается с новыми блоками, и менеджер LVM сохраняет старую и новую таблицы отображений. Разумеется, менеджер LVM должен хранить как исходный образ, так и все модифицированные блоки, поэтому в конце концов он исчерпает память, если мгновенные копии никогда не удаляются.

- Файловая система служит посредником между набором блоков, представленных разделом, массивом RAID, логическим блоком и стандартным интерфейсом файловой системы, ожидаемым программами: путями вроде `/var/spool/mail`, типами файлов системы UNIX, разрешениями системы UNIX и т.п. Файловая система решает, где и как хранится содержимое файлов, как представить пространство имен и организовать поиск на диске, а также как избежать сбоя (и восстанавливаться после него).

  Большая часть накопителя является частью файловой системы, но область подкачки и хранение в базе данных могут быть потенциально чуть более эффективными без "помощи" от файловой системы. Ядро или база данных навязывают накопителю свою собственную структуру, делая файловую систему излишней.

Если вы считаете, что эта система содержит слишком много маленьких компонентов, которые просто реализуют один блочный накопитель через другой, то совершенно правы. В последнее время наблюдается тенденция в сторону консолидации этих компонентов, чтобы повысить эффективность и устранить дублирование. Хотя менеджеры логических томов изначально не функционировали как контроллеры RAID, большинство из них воплотили некоторые функциональные возможности массивов RAID (в частности, распределение данных и мониторинг). Как только администраторы получили удобные средства управления логическими томами, разделы также исчезли.

В настоящее время на передовом крае находятся системы, объединяющие файловую систему, контроллер RAID и систему LVM в один интегрированный пакет. Ярким примером является файловая система ZFS, но и файловая система Btrfs для Linux предназначена для решения аналогичных задач. Система ZFS описывается в разделе 8.10.

Большинство установок относительно просты. На рис. 8.3 показана традиционная схема разбиения диска, которую можно найти на многих дисках в системе Linux. (Загрузочный диск не показан.) Если подставить вместо разделов логические тома, то установка станет похожей на инсталляцию других систем.

В следующих разделах мы детальнее рассмотрим этапы, из которых состоят фазы конфигурирования носителей, — контроль устройств, разбиение, RAID, управление логическим томом и инсталляция файловой системы. В заключение мы опишем систему ZFS и сети хранения данных.

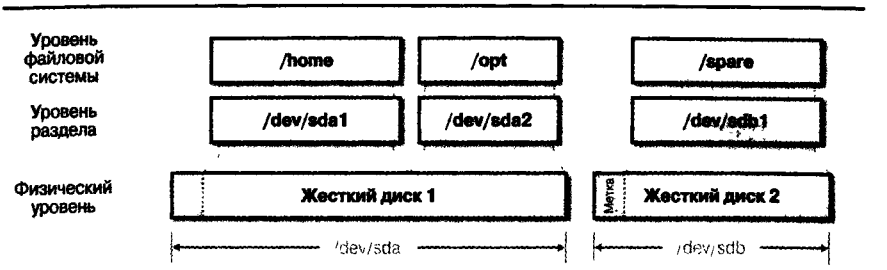



*Рис. 8.3. Традиционная схема разбиения диска (имена устройств в системе Linux)*

# 8.5. Присоединение и низкоуровневое управление накопителями

Способ присоединения диска к системе зависит от используемого интерфейса. Остальное определяется монтажными кронштейнами и кабелями. К счастью, разъемы SAS и SATA хорошо защищены.

При работе с параллельным интерфейсом SCSI необходимо перепроверить, что оба конца шины SCSI имеют терминаторы, длина кабеля не превышает максимально допустимую и новый целевой номер SCSI не конфликтует с контроллером или другим устройством, присоединенным к шине.

Даже если интерфейсы допускают подключение без перезагрузки системы ("горячее" подключение), все же безопаснее выключить систему перед тем, как вносить изменения в аппаратное обеспечение. Некоторые старые системы, такие как AIX, устанавливают конфигурацию устройств только во время загрузки, поэтому "горячее" подключение аппаратного обеспечения может не сразу вступить в силу на уровне операционной системы. При работе с интерфейсами SATA, "горячее" подключение зависит от реализации. Некоторые адаптеры компьютеров не поддерживают эту функциональную возможность.

## Верификация инсталляции на уровне аппаратного обеспечения

После инсталляции нового диска следует убедиться, что система знает о его существовании на самом низком из возможных уровней. Для персональных компьютеров это несложно: система BIOS показывает диски IDE и STAT, а большинство микросхем SCSI имеет собственный экран настроек, который вызывается перед загрузкой системы.

Для других типов аппаратного обеспечения может потребоваться перезагрузка системы и проверка результатов диагностики, выполняемой ядром. Например, одна из проверенных нами систем вывела на экран следующие сообщения для старого диска SCSI, присоединенного к адаптеру SCSI BusLogic.

```
scsi0: BusLogic BT-948
scsi: 1 host.
   vendor: SEAGATE  Model: ST446452W     Rev: 0001
   Type:   Direct-Access              ANSI SCSI revision: 02
Detected scsi disk sda at scsi0, channel 0, id 3, lun 3
scsi0: Target 3: Queue Depth 28, Asynchronous
SCSI device sda: hdwr sector=512 bytes. Sectors=91923356 [44884 MB] [44,8 GB]
```

Эту информацию можно обнаружить в файлах системного журнала после загрузки системы. Более подробно обработка сообщений, выдаваемых ядром при загрузке, описывается в разделе 11.2.

## Файлы накопителя

Вновь добавленный диск представляется файлами накопителя в каталоге **/dev**.

Все рассмотренные нами системы создают такие файлы автоматически, но системный администратор по-прежнему обязан знать, где искать файлы накопителя и какие из них соответствуют новому устройству. Форматирование неправильного накопителя — это прямой путь к катастрофе. В табл. 8.2 приведены соглашения об именах дисков, принятые в рассматриваемых операционных системах. Вместо демонстрации абстрактного шаблона, по которому дискам присваиваются имена, в табл. 8.2 приводится типичный пример выбора имени первого диска в системе.

**Таблица 8.2. Стандарты именования дисков**

| Система | Блочное устройство | Устройство | Раздел |
|---|---|---|---|
| Linux | **/dev/sda** | не используется | **/dev/sda1** |
| Solaris | **/dev/dsk/c0t0d0s2** | **/dev/rdsk/c0t0d0s2** | **/dev/dsk/c0t0d0s0** |
| HP-UX [a] | **/dev/disk/disk0** | **/dev/rdisk/disk0** | **/dev/disk/disk0_p1** [6] |
| AIX | **/dev/hdisk0** | **/dev/rhdisk0** | |

[a] Для сохранения совместимости система HP-UX использует также имена устройств, принятые в системе Solaris.

[6] Только для системы на процессоре Itanium; остальные системы не имеют разделов.

Столбцы, соответствующие блокам и устройствам, содержат пути к диску в целом, а столбец, соответствующий разделу, демонстрирует путь к конкретному разделу.

### *Дисковые накопители для системы Linux*

Имена дисков в системе Linux присваиваются ядром последовательно при нумерации разных интерфейсов и устройств, существующих в системе. При добавлении нового диска имя старого может измениться. Фактически даже перезагрузка системы может вызвать изменение имен.[5] *Никогда* не вносите изменения, не идентифицировав диск, с которым работаете, даже в устойчивой системе.

Система Linux предлагает несколько способов решения этой проблемы. Подкаталоги каталога **/dev/disk** содержат характеристики дисков, например их идентификационный номер, присвоенный производителем, или информацию о разъеме. Эти имена устройств (которые на самом деле представляют собой обычные ссылки на **/dev/sd***) являются постоянными, но они длинные и громоздкие.

На уровне файловых систем и массивов дисков система Linux использует уникальные строки, которые персистентно идентифицируют объект. Во многих случаях эти длинные строки скрыты, так что вам не придется работать с ними непосредственно.

[5] Для того чтобы решить эту проблему, система Linux использует идентификаторы UUID из файла **/etc/fstab**, а не имена устройств (см. раздел 8.9).

Система Linux не хранит файлы для отдельных накопителей или разделов дисков, поэтому следует указывать блочные устройства, когда необходимо сослаться на отдельное устройство.

Команда **parted -l** перечисляет размеры, таблицы разделов, номера моделей и производителей каждого диска, существующего в системе.

### Дисковые накопители для системы Solaris

Система Solaris присваивает дискам имена, подобные **/dev/[r]dsk/cWtXdYsZ**, где W — номер контроллера, **X** — целевой номер SCSI, Y — номер логического модуля SCSI (или номер LUN, который практически всегда равен 0), а Z — номер раздела (сектора). Здесь существует множество тонкостей: накопители ATA называются **cWdYsZ** (без пункта **t**), а диски могут иметь как ряд разделов в стиле DOS, обозначаемых как **p**Z, так и сектора в стиле Solaris, обозначаемые как **s**Z.

Эти файлы устройств на самом деле представляют собой всего лишь символические ссылки в дереве **/devices**, в котором размещены реальные файлы устройств. В целом, система Solaris предприняла попытку обеспечить постоянство имен устройств, даже при изменении аппаратного обеспечения. Как только диску присвоено определенное имя, он будет найден под этим именем в любой момент в будущем, если вы не измените контроллеры или целевые номера SCSI.

По соглашению сектор 2 соответствует полному, не разбитому на разделы диску. В отличие от системы Linux, система Solaris создает файлы устройства для каждого сектора и раздела, независимо от того, существуют ли они на самом деле. Система Solaris также поддерживает перекрывающиеся разделы, но это просто безумная идея. С таким же успехом компания Oracle могла бы поставлять системы Solaris вместе с заряженным пистолетом в нагрузку.

В системе Solaris "горячее" подключение должно работать хорошо. Когда вы добавляете новый диск, команда **devfsadmd** должны распознать его и создать соответствующие файлы устройств. При необходимости эту команду можно выполнить вручную.

### Дисковые накопители для системы HP-UX

Традиционно система HP-UX заимствовала правила именования дисков у системы Solaris, которая записывает в каталогах много информации об аппаратном устройстве. Однако в версии HP-UX 11i v3 эти каталоги были исключены и предпочтение было отдано гибким адресам, имеющим вид **/dev/disk/disk1**. Такие пути являются более устойчивыми и не заполняются деталями, относящимися к конфигурации системного аппаратного обеспечения.

Прежде чем загружать систему UNIX, от монитора PROM можно получить листинг системных устройств SCSI. К сожалению, точный способ выдачи этого листинга зависит от компьютера. После загрузки можно получить список дисков, выполнив команду **ioscan**.

```
$ sudo ioscan -fNn -C disk
Class I H/W Path          Driver  S/W  State H/W  TypeDescription
=============================================================
disk  3 64000/0xfa00/0x0  esdisk  CLAIMED    DEVICE  TEACDV-28E-B
          /dev/disk/disk3         /dev/rdisk/disk3
disk  4 64000/0xfa00/0x1  esdisk  CLAIMED    DEVICE HP 73.4GMAS3735NC
          /dev/disk/disk4         /dev/rdisk/disk4
```

```
            /dev/disk/disk4_p1      /dev/rdisk/disk4_p1
            /dev/disk/disk4_p2      /dev/rdisk/disk4_p2
            /dev/disk/disk4_p3      /dev/rdisk/disk4_p3
disk  5 64000/0xfa00/0x2  esdisk  CLAIMED    DEVICE HP 73.4GMAS3735NC
            /dev/disk/disk5        /dev/rdisk/disk5
            /dev/disk/disk5_p1      /dev/rdisk/disk5_p1
            /dev/disk/disk5_p2      /dev/rdisk/disk5_p2
            /dev/disk/disk5_p3      /dev/rdisk/disk5_p3
```

В каталогах **/dsk** и **/rdsk** все еще используются имена дисков старого стиля, при желании их можно использовать, по крайней мере, пока. Для того чтобы увидеть соответствие между имена в старом и новом стилях, следует выполнить команду **ioscan -m dsf**.

```
hp-ux$ ioscan  -m  dsf
Persistent DSF[6]      Legacy DSF(s)
================================
/dev/rdisk/disk3      /dev/rdsk/c0t0d0
/dev/rdisk/disk4      /dev/rdsk/c2t1d0
/dev/rdisk/disk4_p1   /dev/rdsk/c2t1d0s1
/dev/rdisk/disk4_p2   /dev/rdsk/c2t1d0s2
/dev/rdisk/disk4_p3   /dev/rdsk/c2t1d0s3
/dev/rdisk/disk5      /dev/rdsk/c2t1d0
/dev/rdisk/disk5_p1   /dev/rdsk/c2t1d0s1
/dev/rdisk/disk5_p2   /dev/rdsk/c2t1d0s2
/dev/rdisk/disk5_p3   /dev/rdsk/c2t1d0s3
```

Обратите внимание на то, что разделы теперь обозначаются буквой **p**, а не **s**, как в системе Solaris ("s" означает "slice" — сектор). В отличие от системы Solaris, для представления целых дисков система HP-UX использует такие имена, как **disk3** без суффикса раздела. В системе Solaris раздел 2 соответствует целому диску, а в системе HP-UX — это просто другой раздел.

Система, из которой взят данный пример, основана на процессоре Itanium и поэтому имеет разделы дисков. Другие системы HP-UX вместо разбиения дисков используют управление логическими томами.

### *Дисковые накопители для системы AIX*


Пути **/dev/hdisk** и **/dev/rhdisk** являются очень простыми. К сожалению, при изменении конфигурации аппаратного обеспечения имена дисков изменяются. Однако большинство дисков в системе AIX управляется менеджерами логических томов, поэтому имена устройств не имеют большого значения. Включая диск в группу томов, менеджер логического тома записывает на него уникальный идентификатор. Эти метки позволяют системе автоматически классифицировать диски, так что изменения в именах устройств не имеют таких последствий, как в других системах.

Для того чтобы увидеть список дисков, инсталлированных в системе, следует выполнить команду **lsdev -C -c disk**.

[6] DSF - device special file (специальный файл устройства).

## форматирование и плохое управление блоками

Все жесткие диски поступают отформатированными, причем фабричное форматирование не хуже любого другого. Лучше избегать низкоуровневого форматирования, если это не диктуется крайней необходимостью. В общем, никогда не следует форматировать заново новые диски.

Если на диске обнаруживаются ошибки чтения или записи, сначала необходимо проверить кабели, терминаторы и адресацию. Каждый из них может вызвать симптомы, похожие на существование плохих блоков. Если после проверки вы все еще уверены, что на диске есть дефект, лучше заменить его сразу на новый диск, а не ждать много часов, пока он отформатируется, в надежде, что проблемы исчезну сами по себе.

В процессе форматирования происходит запись адресов и временных отметок на пластинах, для того чтобы установить размеры каждого сектора. Кроме того, обнаруживаются плохие блоки и дефекты покрытия, препятствующие надежному чтению или записи. Все современные блоки имеют встроенные механизмы управления плохими блоками, поэтому ни вам, ни драйверу не следует беспокоиться об управлении дефектами. Для этой цели встроенные программы накопителя хранят в резервной области адреса хороших блоков.

Плохие блоки, проявившиеся после форматирования диска, могут обрабатываться автоматически, но могут и не обрабатываться. Если программа накопителя полагает, что эти блоки можно восстановить надежным образом, то вновь выявленный дефект может быть исправлен на лету, а данные записаны в новое место. При выявлении более серьезных или менее очевидных ошибок, накопитель прекращает выполнение операций чтений или записи и сообщает об ошибке операционной системе компьютера.

Диски ATA обычно не предназначены для форматирования за пределами фабрики. Однако вы можете получить информацию о программном обеспечении для форматирования дисков от его производителей, как правило, для системы Windows. Убедитесь, что программное обеспечение предназначено именно для того накопителя, который вы собираетесь форматировать, и точно следуйте инструкциям производителя.[7]

Диски SCSI форматируются сами по стандартной команде, поступающей от компьютера. Процедура отправки этой команды зависит от системы. На персональных компьютерах часто существует возможность посылать эту команду из системы BIOS контроллера SCSI. Для того чтобы отдать команду на форматирование диска SCSI от имени операционной системы, следует выполнить команду **`sg_format`** в системе Linux, **`format`** — в системе Solaris и **`mediainit`** — в системе HP-UX.

Для проверки целостности диска существуют разнообразные утилиты, которые выполняют запись данных в случайно выбранные области и считывают их оттуда. Тщательные тесты занимают много времени и, к сожалению, имеют небольшое прогностическое значение. Если вы подозреваете, что диск испорчен, и можете его просто заменить (или заказать новый в течение часа), то эти тесты можно проигнорировать. В противном случае запустите тест на ночь. Не беспокойтесь об "изнашивании" диска вследствие избыточного или агрессивного тестирования. Промышленные диски предназначены для постоянной работы.

[7] С другой стороны, если накопитель емкостью 1 Тбайт стоит 80 долл., зачем беспокоиться?

## Безопасное стирание дисков ATA

Начиная с 2000 года диски PATA и ATA реализуют команду "безопасного стирания", которая переписывает все данные на диске, используя метод, разработанный производителями для защиты от несанкционированного извлечения информации. Безопасное стирание сертифицировано Национальным институтом стандартов и технологии (NIST) и в большинстве случаев вполне удовлетворяет пользователей. По классификации Министерства обороны США, оно одобрено к использованию на уровне конфиденциальности ниже, чем "секретно".

Зачем вообще нужна такая функциональная возможность? Во-первых, файловые системы сами ничего не стирают, поэтому команды вроде **rm -rf*** оставляют данные, находящиеся на диске, нетронутыми и допускают их восстановление с помощью специальных программ. При ликвидации дисков об этом очень важно помнить, чтобы конфиденциальная информация попала в мусоросборник, а не на аукцион eBay.

Во-вторых, даже неавтоматическая запись каждого сектора на диске может оставлять магнитные следы, которые злоумышленник может выявить в лаборатории. Безопасное стирание переписывает данные столько раз, сколько нужно для того, чтобы уничтожить побочные сигналы. Остаточные магнитные сигналы для большинства организаций не составляют большой проблемы, но всегда полезно знать, что вы защищены от разглашения конфиденциальных данных организации.

В заключение, безопасное стирание делает накопители SSD абсолютно пустыми. Это позволяет повысить их производительность в ситуациях, в которых нельзя использовать команду TRIM в стандарте ATA (команду для стирания блока) либо потому, что файловая система, используемая на накопителе SSD, не способна ее выполнить, либо потому, что накопитель SSD присоединен через адаптер компьютера или интерфейс RAID, которые не пропускают команду TRIM.

К сожалению, поддержка команды безопасного стирания диска в системе UNIX остается слабой. С этой точки зрения для очистки накопителей их лучше переключить в систему Windows или Linux. Программное обеспечение операционной системы DOS можно найти на сайте исследовательской группы Center of Magnetic Recording Research `tinyurl.com/2xoqqw`. Утилита MHDD также поддерживает безопасное стирание с помощью команды **fasterase** (см. сайт `tinyurl.com/2g6r98`).

В системе Linux можно использовать команду **hdparm**.

```
$ sudo hdparm --user-master u -security-set-pass password /dev/sda8
$ sudo hdparm --user-master u -security-set-erase password /dev/sda
```

Команда безопасного стирания в стандарте ATA не имеет аналогов в стандарте SCSI, но команда "форматировать модуль" в стандарте SCSI, описанная выше, представляется вполне разумной альтернативой. Другой возможностью является обнуление секторов с помощью команды **dd if=/dev/zero of=***накопитель* **bs=8k**.

Многие системы имеют утилиту **shred**, которая выполняет безопасное стирание отдельных файлов. К сожалению, ее работа основана на предположении, что блоки файлов могут быть перезаписаны на том же месте. Однако во многих ситуациях это условие не выполняется (в частности, это относится к любым файловым системам на любых накопителях SSD, любым логическим томам, имеющим механизм мгновенных копий, и,

[8] Команда безопасного стирания в стандарте ATA защищена паролем, для того чтобы затруднить к ней доступ. Следовательно, перед ее вызовом необходимо ввести пароль. Впрочем, не стоит его записывать, вы можете установить его заново, когда захотите. В этом случае блокировка накопителя никакой опасности не несет.

возможно, ко всей файловой системе ZFS), поэтому полезность универсальной утилиты **shred** вызывает сомнения.

Для очистки всей дисковой системы на персональном компьютере существует программа Darik's Boot and Nuke (`dban.org`). Этот инструмент запускается со своего загрузочного диска, поэтому он не используется каждый день. Тем не менее он довольно удобен для вывода из эксплуатации старого аппаратного обеспечения.

## Команда hdparm: параметры диска и интерфейса (Linux)

В системе Linux команда **hdparm** имеет немного больше возможностей, чем команды безопасного стирания. Обычно она используется для взаимодействия со встроенными программами жестких дисков SATA, IDE и SAS. Помимо прочего, команда **hdparm** может устанавливать настройки электропитания накопителя, включать или отключать подавление шумов, устанавливать флаг "только для чтения" и распечатывать информацию о накопителе. Некоторые из этих опций относятся и к накопителям SCSI (в современных ядрах системы Linux).

Синтаксис этой команды выглядит следующим образом.

**hdparm** [*параметры*] *накопитель*

Источники параметров доступны, но большинство из них интересно только для разработчиков драйверов и ядер. Параметры, представляющие интерес для администраторов, приведены в табл. 8.3.

**Таблица 8.3. Параметры команды hdparm, полезные для системных администраторов**

| Параметр | Функция |
|---|---|
| **-I**[a] | Выдает много идентифицирующей информации и данных о состоянии накопителя |
| **-M** значение | Устанавливает акустические параметры |
| **-S** значение | Устанавливает временную задержку для режима ожидания (замедление вращения) |
| **-y** | Немедленно переводит накопитель в режим ожидания |
| **-C** | Посылает запросы о текущем состоянии системы управления электропитанием накопителя |
| **-T** | Быстро тестирует полосу пропускания интерфейса (без реального чтения диска) |
| **-t** | Быстро тестирует скорость считывания данных с пластины в компьютер |

[a] Это прописная буква `I`, как в слове India.

Для того чтобы проверить, что каждое устройство работает в максимально быстром режиме DMA, следует использовать команду **hdparm -I**. Команда **hdparm** перечисляет все поддерживаемые модели дисков и помечает их текущий режим звездочкой, как показано в примере, приведенном ниже.

```
linux$ sudo hdparm -I /dev/sdf

/dev/sdf

ATA device, with non-removable media
   Model number:       WDC  WD1001FALS-00J7B0
   Serial Number:      WD-WMATV0998277
   Firmware Revision:  05.00K05
   Transport:          Serial, SATA 1.0a, SATA II Extensions, SATA Rev 2.5
...
```

```
Capabilities:
   LBA, IORDY (can be disabled)
   Queue depth: 32
   Standby timer values: spec'd by Standard, with device specific minimum
   R/W multiple sector transfer: Max = 16  Current = 16
   Recommended acoustic management value: 128, current value: 254
   DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5 *udma6
   Cycle time: min=120ns recommended=120ns
PIO: pio0 pio1 pio2 pio3 pio4
   Cycle time: no flow control=120ns  IORDY flow control=120 ns
...
```

Во всех современных системах оптимальный режим DMA устанавливается по умолчанию. Если это не так, проверьте систему BIOS и журналы ядра для поиска информации, которая поможет понять причины.

Многие накопители предлагают управление акустическим режимом, позволяя замедлять движение считывающих головок и уменьшать громкость тиканья и звуковых импульсов. Накопители, поддерживающие такую возможность, обычно поставляются с уже включенным механизмом управления акустическим режимом, но это иногда не важно для накопителей, установленных в серверной комнате. Для того чтобы отключить эту возможность, следует выполнить команду **hdparm -M 254**.

Большая часть электроэнергии, потребляемой накопителем, расходуется на вращение пластин. Если ваши диски используются редко и вы имеете возможность установить задержку на 20 с или управлять повторным запуском двигателя дисков, выполните команду **hdparm -S**, чтобы включить возможность внутреннего управления электропитанием дисков. Аргумент **-S** задает время простоя, после которого накопитель переходит в режим ожидания и отключает двигатель. Эта величина занимает один байт, поэтому кодирование представляет собой нелинейную задачу. Например, значения от 1 до 240 умножаются на 5 секунд, а значения от 241 до 251 — на 30 минут. В ходе выполнения команда **hdparm** демонстрирует свою интерпретацию этого значения. Его проще угадать, подобрать и повторить, чем определить, точно следуя правилам кодирования.

Команда **hdparm** предусматривает простой тест производительности накопителя, позволяющий оценить влияние изменений конфигурации. Параметр **-T** означает считывание данных из кеша накопителя и выдачу скорости передачи данных по шине, не зависящую от пропускной способности физических устройств накопителя. Параметр **-t** означает считывание с пластин. Как и следовало ожидать, считывание с пластин происходит намного медленнее.

```
$ sudo /sbin/hdparm -Tt /dev/hdb

/dev/sdf:
   Timing cached reads:  2092 MB in 2.00 seconds = 1046.41 MB/sec
   Timing buffered disk reads:  304 MB in 3.00 seconds = 101.30 MB/sec
```

Скорость, равная 100 Мбайт/с, близка к максимальному пределу современных накопителей емкостью 1 Тбайт, так что эти результаты (и информация, приведенная выше при иллюстрации команды **hdparm -I**) подтверждают, что накопитель настроен правильно.

## Мониторинг жесткого диска с помощью стандарта SMART

Жесткие диски представляют собой отказоустойчивые системы, использующие кодирование с исправлением ошибок, и встроенные программы с развитой логикой для

сокрытия своих недостатков от операционной системы компьютера. В некоторых случаях жесткие диски сообщают операционной системе о неисправимой ошибке только после многочисленных попыток исправить ее. Было бы хорошо заранее распознавать такие ошибки, пока не разразился кризис.

Устройства ATA, включая накопители SATA, реализуют подробную форму отчетов о состоянии накопителя, которая иногда позволяет предсказать сбой. Этот стандарт под названием SMART ("self-monitoring, analysis, and reporting technology") предусматривает более 50 параметров, которые может анализировать компьютер.

Ссылаясь на исследование дисков, проведенное компанией Google (см. раздел 8.11), часто утверждают, что данные стандарта SMART не могут предсказать отказ накопителя. Это не совсем так. На самом деле компания Google обнаружила, что четыре параметра стандарта SMART обладают высокой прогнозной точностью, но сам сбой невозможно предотвратить с помощью изменения настроек стандарта SMART. Среди накопителей, давших сбой, 56% не содержали изменений в этих четырех параметрах. С другой стороны, мы считаем, что предсказание сбоя у более половины дисков является неплохим результатом!

Четырьмя чувствительными параметрами стандарта SMART являются количество ошибок сканирования (scan error count), количество повторного распределения памяти (reallocation count), количество автономного повторного распределения памяти (off-line reallocation count) и количество секторов "с испытательным сроком" (number of sectors "of probation"). Все эти значения должны быть равны нулю. Ненулевые значения этих параметров повышают вероятность отказа в течение 60 дней в 39, 14, 21 и 16 раз соответственно.

Для того чтобы извлечь пользу из параметров стандарта SMART, необходимо иметь специальное программное обеспечение, которое опрашивает накопители и прогнозирует вероятность сбоя. К сожалению, стандарты отчетов варьируются в зависимости от производителей накопителей, поэтому декодирование параметров не всегда является простой задачей. Большинство мониторов SMART накапливают основные параметры, а затем ищут внезапные изменения в неблагоприятном направлении, вместо того чтобы интерпретировать их абсолютные величины. (В соответствии с отчетом компании Google, учет этих "мягких" параметров стандарта SMART в дополнение к "большой четверке" позволяет предсказать 64% сбоев.)

Стандартным программным обеспечением стандарта SMART для контроля накопителей в системах UNIX и Linux является пакет smartmontool с сайта `smartmontool.sourceforge.net`. В системах SUSE и Red Hat он инсталлируется по умолчанию; в системе Ubuntu для его инсталляции необходимо выполнить команду **`apt-get install smartmontools`**. Для того чтобы запустить этот пакет в системе Solaris, его необходимо скомпилировать из исходного кода.

Пакет **`smartmontool`** состоит из демона **`smartd`**, который постоянно следит за накопителями, и команды **`smartctl`**, которую можно использовать для выполнения интерактивных запросов или сценариев. Демон имеет один конфигурационный файл, как правило, **`/etc/smartd.conf`**, содержащий подробные комментарии и много примеров.

Накопители SCSI имеют собственную систему отчетов о нестандартном состоянии, но, к сожалению, они менее подробные, чем отчеты системы SMART. Пакет smartmontools пытается включить накопители SCSI в свою схему, но прогнозируемая значимость данных стандарта SCSI является мнее ясной.

## 8.6. Разбиение диска

Разбиение логического тома и управление им — это два способа разделения диска (или совокупности дисков в случае менеджера LVM) на отдельные части определенного объема. Все рассматриваемые нами операционные системы поддерживают управление логическими томами, но только Linux, Solaris и иногда HP-UX допускают традиционное разбиение.

Каждый раздел можно отдать под контроль менеджера логических томов, но сам логический том разделить нельзя. Разбиение — это самый низкий уровень управление дисками.

В системе Solaris разбиение необходимо, но по существу носит рудиментарный характер; система ZFS скрывает его так хорошо, что пользователь может об этом даже не догадываться. В этом разделе приведена общая информация, которая может оказаться полезной для администратора системы Solaris, но поскольку с процедурной точки зрения система Solaris резко отличается от систем Linux, HP-UX и AIX, читатель может сразу перейти к разделу 8.10. (Впрочем, читая этот раздел дальше, вы узнаете, что команда **zpool create** *новый_пул новый_накопитель* решает большую часть проблем, связанных с конфигурацией.)

И разбиение диска, и логические тома облегчают резервное копирование, предотвращая незаконный захват пользователями дискового пространства и потенциальные повреждения от программ, вышедших из-под контроля. Все системы имеют корневой "раздел", содержащий большинство данных о конфигурации локального компьютера. Теоретически, для того чтобы перевести систему в однопользовательский режим работы, достаточно одного корневого раздела. Различные подкаталоги (самые распространенные из них — **/var**, **/usr**, **/tmp**, **/share** и **/home**) можно разделять на более подробные разделы или тома. Большинство систем имеет, по крайней мере, одну область подкачки.

Мнения о наилучшем способе разбиения диска по умолчанию разделяются. Рассмотрим некоторые из них.

- Целесообразно иметь резервный корневой накопитель, который можно загрузить в ситуации, когда с обычным корневым разделом что-то не так. В идеале резервный корневой раздел и обычный корневой раздел (root device) должны существовать на разных дисках, чтобы иметь возможность защищаться от проблем, связанных с аппаратным обеспечением и повреждениями. Однако даже если резервный корневой раздел находится на том же диске, все равно он имеет определенную ценность.[9]
- Проверьте, можете ли вы загружаться с резервного корневого раздела. Эта процедура зачастую нетривиальна. Возможно, во время загрузки вам понадобится передать специальные аргументы для ядра или внести минимальные изменения в конфигурацию альтернативного корневого раздела, чтобы все работало гладко.

Поскольку корневой раздел часто дублируется, он должен быть достаточно небольшим, чтобы две его копии не потребляли неразумно большой объем дискового пространства. Это основная причина, по которой каталог **/usr** часто представляет собой отдельный том; он содержит большинство системных библиотек и данных.

[9] Системы Solaris и HP-UX имеют даже файловые системы "динамического корневого диска", облегчающие эксплуатацию нескольких корней. См. справочные страницы, посвященные командам beadm или lucreate в системе Solaris и drd в системе HP-UX.

Размещение каталога **/tmp** в отдельной файловой системе ограничивает размеры временных файлов и позволяет их не восстанавливать. Некоторые системы для повышения производительности используют для хранения каталога **/tmp** файловую систему, расположенную в оперативной памяти. Файловые системы, расположенные в оперативной памяти, поддерживаются областью подкачки, поэтому они хорошо работают в большинстве ситуаций.

Поскольку файлы системного журнала хранятся в каталоге **/var**, целесообразно, чтобы каталог **/var** представлял собой отдельный раздел диска. Если каталог **/var** хранится в маленьком корневом разделе, то легко переполнить этот раздел и вызвать остановку машины.

Пользовательские рабочие каталоги полезно размещать в отдельном разделе или каталоге. В этом случае, даже если корневой раздел будет поврежден или разрушен, пользовательские данные с высокой вероятностью останутся целыми. И наоборот, система сможет продолжать работу, даже если ошибочный пользовательский сценарий переполнит каталог **/home**.

Разделение области подкачки между несколькими физическими дисками повышает производительность работы. Этот метод работает и по отношению к файловым системам; размещайте интенсивно используемые файловые системы на разных дисках. Эта тема освещается в разделе 29.4.

Добавляя модули памяти в компьютер, увеличьте область подкачки. Более подробно виртуальная память описывается в разделе 29.4.

Резервное копирование раздела можно упростить, если весь раздел заполняет один фрагмент памяти на диске (см. раздел 10.1).

Старайтесь разделять быстро изменяющуюся информацию на кластеры, размещенные в нескольких разделах, которые часто подвергаются резервному копированию.

## Традиционное разбиение

Системы, допускающие разбиение, реализуют его, записывая "метку" в начало диска, чтобы определить диапазон блоков, включенных в каждый раздел. Подробности этой процедуры могут варьироваться; метка часто может существовать одновременно с другой установочной информацией (например, загрузочным блоком) и часто содержит дополнительную информацию, например имя или уникальный идентификатор всего диска. В системе Windows эта метка называется MBR (master boot record — главная загрузочная запись).

Драйвер устройства, представляющий диск, считывает эту метку и использует таблицу разбиения диска для вычисления физического местоположения каждого раздела. Как правило, каждому разделу соответствует один или два файла устройства (для блочного устройства и для устройства посимвольного ввода-вывода; в системе Linux есть только блочные устройства). Кроме того, отдельный набор файлов устройств представляет диск в целом.



В системе Solaris разделы называются секторами ("slices"). Точнее, они называются секторами, если реализованы с помощью метки в стиле системы Solaris, и разделами, если реализованы с помощью метки MBR в стиле системы Windows. Сектор 2 содержит все пространство диска, иллюстрируя несколько устрашающий факт, что один блок диска может принадлежать нескольким секторам одновременно. Возможно, слово "сектор" было выбрано потому, что "раздел" предполагает простое разбиение, а секторы могут перекрываться. В остальных ситуациях эти термины являются синонимами.

Несмотря на всеобщую доступность менеджером логических томов, в некоторых ситуациях необходимо или желательно традиционное разбиение.

- На персональных компьютерах загрузочный диск может иметь таблицу разбиения. Большинство систем предпочитает разбиение с помощью метки MBR (см. далее раздел "Разбиение диска в стиле системы Windows"), но системы на процессоре Itanium требуют разбиения с помощью таблицы GPT (см. ниже). Диски с данными могут оставаться неразделенными.

🕮 Более подробно двойственная загрузка системы Windows описывается в разделе 3.3.

- Инсталлирование меток MBR делает диск понятным для системы Windows, даже если содержание его отдельных разделов остается недоступным. Если вы хотите взаимодействовать с системой Windows (например, посредством двойственной загрузки), необходимо инсталлировать метки Windows MBR. Однако даже если вы не собираетесь делать что-нибудь подобное, иногда целесообразно учесть повсеместную распространенность системы Windows и вероятность того, что ваш диск в один прекрасный день может вступить с ней в контакт.
- Текущие версии системы Windows являются хорошо продуманными и *никогда* не записывают данные на диск, который они не могут расшифровать. Однако они определенно могут предложить эти действия администратору, который на них зарегистрируется. На соответствующем диалоговом окне даже есть полезная кнопка "OK, преврати этот диск в кашу!".[10] Ничего плохого не произойдет, пока кто-то не сделает ошибку, но обеспечение безопасности — это структурный и организационный процесс.
- Разделы занимают точно определенное место на диске и гарантируют локальность ссылок. Логические тома этого не обеспечивают (по крайней мере, по умолчанию). В большинстве случаев это не имеет большого значения. Однако поиск в смежных областях выполняется быстрее, нежели поиск в отдаленных ячейках, и пропускная способность внешних цилиндров диска (т.е. цилиндров, содержащих блоки с наименьшими номерами) может превосходить пропускную способность внутренних цилиндров на 30% и более.[11] Для ситуаций, в которых требуется высокая производительность, разбиение диска можно использовать для достижения дополнительного выигрыша. (Для восстановления потерянной гибкости всегда можно применить управление логическими томами *внутри* раздела.)
- Системы RAID (см. раздел 8.7) используют диски или разделы одинакового объема. Хотя конкретная реализация системы RAID может допускать диски разных объемов, вероятно, она будет использовать только те диапазоны блоков, которые являются общими для всех дисков. Вместо того чтобы терять избыточное пространство, его можно изолировать в отдельном разделе. Однако в этом случае следует предусмотреть запасной раздел для редко используемых данных, иначе такое разбиение снизит производительность работы массива RAID.

[10] Хорошо, хорошо. Возможно, на ней написано просто "Форматировать" или "OK", но это именно то, что на ней **должно** быть написано.

[11] Использование только внешних цилиндров диска для повышения производительности называется "коротким рабочим ходом" (short stroking). Здесь ходом называется перемещение головки диска.

## Разбиение диска в стиле системы Windows

Метка Windows MBR занимает один блок диска размером 512 байт; большую часть этого блока занимает код загрузки. Оставшегося места достаточно для определения только четырех разделов. Эти разделы называются первичными, потому что они определяются непосредственно в метке MBR.

Один из первичных разделов можно определить как "расширенный". Это значит, что он содержит собственную вспомогательную таблицу разделов. Расширенный раздел — это обычный раздел, который занимает определенное место на физическом диске. Вспомогательная таблица разделов хранится в начале данных, относящихся к разделу.

Разделы, создаваемые внутри расширенного раздела, называются вторичными. Они являются частью расширенного раздела.

Запомните следующие эмпирические правила установки дисков, разбитых в стиле системы Windows. Первое из них является настоящим правилом. Остальные относятся к работе с системами BIOS, загрузочными блоками или операционными системами.

- На диске может существовать только один расширенный раздел.
- Расширенный раздел должен быть последним разделом, определенным в метке MBR; после него нельзя определять никаких первичных разделов.
- В некоторых старых операционных системах вторичные разделы нежелательны. Для того чтобы избежать неприятностей, инсталлируйте операционную систему в первичном разделе.

Схема разбиения дисков в системе Windows позволяет пометить один из разделов как "активный". Загрузчики ищут активный раздел и пытаются загрузить операционную систему из него.

Кроме того, каждый раздел имеет однобайтовый атрибут типа, который обозначает содержимое раздела. Как правило, эти коды представляют либо типы файловой системы, либо операционные системы. Эти коды не присваиваются централизованно, но со временем были приняты определенные соглашения. Они сформулированы Андрисом Э. Брауэром (Andries E. Brouwer) на сайте `tinyurl.com/part-types`.

Команда MS-DOS, осуществляющая разбиение жесткого диска, называется **`fdisk`**. Большинство операционных систем, поддерживающих разбиение в стиле системы Windows, унаследовали это имя для обозначения своих собственных команд разбиения, но они очень разнообразны. Сама система Windows от нее отказалась: ее современная версия инструмента для разбиения диска называется **`diskpart`**. Кроме того, система Windows имеет графический пользовательский интерфейс для разбиения дисков, который доступен благодаря утилите Disk Management из консоли управления **`mmc`**.

Не имеет значения, с помощью какой операционной системы вы разбили диск на разделы — Windows или какой-то другой. Окончательный результат будет таким же.

## GPT: таблица разделов GUID

Проект компании Intel по созданию расширяемого микропрограммного интерфейса (extensible firmware interface — EFI) был направлен на замену неустойчивых соглашений, касающихся систем BIOS персональных компьютеров, более современной и функциональной архитектурой.[12] Несмотря на то что системы, полноценно использующие рас-

[12] Проект EFI недавно был переименован в UEFI. Буква U означает "unified" (универсальный). Этот проект поддерживается многими производителями. Однако название EFI употребляется чаще. По существу, названия UEFI и EFI являются синонимами.

ширяемый микропрограммный интерфейс, в настоящее время являются редкостью, схема разбиения EFI получила широкое распространение среди операционных систем. Основная причина этого заключается в том, что метка MBR не поддерживает диски, объем которых больше 2 Тбайт. Поскольку диски объемом 2Тбайт стали широко доступными, эту проблему нужно срочно решать.

Схема разделения EFI, известная как "таблица разделов GUID", или GPT, устраняет очевидные недостатки метки MBR. Она определяет только один тип разбиения, а самих разделов может быть сколько угодно. Каждый раздел имеет тип, определенный 16-байтовым идентификатором (глобально уникальным ID, или GUID), который не требует централизованного управления.

Следует подчеркнуть, что таблица GPT сохраняет простейшую совместимость с системами, использующими метку MBR, записывая метку MBR в первый блок таблицы разделов. Эта "ложная" метка MBR позволяет диску выглядеть так, будто он содержит только один раздел MBD (по крайней мере, в пределах до 2 Тбайт). Само по себе это не имеет значения, но присутствие "ложной" метки MBR защищает диск от переформатирования простыми системами.

Версии системы Windows, начиная с версии Vista, открыли эру дисков GPT для хранения данных, но только системы, поддерживающие расширяемый микропрограммный интерфейс, могут загружаться с этих дисков. Система Linux и ее загрузчик GRUB продвинулись дальше: они поддерживают диски GPT, с которых могут загружаться любые системы. Системы Mac OS, основанные на процессорах компании Intel, поддерживают обе схемы разбиения EFI и GPT. Система Solaris понимает схему разбиения GPT, а система ZFS использует ее по умолчанию. Однако загрузочные диски системы Solaris не могут использовать разбиение GPT.

Несмотря на то что разбиение GPT хорошо принято ядрами операционных систем, ее поддержка среди утилит управления дисками остается эпизодической. Разбиение GPT остается форматом "далекого будущего". В настоящее время нет разумных причин для его использования на дисках, которые этого не требуют (т.е. на дисках объемом до 2 Тбайт).

## Разбиение дисков в системе Linux

Системы семейства Linux предусматривают несколько вариантов разбиения дисков. В них основным средством для разбиения дисков является команда **fdisk**. Инструмент для разбиения дисков в системе GNU запускается из командной строки, понимает несколько форматов меток (включая метки системы Solaris), может перемещать разделы и изменять их размеры, а также просто создавать и удалять их. Его версия с графическим пользовательским интерфейсом под названием **gparted** запускается в среде GNOME. Другой возможностью является утилита **cfdisk**, представляющая собой прекрасную альтернативу команде **fdisk**, использующую терминалы.

Утилиты **parted** и **gparted** теоретически могут изменять размеры некоторых типов файловых систем вместе с разделами, которые они содержат, но на домашней странице соответствующего проекта эта функциональная возможность названа "дефектной и ненадежной". Утилиты, предназначенные для файловых систем, лучше выполняют их настройку, но, к сожалению, утилита **parted** не имеет команды "изменить размер раздела, но не файловой системы". Если это необходимо, следует использовать утилиту **fdisk**.

В целом, мы рекомендуем использовать утилиту **gparted**, а не **parted**. Обе утилиты просты, но **gparted** позволяет задавать размеры разделов, а не начало и конец диапазона блоков. Для разбиения загрузочного диска лучше всего использовать графические инсталляторы из дистрибутивных пакетов, так как они обычно предлагают разбиение, которое является оптимальным для данного дистрибутива.

## Разбиение дисков в системе Solaris

Файловая система ZFS автоматически помечает диски, применяя таблицу разделов GPT. Однако разбить диски на разделы можно и вручную, используя команду **format**. В операционных системах x86 можно также применять команду **fdisk**. Оба инструмента предусматривают работу с меню и являются относительно простыми.

Команда **format** предоставляет пользователю удобный список дисков, в то время как команда **fdisk** требует указать диск в командной строке. К счастью, утилита **format** имеет команду **fdisk**, запускающую ее как подпроцесс, поэтому утилиту **format** можно рассматривать как оболочку, позволяющую найти требуемый диск.

Система Solaris допускает три схемы разбиения дисков: Windows MBR, GPT и старые таблицы разбиения Solaris, известные как SMI. Для загрузочных дисков следует использовать схемы MBR или SMI, в зависимости от аппаратного обеспечения и версии системы, Solaris или OpenSolaris. Пока, вероятно, лучше придерживаться этих правил для всех дисков объемом до 2 Тбайт, разбиваемых на разделы вручную.

## Разбиение дисков в системе HP-UX

Операционная система HP применяет разбиение только для загрузочных дисков компьютеров на процессорах Itanium (Integrity), требующих использования таблицы GPT и разбиения загрузочного диска с помощью интерфейса EFI. Команда **idisk** распечатывает и создает таблицы разделов. Она не работает в интерактивном режиме, а просто считывает план разбиения из файла или со стандартного средства ввода и использует его для создания таблицы разделов.

Спецификации команды **idisk** очень просты. Первая строка содержит только количество создаваемых разделов. Каждая следующая строка содержит тип разбиения (EFI, HPUX, HPDUMP или HPSP для подкачки), символ пробела и спецификацию объема, например 128 Мбайт или 100%. Если используются проценты, то они интерпретируются по отношению к пространству на накопителе, оставшемуся после выделения памяти для предыдущих разделов.

# 8.7. RAID: избыточные массивы недорогих дисков

Даже при резервном копировании последствия сбоя диска на сервере могут быть разрушительными. RAID (redundant arrays of inexpensive disks — избыточные массивы недорогих дисков) — это система, распределяющая или дублирующая данные с помощью многочисленных дисков.[13] Система RAID не только помогает избежать потери данных,

[13] Аббревиатуру RAID иногда расшифровывают как "redundant arrays of independent disks" ("избыточные массивы независимых дисков"). Оба варианта с исторической точки зрения являются правильными.

но и минимизирует время простоя, связанного с отказом аппаратуры (часто сводя его к нулю), и потенциально повышает производительность.

Систему RAID можно реализовать с помощью специального оборудования, благодаря которому операционная система интерпретирует группу жестких дисков как один составной накопитель. Кроме того, ее можно реализовать так, чтобы операционная система просто считывала или записывала данные на жестких дисках по правилам системы RAID.

## Программная и аппаратная реализации системы RAID

Поскольку диски сами по себе являются узким местом в реализации системы RAID, нет причин предполагать, что аппаратная реализация системы RAID всегда будет работать быстрее, чем оперативная.

В прошлом аппаратная система RAID была доминирующей по двум причинам: из-за недостатка программной поддержки (отсутствие прямой поддержки со стороны операционных систем) и способности аппаратного обеспечения буферизовать записи в форме некой энергонезависимой памяти.

Последнее свойство, упомянутое выше, повышает производительность, поскольку записи заполняются мгновенно. Кроме того, оно защищает диск от потенциального повреждения, которое получило название "RAID 5 write hole". Этот эффект описывается ниже. Однако будьте осторожны: многие широко распространенные "микросхемы RAID" для персональных компьютеров вообще не имеют энергонезависимой памяти; на самом деле они представляют собой хорошо известный интерфейс SATA с элементами встроенного программного обеспечения RAID. Реализации системы RAID на материнских платах персональных компьютеров также относятся к этой категории. В таких случаях лучше вообще отказаться от возможностей RAID в системах Linux или OpenSolaris.

Недавно произошел сбой контроллера диска на важном промышленном сервере. Несмотря на то что данные были скопированы на нескольких физических накопителях, неисправный контроллер RAID разрушил данные на всех дисках. В результате пришлось более двух месяцев долго и нудно восстанавливать магнитную ленту, пока сервер не был восстановлен полностью. Для управления средой RAID восстановленный сервер теперь использует программное обеспечение ядра, тем самым исключая возможность нового сбоя контроллера RAID.

## Уровни системы RAID

Система RAID может выполнять две основные операции. Во-первых, она может повысить производительность, "разбросав" данные по многим накопителям и позволив нескольким дискам одновременно передавать или получать поток данных. Во-вторых, она может реплицировать данные по многим накопителям, уменьшая риск, связанный со сбоем одного диска.

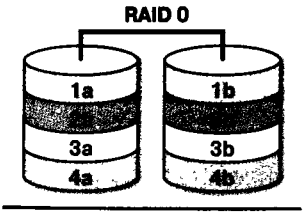
Репликация может осуществляться в двух видах: зеркальное отражение (mirroring), при котором блоки данных побитово репродуцируются на нескольких накопителях, и схемы четности (parity schemes), в которых один или несколько накопителей содержат контрольную сумму для проверки ошибок на остальных накопителях. Зеркальное отражение быстрее, но занимает много места на диске. Схемы четности более экономно используют дисковую память, но работают медленнее.

Система RAID традиционно описывается в терминах "уровней", определяющих конкретные детали параллелизма и избыточности. Возможно, этот термин является неудач-

ным, потому что более высокий уровень не обязательно оказывается лучшим. Уровни — это просто разные конфигурации; вы можете использовать любой из них.

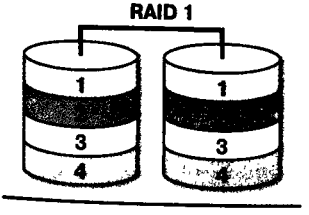
На рисунках, приведенных ниже, числа обозначают магнитные ленты, а буквы a, b и c — блоки данных на этих лентах. Блоки, помеченные буквами p и q, являются блоками четности.

- "Линейный режим", известный также как JBOD (just a bunch of disks — просто группа дисков), нельзя даже назвать реальным уровнем системы RAID. Его реализует каждый контроллер RAID. В режиме JBOD адресные блоки нескольких накопителей конкатенируются, образуя один большой виртуальный накопитель. Это не обеспечивает ни избыточности данных, ни повышения производительности. В настоящее время функциональная возможность JBOD лучше обеспечивается с помощью менеджера логических томов, а не контроллера RAID.
- Уровень RAID 0 используется только для повышения производительности. Он объединяет два или более накопителей одинаковых размеров, но, вместо их объединения последовательно один за другим, он распределяет данные между дисками, входящими в пул. Операции последовательного чтения и записи в этом случае оказываются распределенными между несколькими дисками, что снижает время записи и доступа к диску.

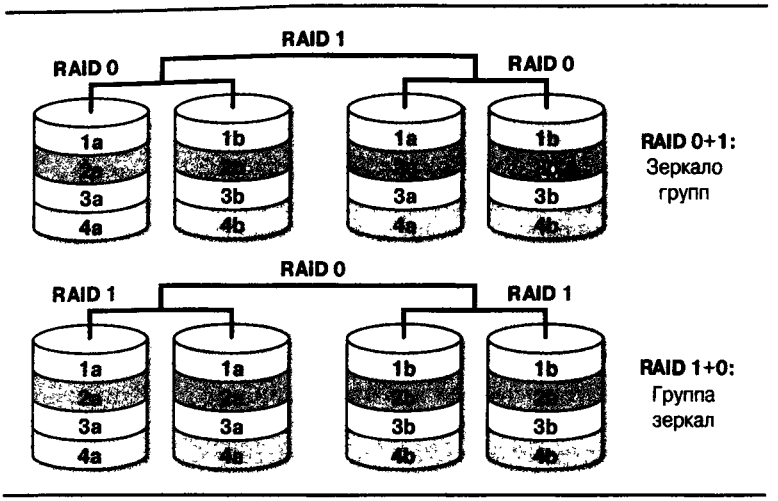



Обратите внимание на то, что надежность уровня 0 *ниже*, чем надежность отдельных дисков. Вероятность отказа массива из двух дисков в течение года примерно в два раза выше, чем у отдельного диска, и т.д.

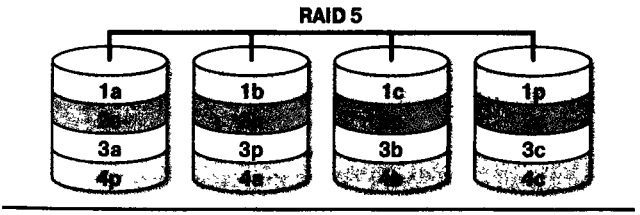
- Уровень RAID 1 известен как "зеркальное отражение". Записи дублируются одновременно на нескольких дисках. Эта схема замедляет процесс записи по сравнению с записью данных на отдельный диск. Однако она обеспечивает скорость считывания, сравнимую с уровнем RAID 0, потому что чтение можно распределять между несколькими дублирующими накопителями.



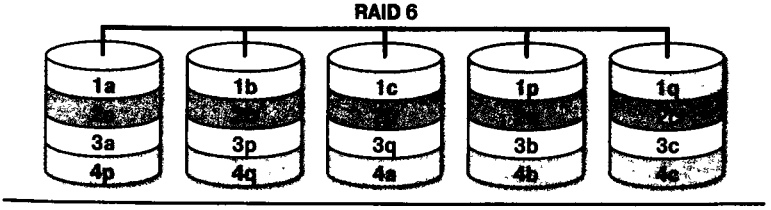

- Уровни RAID 1+0 и 0+1 представляют собой группы зеркал или зеркала групп. С логической точки зрения они представляют собой сочетание уровней RAID 0 и RAID 1, но многие контроллеры и программы обеспечивают их непосредственную поддержку. Цель обоих режимов — одновременно достичь производительности уровня RAID 0 и избыточности уровня RAID 1.




- Уровень RAID 5 распределяет и данные, и информацию о четности, добавляя избыточность и одновременно повышая производительность чтения. Кроме того, уровень RAID 5 более эффективно использует дисковое пространство, чем уровень RAID 1. Если массив состоит из N накопителей (требуется не менее трех), то N–1 из них могут хранить данные. Следовательно, эффективность использования дисковой памяти на уровне RAID 5 превышает 67%, в то время как зеркальное отражение не может превысить 50%.




- Уровень RAID 6 аналогичен уровню RAID 5 с двумя дисками четности. Массив RAID 6 может сохранить работоспособность при полном отказе двух накопителей без потери данных.




Уровни RAID 2, 3 и 4 хотя и определены, но используются редко. Менеджеры логических томов обычно могут и распределять данные (RAID 0), и осуществлять зеркальное отражение (RAID 1).

Для обычных конфигураций распределения и зеркального отражения система Linux предоставляет выбор: использовать специальную систему RAID (например, команду **md**; см. ниже) или менеджер логического диска. Подход LVM, вероятно, является более гибким, хотя команда **md** может оказаться немного более предсказуемой. Если у вас есть возможность использовать команду **md**, вы можете продолжать использовать механизм LVM для управления пространством на томе RAID. Для уровней RAID 5 и RAID 6 вы должны использовать команду **md**.

Файловая система ZFS в операционной системе Solaris осуществляет и распределение данных, и зеркальное отражение, и конфигурации, аналогичные уровням RAID 5 и RAID 6, как будто система RAID, менеджер логического тома и файловая система объединились в одно целое. Архитектура ZFS помещает зеркальное отражение и организацию проверки четности на самый низкий уровень, в то же время автоматически выполняя распределение данных среди накопителей, входящих в пул (уровень этой конфигурации на единицу выше). Это прекрасный способ реализации таких возможностей, поскольку он сохраняет ясность конфигурации RAID. Более подробно система ZFS описывается в разделе 8.10.

Управление логическим томом расширяет поддержку конфигурации RAID со стороны операционных систем HP-UX и AIX. (Компания HP даже предлагает приобретать средства для зеркального отражения отдельно, хотя в некоторых промышленных конфигурациях она включена изначально.) Если вы хотите иметь систему, основанную на проверке четности, вам понадобится дополнительное аппаратное обеспечение. В то же время система AIX поставляется с интегрированными инструментами для управления аппаратным обеспечением RAID; см раздел Disk Array under Devices справочной системы SMIT.

## Восстановление диска после сбоя

Исследование причин отказов дисков, проведенное компанией Google, свидетельствует о том, что в большинстве промышленных компаний необходимо обеспечивать избыточность хранения информации на диске. При ежегодном уровне отказов, равном 8%, вашей организации потребуется около 150 жестких дисков только для того, чтобы снизить этот показатель до одного сбоя в месяц.

Режимы JBOD и RAID 0 ничем не могут помочь, когда возникает проблема с программным обеспечением, — необходимо выполнять резервное копирование. Другие конфигурации RAID при сбое отмечают вышедшие из строя диски как неисправные. С точки зрения клиентов массивы RAID продолжают нормально функционировать, хотя и на более низком уровне производительности.

Неисправные диски необходимо как можно быстрее заменить новыми, чтобы восстановить избыточность массива. Массив RAID 5 или двухдисковый массив RAID 1 может только смягчить последствия отказа отдельного накопителя. Как только один сбой произошел, массив подвергается опасности второго сбоя.

Процедура замены диска очень простая. Следует заменить неисправный диск другим диском, имеющим такой же или больший объем памяти, а затем сообщить системе RAID о том, что старый диск был заменен новым. После этого последует продолжительный период, в течение которого информация о четности будет переписана на новый, чистый, диск. Часто эту операцию выполняют в ночное время. В течение этого периода

массив остается доступным для клиентов, но производительность, скорее всего, будет очень низкой.

Для того чтобы ограничить время простоя и уменьшить уязвимость массива при втором сбое, большинство систем RAID позволяет назначать один или несколько дисков на роль "горячего резерва". Когда возникает сбой, неисправный диск автоматически заменяется резервным и немедленно начинается процесс повторной синхронизации массива. Если в массиве предусмотрены диски "горячего резерва", то их, разумеется, также следует использовать.

## Недостатки конфигурации RAID 5

RAID 5 — популярная конфигурация, но у нее есть ряд недостатков. Все, что будет сказано ниже, относится и к конфигурации RAID 6, но для простоты мы ограничим обсуждение системой RAID 5.

🕮 Общие рекомендации по резервированию системы см. в главе 10.

Во-первых, что следует особо подчеркнуть, конфигурация RAID 5 не отменяет регулярное автономное резервирование. Она защищает систему лишь от сбоя диска, и все. Она не защищает систему ни от случайного удаления файлов, ни от сбоев контроллера, ложного срабатывания, хакеров и других опасностей.

Во-вторых, конфигурация RAID 5 не отличается высокой производительностью. Она записывает блоки данных на N-1 диск, а блоки четности — на N-й диск.[14] После записи случайного блока должны быть обновлены, по крайней мере, один блок данных и один блок четности данных для указанной последовательности дисков. Более того, система RAID не знает, что именно должен содержать новый блок четности, пока не прочитает старый блок четности и старые данные. Следовательно, каждая запись в случайно выбранное место состоит из четырех операций: двух операций считывания и двух операций записи. (Если реализация обладает развитой логикой, то последовательные записи могут оказаться намного эффективнее.)

И наконец, конфигурация RAID 5 уязвима к повреждениям при определенных обстоятельствах. Ее последовательное обновление данных о четности является более эффективным, чем чтение всей последовательности дисков и повторное вычисление ее четности на основе исходных данных. С другой стороны, это значит, что данные о четности больше нигде не вычисляются и не хранятся. Если будет нарушена синхронизация какого-нибудь блока в последовательности дисков с блоком четности, этот факт в нормальном режиме никак не проявится; операции считывания блоков данных попрежнему будут возвращать правильные данные.

Эта проблема выяснится только при сбое диска. Блок четности, вероятно, будет перезаписан много раз, пока не проявится исходная десинхронизация. Следовательно, реконструированный блок данных на запасном диске будет состоять, по существу, из случайных данных.

Эта разновидность десинхронизации между блоками данных и четности не единственная проблема. Дисковые накопители не являются трансакционными устройствами. Без дополнительного уровня защиты сложно гарантировать, что на разных дисках будут правильно обновлены либо два блока, либо ни одного. Для искажения синхронно-

[14] Данные о четности распределяются между всеми накопителями массива; каждая полоса имеет свою информацию о четности, которая хранится на отдельном диске. Поскольку в массиве нет специального диска для хранения информации о четности, маловероятно, что любой отдельный диск может затормозить работу системы.

сти между блоком данных и блоком четности достаточно, чтобы произошел сбой, отказ электропитания или разрыв связи в неподходящий момент.

Эта проблема известна под названием "дыра записи" RAID 5 (RAID 5 "write hole"). Она является объектом пристального изучения в течение последних пяти лет. Одним из полезных ресурсов, посвященных борьбе с RAID 5 (Battle Against Any Raid Five — BAARF), является сайт `baarf.org`, содержащий ссылки на статьи, посвященные этому вопросу. Вы сами должны решить, действительно ли это важная проблема или искусственно раздута. (Мы склоняемся к тому, что она является важной.)

Авторы файловой системы ZFS в операционной системе Solaris утверждают, что, благодаря переменной длине последовательности дисков в системе ZFS, она защищена от проблемы "дыры записи RAID 5". Кстати, это одна из причин, по которой авторы назвали свою реализацию системы RAID в ZFS именем RAID-Z, а не RAID 5, хотя на практике эти концепции похожи.

Еще одним потенциальным решением является "чистка" ("scrubbing"), которая состоит из поочередной проверки блоков четности во время простоя массива. Многие реализации системы RAID имеют ту или иную функцию очистки.

## Команда mdadm: программное обеспечение RAID в системе Linux

Стандартная реализация программного обеспечения RAID для системы Linux называется **md**, драйвер "нескольких дисков" ("multiple disks" driver). Его внешний интерфейс обеспечивает команда **mdadm**. Драйвер **md** поддерживает все конфигурации RAID, перечисленные выше, а также RAID 4. Более ранняя система под названием **raidtools** больше не используется.

Следующий сценарий выполняет конфигурацию массива RAID 5, состоящего из трех идентичных жестких дисков объемом 500 Гбайт. Несмотря на то что драйвер **md** может использовать в качестве компонентов необработанные диски, мы предпочитаем снабдить каждый диск таблицей разделов, поэтому начинаем с запуска утилиты **gparted**, создающей таблицу разделов MBR на каждом диске (утилита **gparted** предпочтает создавать таблицы разделов в стиле "msdos"), и выделения всего дискового пространства одному разделу типа "неформатированный" (к сожалению, именно так это бывает в действительности). Совершенно не обязательно задавать тип раздела, но это будет полезной информацией для тех, кто станет просматривать таблицу разделов позднее. Кроме того, существует битовый флаг "raid", который можно установить на разделе, хотя с помощью утилиты **gparted** сделать это довольно сложно: вы должны создать раздел, выполнить незаконченные операции, а затем вернуться в новый раздел и отредактировать его флаги.

Следующая команда создает массив RAID 5 из наших трех разделов SCSI.

```
linux$ sudo mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb1
    /dev/sdc1 /dev/sdd1
mdadm: array /dev/md0 started.
```

Виртуальный файл **/proc/mdstat** всегда содержит краткое описание состояния драйвера **md**, а также состояния всех массивов RAID, существующих в системе. Он особенно полезен для анализа файла **/proc/mdstat** после добавления нового диска или замены неисправного. (Рекомендуем запомнить команду **cat /proc/mdstat**.)

```
linux$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
[raid10]
```

```
md0 : active raid5 sdd1[3] sdc1[1] sdb1[0]
      1023404544 blocks level 5, 64k chunk, algorithm 2 [3/2] [UU_]
      [>....................] recovery = 0.1% (672640/511702272)
      finish=75.9min speed=112106K/sec
unused devices: <none>
```

Драйвер **md** не следит за тем, какой блок в массиве был использован, поэтому он должен вручную синхронизировать все блоки четности с их соответствующими блоками данных. Драйвер **md** вызывает операцию "recovery", потому что, по существу, она совпадает с процедурой, использованной для замены неисправного диска. Для больших массивов она может выполняться несколько часов.

В файле **/var/log/messages** также записаны полезные сообщения.

```
md0:RAID5 conf printout:
--- rd:3 wd:2
disk 0, o:1, dev:sdb1
disk 1, o:1, dev:sdc1
disk 2, o:1, dev:sdd1
md: recovery of RAID array md0
md: minimum _guaranteed_ speed: 1000 KB/sec/disk.
md: using max available idle IO bandwidth (but not more than 200000 KB/sec)
for recovery.
md: using 128k window, over a total of 511702272 blocks.
```

Команда первоначального создания также предназначена для "активирования" массива (т.е. для приведения его в состояние, пригодное для использования), но при последующих загрузках может понадобиться активировать массив отдельно, как правило, за рамками сценария загрузки. Системы Red Hat и SUSE имеют простые сценарии загрузки для массивов RAID, а система Ubuntu активирует эти массивы по умолчанию.

Формально, команда **mdadm** не требует наличия файла конфигурации, но если он есть, она его использует (как правило, это файл **/etc/mdadm.conf**). Мы настоятельно рекомендуем использовать файл конфигурации. Он документирует конфигурацию системы RAID стандартным образом, позволяя администратору просмотреть информацию при возникновении проблемы. В качестве альтернативы использованию файла конфигурации можно задавать конфигурацию в командной строке при каждой активации массива.

Команда **mdadm --detail --scan** записывает текущие настройки системы RAID в файл конфигурации. К сожалению, конфигурация, которую она записывает в файл, является неполной. Для получения полной конфигурации необходимо выполнить следующие команды.

```
linux$ sudo sh -c 'echo DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1 >
    /etc/mdadm.conf'
linux$ sudo sh -c 'mdadm --detail --scan >> /etc/mdadm.conf'
linux$ cat /etc/mdadm.conf
DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1
ARRAY /dev/md0 level=raid5 num-devices=3 metadata=00.90 spares=1
UUID=f18070c5:e2b6aa18:e368bf24:bd0fce41
```

Теперь команда **mdadm** может читать этот файл при загрузке или закрытии, чтобы легко управлять массивом. Для того чтобы активировать массив с помощью вновь созданного файла **/etc/mdadm.conf**, можно выполнить следующую команду.

```
$ sudo mdadm -As /dev/md0
```

Для того чтобы остановить массив вручную, можно выполнить такую команду.

```
$ sudo mdadm -S /dev/md0
```

Создав файл **mdadm.conf**, распечатайте его и передайте на сервер. Если что-то произойдет, восстановить компоненты массива RAID бывает сложно.

Команда **mdadm** имеет режим **-monitor**, в котором она выполняется постоянно как демон и сообщает по электронной почте о проблемах, возникших в массиве RAID. Используйте эту возможность! Для того чтобы ее настроить, добавьте строку MAILADDR в свой файл **mdadm.conf**, чтобы указать адресата, которому следует отправлять предупреждения, и настройте демон монитора так, чтобы он запускался во время загрузки. Рассмотренные нами дистрибутивные пакеты операционных систем содержали первичный сценарий, реализующий эту задачу, но имена и процедуры могут немного отличаться.

```
ubuntu$ sudo update-rc.d mdadm enable
suse$ sudo chkconfig -s mdadmd on
redhat$ sudo chkconfig mdmonitor on
```

Что произойдет, если диск действительно выйдет из строя? Попробуем разобраться! Команда **mdadm** предоставляет нам прекрасную возможность сымитировать сбойный диск.

```
$ sudo mdadm /dev/md0 -f /dev/sdc1
mdadm: set /dev/sdc1 faulty in /dev/md0

$ sudo tail /var/log/messages
May 30 16:14:55 harp kernel: raid5: Disk failure on sdc, disabling device.
  Operation continuing on 2 devices
kernel: RAID5 conf printout:
kernel: --- rd:3 wd:2 fd:1
kernel: disk 0, o:1, dev:sdb1
kernel: disk 1, o:0, dev:sdc1
kernel: disk 2, o:1, dev:sdd1
kernel: RAID5 conf printout:
kernel: --- rd:3 wd:2 fd:1
kernel: disk 0, o:1, dev:sdb1
kernel: disk 2, o:1, dev:sdd1

$ cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [linear] [multipath] [raid0] [raid1]
[raid10]
md0 : active raid5 sdb1[0] sdd1[2] sdc1[3](F)
     1023404544 blocks level 5, 64k chunk, algorithm 2 [3/2] [U_U]

unused devices: <none>
```

Поскольку RAID 5 представляет собой избыточную конфигурацию, массив продолжает функционировать в режиме деградации, так что пользователи могут не заметить возникшей проблемы.

Для того чтобы удалить накопители из конфигурации RAID, выполните команду **mdadm -r**.

```
$ sudo mdadm /dev/md0 -r /dev/sdc1
mdadm: hot removed /dev/sdc1
```

Как только диск будет логически удален, вы можете остановить систему и заменить накопитель. Аппаратное обеспечение, допускающее "горячую замену", позволяет производить замену дисков без остановки системы и ее перезагрузки.

Если компонентами массива RAID являются необработанные диски, вы должны заменить их только идентичными дисками. Компоненты, имеющие разделы, можно заме-

нить любым разделом такого же размера, хотя, с точки зрения соответствия полос пропускания, лучше, если аппаратное обеспечение дисков является одинаковым. (Если ваша конфигурация RAID построена на основе разделов, вы должны запустить утилиту разбиения диска и правильно определить разделы, прежде чем добавлять диск в массив.)

В нашем примере сбой всего лишь смоделирован, поэтому мы можем добавить накопитель обратно в массив, не заменяя никакого аппаратного обеспечения.

```
$ sudo mdadm /dev/md0 -a /dev/sdc1
mdadm: hot added /dev/sdc1
```

Драйвер **md** немедленно начнет перестраивать массив. Как обычно, за выполнением этой процедуры вы можете следить с помощью файла **/proc/mdstat**. Перестройка может идти часами, так что учтите этот факт, составляя план по восстановлению системы после сбоя.

# 8.8. Управление логическими томами

Представьте себе ситуацию, в которой вы не знаете, насколько большим должен быть раздел. Через шесть месяцев после создания раздела выясняется, что он слишком большой, а в соседнем разделе недостаточно памяти... Знакомо? Менеджер логического тома позволяет вам динамически перераспределить память между переполненным разделом и разделом, испытывающим нехватку памяти.

Управление логическими томами, по существу, является максимально эффективной и абстрактной версией процедуры разделения диска. Оно группирует отдельные накопители в "группу томов". Блоки в группе томов могут быть затем выделены "логическим томам", которые представлены файлами блочных устройств и функционируют как разделы диска.

Однако логические тома являются более гибкими и мощными, чем разделы дисков.

Перечислим некоторые операции, которые можно выполнять с помощью менеджера тома.

- Перемещение логических томов между несколькими физическими устройствами.
- Увеличение и уменьшение логических томов на лету.
- Копирование при записи "моментальных снимков" логических томов.
- Замена накопителей без прекращения работы.
- Создание зеркал или дисковой последовательности на логических томах.

Компоненты логического тома можно объединить в одно целое разными способами. При конкатенации физические блоки устройств объединяются вместе путем их последовательного расположения друг за другом. При создании последовательности дисков компоненты чередуются так, чтобы соседние виртуальные блоки действительно распределялись по нескольким физическим дискам. При устранении узких мест, связанных с отдельными дисками, создание последовательности дисков часто позволяет расширить полосу пропускания и снизить время простоя.

## Реализации управления логическими томами

Почти все рассмотренные нами операционные системы поддерживали управление логическими томами. За исключением системы Solaris с ее файловой системой ZFS, все системы похожи друг на друга.

Кроме файловой системы ZFS, операционная система Solaris поддерживает предыдущую версию менеджера управления системными дисками под названием Solaris Volume Manager или Solstice DiskSuite. Этот менеджер томов по-прежнему поддерживается, но для инсталляции новых версий требуется использование файловой системы ZFS.

Менеджер логических томов (logical volume manager — LVM) в системе Linux, который называется LVM2, по существу, представляет собой клон менеджера логических томов системы HP-UX, который в свою очередь основан на программном обеспечении Veritas. Команды в этих двух системах практически идентичны, но мы покажем варианты для обеих систем, поскольку вспомогательные команды все же немного отличаются друг от друга. Система AIX основана на тех же абстракциях, но использует другой синтаксис команд. Параллели между этими системами приведены в табл. 8.4.

**Таблица 8.4. Сравнение команд LVM**

| Операция | Linux | HP-UX | AIX |
|---|---|---|---|
| | | Физический том | |
| Создать | **pvcreate** | **pvcreate** | – |
| Инспектировать | **pvdisplay** | **pvdisplay** | **lspv** |
| Изменить | **pvchange** | **pvchange** | **chpv** |
| Проверить | **pvck** | **pvck** | – |
| | | Группа томов | |
| Создать | **vgcreate** | **vgcreate** | **mkvg** |
| Изменить | **vgchange** | **vgchange** | **chvg** |
| Расширить | **vgextend** | **vgextend** | **extendvg** |
| Инспектировать | **vgdisplay** | **vgdisplay** | **lsvg** |
| Проверить | **vgck** | – | – |
| Enable | **vgscan** | **vgscan** | **varyonvg** |
| | | Логический том | |
| Создать | **lvcreate** | **lvcreate** | **mklv** |
| Изменить | **lvchange** | **lvchange** | **chlv** |
| Изменить объем | **lvresize** | **lvextend, lvreduce** | **extendlv** |
| Инспектировать | **lvdisplay** | **lvdisplay** | **lslv** |

Кроме команд, работающих с группами томов и логическими томами, в табл. 8.4 также показана совокупность команд, относящихся к "физическим томам". Физический том — это накопитель, к которому применяется метка LVM; применение этой метки является первым этапом использования устройства с помощью менеджера логических томов. Для применения этой метки системы Linux и HP-UX используют команду **pvcreate**, а в системе AIX эту задачу автоматически выполняет команда **mkvg**. Кроме регистрационной информации, эта метка содержит уникальный идентификатор, позволяющий однозначно распознавать накопитель.

"Физический том"— неудачный термин, потому что физические тома не должны иметь прямого соответствия с физическими устройствами. Они *могут* быть дисками, но они могут быть и разделами дисков, и массивами RAID. Для менеджера логических томов это не имеет значения.

## Управление логическими томами в системе Linux

Менеджером логических томов в системе Linux (LVM2) можно управлять с помощью либо большой группы простых команд (приведенных в табл. 8.4), либо одной команды **lvm** и ее многочисленных подкоманд. Все эти опции имеют одно и то же назначение; фактически отдельные команды действительно прямо обращаются к команде **lvm**, которая проверяет, как ее вызвали, чтобы знать, как реагировать. Хорошим описанием этой системы и ее инструментов является справочная страница **man lvm**.

Процесс настройки менеджера логических томов в системе Linux состоит из нескольких этапов.

- Создание (на самом деле, определение) и инициализация физических томов.
- Добавление физических томов в группу томов.
- Создание логических томов по группе томов.

Команды LVM начинаются с буквы, определяющей уровень абстракции, на котором они действуют: команды с префиксом **pv** манипулируют физическими томами, **vg** — группами томов, а **lv** — логическими дисками. Команды, имеющие префикс **lvm** (например, **lvmchange**), оперируют системой в целом.

В следующем примере мы установили устройство **/dev/md0** RAID 5, созданное ранее, для использования менеджером логических томов и создания логического тома. Поскольку создание последовательности дисков и избыточность обеспечиваются базовой конфигурацией RAID, мы не будем использовать соответствующие возможности менеджера LVM2, хотя они существуют.

```
$ sudo pvcreate /dev/md0
Physical volume "/dev/md0" successfully created
```

Наше физическое устройство теперь готово для добавления в группу томов.

```
$ sudo vgcreate DEMO /dev/md0
Volume group "DEMO" successfully created
```

Несмотря на то что в этом примере используется только одно физическое устройство, мы, разумеется, могли добавить дополнительные накопители. В данном случае было бы странным добавлять еще один массив RAID 5, поскольку частичная избыточность не принесет никакой выгоды. Имя DEMO было выбрано произвольным образом.

Для проверки наших действий используем команду **vgdisplay**.

```
$ sudo vgdisplay DEMO
--- Volume group ---
VG Name               DEMO
System                ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
```

```
VG Size               975.99 GB
PE Size               4.00 MB
Total PE              249854
Alloc PE / Size       0 / 0
Free PE / Size        249854 / 975.99 GB
VG UUID               NtbRLu-RqiQ-3Urt-iQZn-vEvJ-u0Th-FVYKWF
```

Аббревиатура "PE" обозначает физический блок — единицы памяти, на которые разделяется группа томов.

На последних этапах создается логический том в группе DEMO, а затем создается файловая система внутри этого тома. Мы создаем логический том размером 100 Гбайт.

```
$ sudo lvcreate -L 100G -n web1 DEMO
Logical volume "web1" created
```

Большинство интересных возможностей менеджера LVM2 относится к уровню логического тома. В частности, они включают в себя возможности создания последовательности дисков, зеркал и смежного размещения.

Теперь к логическому тому можно обращаться по имени **/dev/DEMO/web1**. Мы обсудим файловые системы в разделе 8.9, а пока кратко опишем создание стандартной файловой системы для демонстрации некоторых особенностей работы менеджера LVM.

```
$ sudo mkfs /dev/DEMO/web1
...
$ sudo mkdir /mnt/web1
$ sudo mount /dev/DEMO/web1 /mnt/web1
```

*Мгновенные копии тома*

Вы можете создать копию при записи любого логического тома LVM2, независимо от того, содержит он файловую систему или нет. Это свойство удобно для создания статического образа файловой системы, предназначенного для резервного копирования, но, в отличие от мгновенных копий системы ZFS, мгновенные снимки LVM2, к сожалению, не очень полезны для контроля версий.

Проблема заключается в том, что логические тома имеют фиксированный размер. Когда создается логический том, пространство на диске выделяется из группы томов. Копирование при записи изначально не потребляет дисковую память, но при модификации блоков менеджер тома должен найти место для хранения как старой, так и новой версии. При создании мгновенных копий это пространство, необходимое для модифицированных блоков, должно резервироваться, причем, как и любой том LVM, выделенная память должна иметь фиксированный размер.

При этом не имеет значения, что модифицируется — оригинальный том или его мгновенная копия (которая по умолчанию должна записываться). В любом случае затраты на дублирование блоков перекладываются на мгновенные копии. Выделение памяти для мгновенных копий можно сократить за счет активности тома, играющего роль источника данных, во время простоя мгновенных копий.

Если объем памяти, выделенный для мгновенной копии, не совпадает с объемом памяти, выделенным для тома, образ которого копируется, возможен выход за пределы памяти, выделенной для мгновенной копии. Намного хуже, чем кажется, поскольку менеджер томов не сможет обеспечить хранение согласованного образа мгновенной копии; потребуется дополнительная память *только лишь для хранения самой копии*. В результате выхода за пределы выделенного пространства менеджер LVM прекращает поддерживать мгновенные копии, и они безвозвратно повреждаются.

Итак, как показывает практика, мгновенные копии должны быть либо краткосрочными, либо такими же большими, как их источники. Эти аргументы свидетельствуют в пользу "многочисленных дешевых виртуальных копий."

Для того чтобы создать том **/dev/DEMO/web1-snap** как мгновенную копию тома **/dev/DEMO/web1**, можно использовать следующую команду.

```
$ sudo lvcreate -L 100G -s -n web1-snap DEMO/web1
```

Обратите внимание на то, что мгновенная копия имеет самостоятельное имя, а ее источник должен быть задан как *группа_томов/том*.

Теоретически том **/mnt/web1** сначала необходимо демонтировать, чтобы гарантировать согласованность файловой системы. На практике система ext4 защищена от повреждения, хотя существует вероятность потерять самые свежие изменения блоков данных. Это идеальный компромисс для мгновенных копий, используемых при резервном копировании.

Для того чтобы проверить состояние мгновенных копий, следует выпонить команду **lvdisplay**. Если она сообщает, что мгновенная копия "не активна", это значит, что она вышла за пределы выделенного пространства и должна быть удалена, поскольку с такой мгновенной копией практически ничего сделать нельзя.

### Резервирование файловых систем

Переполнение файловых систем происходит чаще, чем сбои дисков, и одно из преимуществ логических томов заключается в том, что манипулировать ими гораздо легче, чем жесткими разделами. Мы сталкивались с разными ситуациями: от хранения личных файлов MP3 на сервере до департаментов, компьютеры которых переполнены почтовым мусором.

Менеджер логических дисков ничего не знает о содержимом своих томов, но изменять размеры необходимо на уровне как тома, так и файловой системы. Порядок зависит от конкретной операции. При уменьшении размера следует работать с файловой системой, а при увеличении — с томом. Это правило запоминать не стоит: просто следуйте здравому смыслу.

Предположим, что в нашем примере том **/mnt/web1** увеличился больше, чем предполагалось, и ему нужно дополнительно 10 Гбайт дисковой памяти. Сначала проверим группу томов, чтобы убедиться, что дополнительное место существует.

```
$ sudo vgdisplay DEMO
--- Volume group ---
VG Name                               DEMO
System ID
Format                                lvm2
Metadata Areas                        1
Metadata Sequence No                  18
VG Access                             read/write
VG Status                             resizable
MAX LV                                0
Cur LV                                2
Open LV                               1
Max PV                                0
Cur PV                                1
Act PV                                1
VG Size                               975.99 GB
PE Size                               4.00 MB
Total PE                              249854
```

```
  Alloc PE / Size          51200 / 200.00 GB
  Free PE / Size           198654 / 775.99 GB
  VG UUID                  NtbRLu-RqiQ-3Urt-iQZn-vEvJ-u0Th-FVYKWF
```

Итак, есть много свободной дисковой памяти, поэтому мы должны демонтировать файловую систему и использовать команду **lvresize**, чтобы добавить место на логическом томе.

```
$ sudo umount /mnt/web1
$ sudo lvchange -an DEMO/web1
$ sudo lvresize -L +10G DEMO/web1
$ sudo lvchange -ay DEMO/web1
Extending logical volume web1 to 110.00 GB
Logical volume web1 successfully resized
```

Команды **lvchange** необходимы для того, чтобы деактивировать том, изменить его объем, а потом вновь активировать. Эта часть необходима только потому, что существует копия тома **web1** из предыдущего примера. После изменения объема копия "увидит" дополнительные 10 Гбайт, но поскольку файловая система содержит только 100 Мбайт, копию можно по-прежнему использовать.

Теперь мы можем изменить объем файловой системы с помощью команды **resize2fs**. (Цифра 2 взята из имени оригинальной файловой системы ext2, но команда поддерживает все версии этой файловой системы.) Поскольку команда **resize2fs** может определять объем новой файловой системы по тому, нам не обязательно указывать новый объем явным образом. Это необходимо только при уменьшении размера файловой системы.

```
$ sudo resize2fs /dev/DEMO/web1
resize2fs 1.41.9 (22-Aug-2009)
Please run 'e2fsck -f /dev/DEMO/web1' first.
```

Ой! Команда **resize2fs** вынуждает дважды проверять согласованность файловой системы до изменения объема.

```
$ sudo e2fsck -f /dev/DEMO/web1
e2fsck 1.41.9 (22-Aug-2009)
Pass 1: Checking inodes, blocks, and sizes
...
/dev/DEMO/web1: 6432/6553600 files (0.1% non-contiguous), 473045/26214400
   blocks
$ sudo resize2fs /dev/DEMO/web1
resize2fs 1.41.9 (22-Aug-2009)
Resizing the filesystem on /dev/DEMO/web1 to 28835840 (4k) blocks.
The filesystem on /dev/DEMO/web1 is now 28835840 blocks long.
```

Ну вот! Проверка информации, которая выводится командой **df**, снова показывает изменения.

```
$ sudo mount /dev/DEMO/web1 /mnt/web1
$ df -h /mnt/web1
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/DEMO-web  1109G  188M  103G   1% /mnt/web1
```

## Управление логическими томами в системе HP-UX

В системе HP-UX 10.20 существует полноценный менеджер логических томов. Это прекрасное дополнение, особенно если учесть, что в прошлом система HP-UX вообще не поддерживала концепции разбиения дисков. Менеджер томов называется LVM, так же как в системе Linux, хотя версия системы HP-UX является совершенно самостоятельной. (На самом деле это программное обеспечение компании Veritas...)

В качестве простого примера посмотрим, как конфигурировать жесткий диск объемом 75 Гбайт с помощью менеджера LVM. По сравнению с ранее рассмотренным примером, относящимся к системе Linux, новая процедура выглядит знакомой. Есть несколько незначительных отличий, но в целом процесс тот же.

Команда **pvcreate** идентифицирует физические тома.

```
$ sudo pvcreate /dev/rdisk/disk4
Creating "/etc/lvmtab_p".
Physical volume "/dev/rdisk/disk4" has been successfully created.
```

Если диск будет использоваться как загрузочный, добавьте в команду **pvcreate** опцию **-B**, чтобы зарезервировать пространство для загрузочного блока, а затем запустите команду **mkboot**, чтобы инсталлировать его.

Определив диск как физический том, добавьте его в новую группу томов с помощью команды **vgcreate**. Для групп томов существует два формата метаданных — версии 1.0 и 2.0. Номер версии задается с помощью опции **-V** при создании группы томов; версия 1.0 задается по умолчанию. Версия 2.0 предусматривает более высокие пределы для объема, но не может использоваться для загрузочных накопителей или для замены томов. Даже метаданные версии 1.0 имеют щедрые лимиты, поэтому она используется в большинстве случаев. Точные пределы можно увидеть с помощью команды **lvmadm**. Для примера посмотрим на пределы, предусмотренные для версии 1.0.

```
$ sudo lvmadm -t -V 1.0
--- LVM Limits ---
VG Version 1.0
Max VG Size (Tbytes) 510
Max LV Size (Tbytes) 16
Max PV Size (Tbytes) 2
Max VGs 256
Max LVs 255
Max PVs 255
Max Mirrors 2
Max Stripes 255
Max Stripe Size (Kbytes) 32768
Max LXs per LV 65535
Max PXs per PV 65535
Max Extent Size (Mbytes) 256
```

Добавить дополнительные диски в группу томов можно с помощью команды **vgextend**, но в данном примере группа томов содержит только один диск.

```
$ sudo vgcreate vg01 /dev/disk/disk4
Increased the number of physical extents per physical volume to 17501.
Volume group "/dev/vg01" has been successfully created.
Volume Group configuration for /dev/vg01 has been saved in
    /etc/lvmconf/vg01.con
```

Добавив диски в подходящую группу томов, пул дискового пространства, относящийся к данной группе томов, можно снова разбить на логические тома. Новый логический том создает команда **lvcreate**. Объем тома в мегабайтах задается опцией **-L**, а в терминах логических диапазонов (как правило, 4 двоичных мегабайта) — с помощью флага **-l**. Размеры, заданные с помощью двоичных мегабайтов, округляются до ближайшего целого числа, кратного размеру логического диапазона.

Для доступа к объему свободного пространства, оставшегося в группе томов, необходимо выполнить команду **vgdisplay** *имя_группы_томов* как корневую. Результаты содержат размер данного логического диапазона и количество свободных логических диапазонов.

```
$ sudo lvcreate -L 25000 -n web1 vg01
Logical volume "/dev/vg01/web1" has been successfully created with character
   device "/dev/vg01/rweb1".
Logical volume "/dev/vg01/web1" has been successfully extended.
Volume Group configuration for /dev/vg01 has been saved in
   /etc/lvmconf/vg01.conf
```

Приведенная выше команда создает логический том объемом 25 Гбайт с именем **web1**. Создав логические тома, можно верифицировать их, выполнив команду **vgdisplay -v /dev/***имя_группы_томов* и проверив их размеры, и убедиться, что они установлены правильно.

В большинстве сценариев после этого можно создать файловую систему **/dev/vg01/web1** и смонтировать ее во время загрузки. Детали изложены в разделе 8.9.

Еще одним распространенным способом создания логического тома является выполнение команды **lvcreate**, которая создает том нулевого объема, и команды **lvextend**, чтобы добавить в этот том дополнительную память. Таким образом можно точно указать, какие физические тома в группе томов должны образовывать логический том. Если память выделяется с помощью команды **lvcreate** (как в приведенном выше примере), она просто использует свободные логические диапазоны из любых доступных физических томов в группе томов. Это удобно во многих ситуациях.

Как и в системе Linux, создание последовательности дисков (striping), которая в системе HP-UX называется "распределенным выделением памяти", а также создание зеркал происходят на уровне логического тома. Эту операцию можно выполнить при создании логического тома с помощью команды **lvcreate** или позднее с помощью команды **lvchange**. В противоположность системе Linux, менеджер логических томов не позволяет делать мгновенные копии. Однако временные копии доступны как свойство файловой системы VxFS в рамках операционной системы HP-UX.

Если вы планируете использовать логический том как загрузочный, менять накопители или хранить снимки памяти ядра системы, должны указать непрерывную область памяти и отключить некорректные блоки с помощью опций **-C** и **-r** команды **lvcreate**.[15]

```
# lvcreate -C y -r n -L 1500 -n root vg01
Logical volume "/dev/vg01/root" has been successfully created with character
    device "/dev/vg01/rroot".
```

[15] Для того чтобы удовлетворить требования системы HP-UX, необходимо сначала создать том подкачки, так чтобы первые два двоичных гигабайта физического диска и загрузочный том были расположены на первом логическом томе. Создание корневого тома объемом 1,5 Гбайт и тома подкачки объемом 500 Мбайт в приведенном примере выполнялось именно из-за упомянутых ограничений. Корневой раздел может иметь больше памяти, чем указано здесь, но загрузочный и корневой тома должны быть разными. Подробности можно найти на справочной странице для команды **lvlnboot**.

```
Logical volume "/dev/vg01/root" has been successfully extended.
Volume Group configuration for /dev/vg01 has been saved in
    /etc/lvmconf/vg01.conf
# lvcreate -C y -r n -L 500 -n swap vg01
Logical volume "/dev/vg01/swap" has been successfully created with character
    device "/dev/vg01/rswap".
...
```

Затем необходимо выполнить команду **lvlnboot**, чтобы уведомить систему о новом корневом томе и томе подкачки. Более подробную информацию о процедурах для томов загрузки, подкачки и создания снимков памяти можно найти на справочной странице **lvlnboot**.

## Управление логическими томами в системе AIX


Набор команд менеджера логических томов в системе AIX отличается от наборов команд менеджеров томов в системах Linux и HP-UX, но его базовая архитектура и подход являются аналогичными. Существует одна тонкость: система AIX вызывает объекты, которые известны как "диапазоны" (т.е. единицы памяти в группе томов). Поскольку эти сущности обычно называются разделами, которых на самом деле в системе AIX не существует, никаких противоречий не возникает. Однако пользователи других систем должны учитывать сложившуюся терминологию.

По отношению к физическим томам, группам томов и логическим томам терминология системы AIX является стандартной. Интерфейс SMIT для управления логическими томами предусматривает практически полный набор функций, но, кроме них, можно использовать команды, перечисленные в табл. 8.4. Следующие четыре команды создают группу томов с именем **webvg**, логический том с именем **web1** и файловую систему JFS2 в томе **web1**. Затем файловая система монтируется в томе **/mnt/web1**.

```
$ sudo mkvg -y webvg hdisk1
webvg
$ sudo crfs -v jfs2 -g webvg -m /mnt/web1 -a size=25G
File system created successfully.
26213396 kilobytes total disk space.
New File System size is 52428800
$ sudo mkdir /mnt/web1
$ sudo mount /mnt/web1
```

Система AIX не требует использования меток дисков для включения их в физические тома. Команды **mkvg** и **extendvg** автоматически наносят метки на диски в процессе индукции. Отметим, что команда **mkvg** получает имя накопителя, а не путь к дисковому накопителю.

Логический том и файловую систему можно создать на разных этапах (с помощью команд **mklv** и **mkfs** соответственно), но команда **crfs** выполняет обе эти задачи, а также обновляет файл **/etc/filesystems**. Точное имя логического тома, содержащего файловую систему, создается в рамках сценария **crfs**, но его можно определить, проверив файл **/etc/filesystems** или выполнив команду **mount**. (С другой стороны, если возникнут проблемы, будет трудно расшифровать файловые системы на томах, имеющих обобщенные имена.)

Если команда **mklv** выполняется непосредственно, можно не только задать имя накопителя по своему выбору, но и указать разные параметры менеджера томов, например

конфигурации для создания последовательности дисков или зеркал. Мгновенные копии реализуются посредством файловой системы JFS2, а не менеджера томов.

## 8.9. Файловые системы

Даже после разбиения жесткого диска на разделы или логические тома он еще не готов хранить файлы. Для этого необходимо реализовать все абстракции и функции, описанные в главе 6, в терминах блоков. Они реализуются кодом, который называется файловой системой и требует дополнительных затрат ресурсов и данных.

Первым стандартом файловой системы, получившим широкое распространение среди систем UNIX, стала система Berkeley Fast File System, реализованная Мак-Кьюзиком и его соавторами (McKusick et al.) в 1980-х годах. С некоторыми поправками она в конце концов стала известна как UNIX File System (UFS) и заложила основы для разработки других файловых систем, включая семейство файловых систем ext в системе Linux, UFS в системе Solaris и JFS компании IBM.

В ранних версиях операционных систем реализация файловой системы была встроена в ядро, но вскоре стало ясно, что поддержка многочисленных типов файловых систем является важной целью. Разработчики систем UNIX создали хорошо определенный интерфейс ядра, позволявший одновременно активировать несколько типов файловых систем. Кроме того, базовое аппаратное обеспечение было абстрагировано с помощью интерфейса файловой системы, так что файловые системы видели примерно такой же интерфейс для накопителей, что и другие программы системы UNIX, имеющие доступ к дискам через файлы устройств в каталоге **/dev**.

Изначально поддержка нескольких типов файловых систем объяснялась необходимостью поддерживать системы NFS и файловые системы для сменных носителей. Однако как только шлюз был открыт, началась эра поисков; многие группы стали работать над улучшением файловых систем. Некоторые из этих файловых систем были предназначены для конкретных операционных систем, а другие (например, ReiserFS) не были связаны ни с какими определенными операционными системами.

Если бы у вас было право выбора файловых систем, разве должны вы исследовать разные варианты и выбирать "лучший"? Конечно, нет, если только вы не работаете с данными, предназначенными для очень специфичного приложения. Практически во всех ситуациях лучше работать с решениями, принятыми в системе по умолчанию. Именно это, вероятно, предполагается разработчиками документации и административных инструментов для операционных систем.

Не подвергаются сомнению только несколько функциональных возможностей.

- Высокая производительность.
- Устойчивость к отказам и перебоям электропитания без повреждения файловой системы.
- Способность работать с дисками и файловыми системами, которые являются достаточно большими для того, чтобы удовлетворить потребности пользователей.

К счастью, решения, принятые по умолчанию в современных файловых системах, уже соответствуют этим требованиям. Любое улучшение, которое можно ожидать от изменения файловых систем, будет незначительным и, в лучшем случае, зависящим от контекста.

В следующих разделах мы обсудим файловые системы, реализованные по умолчанию в системах Linux, HP-UX и AIX. Файловая система ZFS, используемая в операционной

системе Solaris, управляется особым образом и заслуживает описания в отдельном разделе (см. раздел 8.10).

## Файловые системы Linux: семейство ext

"Вторая расширенная файловая система" ("second extended filesystem"), получившая название ext2, долгое время была основным стандартом в системе Linux. Она была разработана и реализована Реми Кардом (Remy Card), Теодором Тс'о (Theodore Ts'o) и Стивеном Твиди (Stephen Tweedie). Помимо того, что код файловой системы ext2 был написан специально для системы Linux, с функциональной точки зрения он похож на файловую систему Berkeley Fast File System.

Файловая система ext3 расширяла возможности журналирования (journaling), которые имелись в существовавшем коде ext2, и представляла собой концептуально простую модификацию, резко повышавшую надежность. Еще более интересным является то, что расширения в файловой системе ext3 были реализованы без изменения фундаментальной структуры файловой системы ext2. Фактически вы можете монтировать файловую систему ext3 так, будто это файловая система ext2, просто без дополнительных возможностей журналирования.

Файловая система ext3 резервирует область памяти на диске для журнала. Журнал размещается на диске так, будто он представляет собой обычный файл в корневом разделе файловой системы и не является отдельным структурным компонентом.

При выполнении операции с файловой системой требуемые модификации сначала записываются в журнал. Когда обновление журнала завершено, в него вносится "запись о фиксации" ("commit record"), чтобы отметить конец вводимых данных. Только затем происходит модификация обычной файловой системы. Если в процессе обновления происходит сбой, файловая система использует журнал для реконструкции полностью согласованной файловой системы.[16]

Журналирование уменьшает время, необходимое для проверки согласованности файловой системы (см. раздел, посвященный команде **fsck**), примерно на одну секунду в расчете на одну файловую систему. После сбоя аппаратного обеспечения определенного типа состояние файловой системы ext3 можно практически мгновенно восстановить.

Файловая система ext4 представляет собой определенное улучшение своих предшественников. Она расширяет пределы максимально допустимой памяти, увеличивает производительность некоторых операций и позволяет использовать для выделения памяти не отдельные блоки диска, а "логические диапазоны" ("extents"). Дисковый формат вполне совместим с файловыми системами ext2 и ext3 и допускает их монтирование в виде файловой системы ext4. Более того, если не используются логические диапазоны, то файловые системы ext4 могут монтироваться так, будто они являются файловыми системами ext3.

При работе в операционной системе Linux с ядром 2.6.28 рекомендуется использовать файловую систему ext4, а не предыдущие версии.[17] В системах Ubuntu и SUSE она используется по умолчанию, а система Red Hat сохраняет файловую систему ext3.

---

[16] В большинстве случаев в журнал заносятся только метаданные. Реальные данные хранятся непосредственно в файловой системе. Однако это можно изменить с помощью опции **data** команды **mount**. Подробности можно найти на справочной странице, посвященной команде **mount**.

[17] Некоторые утверждают, что требование использовать файловую систему ext4 в ядре 2.6.28 является преждевременным. Тем не менее текущая версия этой файловой системы является надежной.

В существующую файловую систему ext2 легко добавить журнал, тем самым расширив ее возможности до систем ext3 или ext4 (благодаря обратной совместимости, отличия между ними трудно уловимы). Для этого достаточно просто выполнить команду **tune2fs** с опцией **-j**.

```
# tune2fs -j /dev/hda4
```

После этого, возможно, потребуется модификация некоторой записи в файле **/etc/fstab**, чтобы заменить строки `ext4` строками `ext2` (информация о файле **fstab** приводится далее).

## Файловые системы HP-UX: VxFS и HFS

Файловая система VxFS является основной в операционной системе HP-UX. Она основана на файловой системе, изначально разработанной компанией Veritas Software, которая в настоящее время стала частью компании Symantec. Поскольку она содержит журнал, ее иногда называют JFS (Journaled File System — журналируемая файловая система). Тем не менее не следует путать ее с файловой системой JFS2 в операционной системе AIX; это разные файловые системы.

Файловая система VxFS — уникальное явление среди файловых систем. Она отличается тем, что поддерживает кластеризацию, т.е. одновременную модификацию со стороны многих независимых компьютеров. Этот вид операции связан с определенным снижением производительности, поскольку файловая система выполняет дополнительные действия, чтобы обеспечить согласованность кеша среди компьютеров. По умолчанию функциональные возможности кластеризации отключены; для того чтобы их включить, необходимо использовать опцию **-o cluster** команды **mount**

Ранее основной файловой системой в системе HP-UX считалась HFS, которая была основана на файловой системе UNIX File System. В настоящее время она устарела, хотя и поддерживается до сих пор.

## Файловая система JFS2 в операционной системе AIX

Еще одной файловой системой, ведущей свое происхождение от системы Berkeley Fast File System, является JFS2. Буква J означает "журналируемая" ("journaled"), хотя система JFS2 имеет и другие особенности. Например, логические разделы, динамическое размещение индексных дескрипторов (inodes) и использование структуры дерева B+ для хранения элементов каталога.

Кроме того, файловая система JFS2 интересна тем, что она доступна по лицензии GNU General Public License и работает в операционной системе Linux тоже.

## Терминология файловой системы

Благодаря своему родству с файловыми системами UFS, многие файловые системы используют общую описательную терминологию. Реализации базовых объектов часто изменяются, но термины по-прежнему широко употребляются системными администраторами как обозначения фундаментальных понятий. Например, индексные дескрипторы ("inodes") — это записи фиксированной длины в таблице, каждый элемент которой хранит информацию об отдельном файле. Ранее они заносились в эту таблицу в момент создания файловой системы, но в настоящее время некоторые файловые системы созда-

ют их динамически при необходимости. В любом случае индексный дескриптор обычно имеет идентификационный номер, который можно увидеть с помощью команды `ls -i`.

Индексные дескрипторы указывают на элементы каталога. При создании жесткой ссылки на существующий файл создается новый элемент каталога, но не новый индексный дескриптор.

В системах с заранее размещенными индексными дескрипторами необходимо заранее решить, сколько их следует создать. Поскольку невозможно предсказать, сколько их понадобится в будущем, команды создания файловой системы используют эмпирическую формулу, основанную на объеме тома и среднем размере файлов. Если вы планируете хранить огромное количество маленьких файлов, это число следует увеличить.

Суперблок (superblock) — это запись, описывающая характеристики файловой системы. Она содержит информацию о длине блока диска, размере и местоположении таблиц индексных дескрипторов, карту блоков и информацию о их использовании, размер групп блоков и некоторые другие важные параметры файловой системы. Поскольку повреждение суперблока может привести к потере очень важной информации, его несколько копий хранятся в разных местах.

Для повышения производительности файловые системы кешируют блоки диска. Кешировать можно все блоки диска, включая суперблоки, блоки индексных дескрипторов и информацию о каталоге. Кеши обычно не допускают "сквозной записи" ("write-through"), поэтому может возникнуть задержка между моментом, когда приложение считает, что блок записан, и моментом, в который блок действительно записывается на диск. Приложения могут требовать от файла более предсказуемого поведения, но в этом случае пропускная способность снизится.

Системный вызов **sync** направляет модифицированные блоки в место их постоянного хранения на диске, стараясь моментально обеспечить полную согласованность дисковой файловой системы. Это периодическое сохранение минимизирует потерю данных, которая может произойти при сбое из-за многочисленных несохраненных блоков. Файловые системы могут самостоятельно выполнять синхронизацию своего расписания или предоставить это операционной системе. Современные файловые системы имеют механизм журналирования, минимизирующий или исключающий возможность повреждения их структуры при сбое, поэтому частота синхронизации должна зависеть от того, сколько блоков могут оказаться потерянными при сбое.

В файловой системе карта дисковых блоков — это таблица содержащихся в ней свободных блоков. При записи новых файлов система изучает эту карту, чтобы найти эффективную схему хранения. Записи об использовании блоков содержат важную информацию об уже используемых блоках. В файловых системах, поддерживающих логические разделы, эта информация может быть значительно сложнее, чем обычная битовая карта, которая применялась в старых файловых системах.

## Полиморфизм файловых систем

Файловая система — это программное обеспечение, состоящее из многочисленных компонентов. Одна ее часть находится в ядре (или же в пользовательском пространстве системы Linux; наберите в поисковой машине google аббревиатуру FUSE) и реализует основные операции, связанные с трансляцией прикладного интерфейса стандартной файловой системы в операции чтения и записи блоков диска. Другие части состоят из пользовательских команд для форматирования новых томов, проверки целостности файловой системы и выполнения других специальных операций, связанных с форматированием.

Ранее стандартные пользовательские команды знали о файловой системе, которую использовала операционная система, и просто реализовали требуемые функциональные возможности. Команда **mkfs** создавала новые файловые системы, команда **fsck** устраняла проблемы, а команда **mount**, в основном, просто вызывала системные функции. Современные файловые системы являются более модульными, поэтому эти команды теперь вызывают реализации утилит, связанных с файловой системой.

Точный способ реализации зависит от операционной системы. Например, операционная система Linux помещает отдельные команды с именами **mkfs**.*имя_файловой_системы*, **fsck**.*имя_файловой_системы* и так далее в обычные каталоги, предназначенные для системных команд. (Эти команды можно также выполнить непосредственно, но такие ситуации возникают редко.) Система AIX имеет переключатель **/etc/vfs**, в который записываются метаданные для файловых систем (не следует его путать с переключателем **/etc/vfstab** в системе Solaris, который эквивалентен файлам **fstab** или **filesystems** в других системах; впрочем, в системе ZFS он не нужен).

## Команда mkfs: форматирование файловых систем

Для создания новой файловой системы следует выполнить следующую команду.

**mkfs** [**-T** *тип_файловой_системы* ] [**-o** *опции*] *накопитель*

По умолчанию *тип_файловой_системы* может быть жестко закодирован в упаковщике или задан в файле **/etc/default/fs**. Допустимые *опции* зависят от конкретной файловой системы, но использовать их приходится редко. Система Linux использует флаг **-t**, а не **-T**, игнорирует флаг **-o** и не имеет файлов устройств, непосредственно соответствующих накопителям. Система AIX использует флаг **-V**, а не **-T**.


Команда **crfs** в системе AIX может выделять новый логический том, создавать файловую систему на нем и обновлять файл **/etc/filesystems** за один шаг.

Кроме того, можно либо включить возможность создания мгновенных копий в файловых системах, которые допускают эту операцию (JFS2 и VxFS), либо разместить журнал файловой системы на отдельном диске. Вторая возможность в соответствующих ситуациях может обеспечить резкий прирост производительности.

## Команда fsck: проверка и исправление файловых систем

Из-за буферизации блока и того факта, что накопители на самом деле не являются трансакционными устройствами, структуры данных на файловых системах могут стать рассогласованными. Если эти проблемы не удается быстро устранить, то это превратится в лавину.

Вначале это повреждение исправлялось с помощью вызова команды **fsck** ("filesystem consistency check"; читается как "FS check" или "fisk"), которая тщательно проверяла все структуры данных и просматривала дерево выделения для каждого файла. Она использовала набор эвристических правил, описывающих возможный внешний вид файловой системы после повреждения, в разные моменты времени в процессе ее обновления.

Оригинальная схема **fsck** работала удивительно хорошо, но поскольку она предусматривала чтение всех данных с диска, для больших дисков процедура могла занимать несколько часов. Одно из первых решений этой задачи было основано на концепции "чистого бита файловой системы", который устанавливался в суперблоке, когда система демонтировалась некорректно. При повторной загрузке системы можно было уви-

деть чистый бит и таким образом узнать, что проверку с помощью команды **fsck** можно пропустить.

В настоящее время журналы файловой системы позволяют команде **fsck** сосредоточить свое внимание на том, что произошло во время сбоя. Команда **fsck** может просто восстановить предыдущее согласованное состояние файловой системы.

Как правило, диски обрабатываются командой **fsck** автоматически во время загрузки, если они перечислены в системных файлах **/etc/fstab**, **/etc/vfstab** или **/etc/filesystems**. Файлы **fstab** и **vfstab** содержат поля, заполненные последовательностью команд **fsck**, которые обычно используются для параллельного распределения проверок файловой системы. Однако в настоящее время команда **fsck** выполняется так быстро, что для нее имеет значение только один фактор — чтобы корневая файловая система проверялась первой.

Команду **fsck** можно запустить вручную, чтобы выполнить глубокую проверку, которая была характерна для ее первых версий, но для этого потребуется время.

Семейство файловых систем ext в операционной системе Linux может быть настроено так, чтобы выполнять повторную проверку после демонтирования определенное количество раз или по истечении определенного периода времени, даже если все демонтированные файловые системы были "чистыми". Это полезная процедура, и в большинстве ситуаций приемлемым является значение, задаваемое по умолчанию (около 20 монтирований). Однако если файловые системы монтируются часто, как это происходит в настольных рабочих станциях, то даже эта частота выполнения команды **fsck** может стать чрезмерной. Для того чтобы увеличить интервал времени до 50 монтирований, следует использовать команду **tune2fs**.

```
$ sudo /sbin/tune2fs -c 50 /dev/sda3
tune2fs 1.41.9 (22-Aug-2009)
Setting maximal mount count to 50
```

Если файловая система повреждена и команда **fsck** не может восстановить ее автоматически, то *не следует с ней экспериментировать*, пока не создан надежный резерв. Лучшая политика страхования — применить ко всему диску команду **dd** и создать резервный файл или диск.

Большинство файловых систем создает каталог **lost+found** в корневом разделе каждой файловой системы, где команда **fsck** может хранить файлы, для которых невозможно определить родительский каталог. Каталог **lost+found** имеет дополнительное пространство, чтобы команда **fsck** могла хранить "осиротевшие" файлы, не создавая дополнительных каталогов в нестабильной файловой системе. Не удаляйте этот каталог.[18]

Поскольку имя, присвоенное файлу, записывается только в родительском каталоге, имена "осиротевших" файлов недоступны, и файлы, хранящиеся в каталоге **lost+found**, именуются по названиям их индексных дескрипторов. Однако таблица индексных дескрипторов содержит идентификаторы UID владельца файла, что позволяет легко возвращать файл его владельцу.

## Монтирование файловой системы

Файловую систему необходимо смонтировать до того, как она станет видимой для процессов. Точкой монтирования для файловой системы может быть любой каталог, но

[18] В некоторых системах для восстановления этого каталога после удаления используется команда **mklost+found**.

файлы и подкаталоги, которые расположены ниже этой точки в иерархии, останутся недоступными, пока файловая система не будет смонтирована в этой точке. Более подробную информацию можно найти в разделе 6.2.

После инсталляции нового диска необходимо смонтировать новые файловые системы вручную, чтобы убедиться, что все работает правильно. Например, команда

```
$ sudo mount /dev/sda1 /mnt/temp
```

монтирует файловую систему в разделе, представленном файлом устройства **/dev/sd1a** (имена устройств зависят от операционной системы), в каталоге **/mnt**, который является традиционным для временных точек монтирования.

Размер файловой системы можно проверить с помощью команды **df**. В приведенном ниже примере флаг **-h** системы Linux используется для выдачи результатов в понятном для человека виде. К сожалению, большинство системных значений, заданных по умолчанию для команды **df**, относятся к бесполезным сущностям, таким как "дисковые блоки", но есть флаг, заставляющий команду **df** выдавать конкретную информацию, например двоичные кило- или гигабайты.

```
$ df -h /mnt/web1
Filesystem Size Used Available Use% Mounted on
/dev/mapper-DEMO-web1 109G 188M 103G 1% /mnt/web1
```

## Настройка автоматического монтирования

Обычно требуется конфигурировать операционную систему так, чтобы монтировать локальные файловые системы на этапе загрузки. Файл конфигурации в каталоге **/etc** содержит имена устройств и точки монтирования всех системных дисков (среди прочей информации). В большинстве систем этот файл называется **/etc/fstab** (сокращение от "filesystem table"), но в системах Solaris и AIX он был реструктурирован и переименован: **/etc/vfstab** в системе Solaris и **/etc/filesystems** в системе AIX. Ссылаясь на эти три файла, мы используем в книге обобщенный термин "каталог файловой системы".

По умолчанию файловые системы ZFS монтируют себя автоматически и не требуют записей из файла **vfstab**. Однако это поведение можно изменить, задав свойства файловой системы ZFS. Области подкачки и точки монтирования нефайловых систем должны по-прежнему указываться в файле **vfstab**.

Команды **mount**, **umount**, **swapon** и **fsck** читают каталог файловой системы, поэтому желательно, чтобы представленные в этом каталоге данные были правильными и полными. Команды **mount** и **umount** используют этот каталог, чтобы выяснить, что мы хотим, указывая в командной строке только имя раздела или точку монтирования. Например, при конфигурации команды **fstab** в системе Linux, приведенной далее, команда

```
$ sudo mount /media/cdrom0
```

эквивалентна следующей команде.

```
$ sudo mount -t udf -o user,noauto,exec,utf8 /dev/scd0 /media/cdrom0
```

Команда **mount -a** монтирует все регулярные файловые системы, перечисленные в каталоге файловой системы; обычно эта команда выполняется в рамках сценария запуска на этапе загрузки.[19] Флаги **-t**, **-F** и **-v** (**-t** — в системе Linux, **-F** — в системах Solaris и HP-UX и **-v** — в системе AIX) в сочетании с аргументом *тип_файловой_си-*

[19] Опция `noauto` команды ***mount*** исключает указанную файловую систему из процесса автоматического монтирования с помощью команды ***mount -a***.

*стемы* ограничивают операции над файловой системой определенного типа. Например, команда

```
$ sudo mount -at ext4
```

монтирует все локальные файловые системы ext4. Команда **mount** читает файл **fstab** последовательно.

Следовательно, файловые системы, смонтированные под другими файловыми системами, должны следовать за своими родительскими разделами, указанными в файле **fstab**. Например, строка для файла **/var/log** должна следовать за строкой **/var**, если **/var** — отдельная файловая система.

Команда **umount**, предназначенная для демонтирования файловых систем, имеет аналогичный синтаксис. Файловую систему, которую некий процесс использует в качестве своего текущего каталога или которая содержит открытый файл, демонтировать нельзя. Существуют команды, позволяющие идентифицировать процессы, препятствующие попыткам выполнить команду **umount**; см. раздел 6.2.

Файл **fstab** в системе HP-UX встречается во всех операционных системах. Рассмотрим его записи о системе, имеющей только одну группу томов.

```
# Device        Mount point  Type   Options    Seq

/dev/vg00/lvol3 /            vxfs   delaylog   0 1
/dev/vg00/lvol1 /stand       vxfs   tranflush  0 1
/dev/vg00/lvol4 /tmp         vxfs   delaylog   0 2
/dev/vg00/lvol5 /home        vxfs   delaylog   0 2
/dev/vg00/lvol6 /opt         vxfs   delaylog   0 2
/dev/vg00/lvol7 /usr         vxfs   delaylog   0 2
/dev/vg00/lvol8 /var         vxfs   delaylog   0 2
```

Каждая строка содержит шесть полей, разделенных пробелами, и описывает отдельную файловую систему. Для повышения читабельности поля обычно выравниваются, но в принципе это не требуется.

🕮 Более подробно система NSF описана в главе 18.

Первое поле содержит имя устройства. Файл **fstab** может содержать точки монтирования из удаленных систем. В этом случае первое поле содержит путь NFS. Обозначение *server:/export* означает каталог */export* на машине с именем *server*.

Второе поле задает точку монтирования, а третье — тип файловой системы. Точное название типа, идентифицирующего локальные файловые системы, зависит от конкретной машины.

Четвертое поле задает опции команды **mount**, которые должны применяться по умолчанию. Существует множество возможностей; опции, общие для всех типов файловых систем, можно найти на справочной странице для команды **mount**. Отдельные файловые системы обычно имеют собственные опции. Все приведенные выше опции относятся к файловой системе VxFS. Например, опция `delaylog` снижает уровень читабельности, но повышает скорость выполнения команды. Более подробную информацию об этой и других опциях монтирования файловой системы VxFS можно найти на справочной странице **mount_vxfs**.

Пятое и шестое поля являются рудиментарными. Предположительно они задавали "частоту создания копии" и параметр управления параллельным выполнением команд **fsck**. Ни одна из этих возможностей в современных системах не актуальна.

Ниже приводятся примеры файла **fstab** из системы Ubuntu. Общий формат тот же, но системы семейства Linux часто имеют разные особенности.

```
proc /proc proc defaults 0 0
UUID=a8e3...8f8a / ext4 errors=remount-ro 0 1
UUID=13e9...b8d2 none swap sw 0 0
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto,exec,utf8 0 0
/dev/scd1 /media/cdrom1 udf,iso9660 user,noauto,exec,utf8 0 0
/dev/fd0 /media/floppy0 auto rw,user,noauto,exec,utf8 0 0
```

Первая строка относится к файловой системе **/proc**, которая фактически представлена драйвером ядра и не имеет вспомогательного запоминающего устройства. Устройство `proc`, указанное в первом столбце, просто занимает место.

Вторая и третья строки для идентификации томов используют идентификаторы разделов (идентификаторы UUID, которые мы сократили, чтобы не снизить читабельность текста), а не имена устройств.

Эта альтернатива полезна для систем семейства Linux, поскольку имена устройств для разделов диска являются нестабильными; добавление или удаление диска может вызвать изменение имен всех дисков (например, с **/dev/sdb1** на **/dev/sdc1**). Идентификатор UUID связан только с содержанием раздела. Это позволяет отследить раздел, если он не был скрыт. Отметим, что это соглашение работает также и для раздела подкачки, и для корневого раздела.

Последние три строки конфигурируют поддержку дисководов CD-ROM и гибких дисков. Опция `noauto` предотвращает попытки системы монтировать эти устройства во время загрузки. (Если носители не вставлены, то попытки монтирования завершатся сбоем и затормозят процесс загрузки.) Опция `user` назначает владельцем всех файлов, записанных на сменные носители, пользователя, который их монтировал.

В системах семейства Solaris формат файла **/etc/vfstab** был немного изменен, и некоторые поля находятся на других местах, по сравнению с системами Linux и HP-UX.

Однако данные по-прежнему выводятся в виде таблицы и легко читаются без дешифровки. Отличительные особенности формата файла **vfstab** заключаются в том, что он имеет отдельный столбец "device to **fsck**" и отдельный столбец "mount at boot".

Файл **/etc/filesystems** в системе AIX организован как ряд списков свойств, напоминающих YAML или JSON, хотя его формат может немного варьироваться. Рассмотрим пример конфигурации для одной файловой системы.

```
/opt:
dev = /dev/hd10opt
vfs = jfs2
log = /dev/hd8
mount = true
check = true
vol = /opt
free = false
```

Этот формат позволяет ассоциировать с каждой файловой системой произвольные свойства, так что параметры ее типа можно легко записать в каталоге **filesystems**. Система AIX автоматически поддерживает этот файл, когда пользователь выполняет дисковые операции посредством интерфейса SMIT, но можно легко редактировать этот файл непосредственно.

## Монтирование USB-накопителя

Гибкие диски окончательно ушли в прошлое. Их место заняли удобные, быстрые и забавные USB-накопители. Применение этих устройств весьма разнообразно: например, персональные флешки, цифровые видеокамеры, устройства iPod и крупные внешние диски. Большинство из них поддерживается системами семейства UNIX в качестве накопителей данных.

В прошлом для управления USB-накопителями приходилось прибегать к специальным трюкам. Однако в настоящее время, когда операционные системы реализуют управление динамическими устройствами в качестве фундаментального требования, USB-накопители представляют собой просто еще один тип накопителей, которые появляются и исчезают без предупреждения.

С точки зрения управления, с USB-накопителями связаны следующие проблемы.

- Ядро должно распознавать устройство и назначать ему файл устройства.
- Необходимо выяснять, какие назначения были сделаны.

Первый этап обычно выполняется автоматически, но в системах есть команды (например, команда **cfgmgr** в системе AIX), которые можно выполнять непосредственно. Как только файл устройства назначен, можно использовать стандартные процедуры, описанные в разделе 8.5.

Дополнительную информацию об управлении динамическими устройствами можно найти в главе 13.

## Включение подкачки

Как правило, в качестве области подкачки используются разделы или логические тома, а не структурированные файловые системы. Вместо использования файловой системы для слежения за содержимым области подкачки, ядро поддерживает собственное упрощенное отображение блоков памяти в блоки области подкачки.

В некоторых системах можно выполнять подкачку с помощью файла из раздела файловой системы. В старых операционных системах эта конфигурация может работать медленнее, чем при использовании специального раздела, но в редких случаях это бывает очень удобным. В любом случае менеджеры логических томов исключают большинство причин, по которым вам может понадобиться использовать файл подкачки, а не том подкачки.

🕮 Информацию о разделении области подкачки можно найти в разделе 29.3.

Чем больше область подкачки, которую вы используете, тем больше виртуальной памяти могут занимать ваши процессы. Максимальная производительность виртуальной памяти достигается, когда область подкачки разделена между несколькими накопителями. Разумеется, лучше всего вообще не использовать подкачку; если вам необходимо оптимизировать производительность подкачки, лучше подумайте об увеличении оперативной памяти.

В системах семейства Linux области подкачки должны инициализироваться с помощью команды **mkswap**, получающей в качестве аргумента имя накопителя, на котором располагается том подкачки.

Вы можете самостоятельно назначить накопитель в качестве инструмента подкачки с помощью команды **swapon** *накопитель* в большинстве систем или команды **swap -a** *накопитель* в системе Solaris. Однако в большинстве случаев предпочтительнее, чтобы

эта функция автоматически выполнялась на этапе загрузки. За исключением системы AIX, существует возможность перечислять области подкачки в обычном каталоге файловой системы (**fstab** или **vfstab**), указав swap в качестве их типа. В системе AIX этот список хранится в отдельном файле с именем **/etc/swapspaces**.

Для того чтобы просмотреть текущую конфигурацию подкачки, следует выполнить команды **swapon -s** в системах семейства Linux, **swap -s** — в системах Solaris и AIX и **swapinfo** — в системе HP-UX.

В системах семейства AIX с помощью команды **mkps** можно создать логический том, предназначенный для подкачки, добавить его в файл **/etc/swapspaces** и приступить к его использованию. Эта команда вызывается из интерфейса SMIT.

# 8.10. Файловая система ZFS: все проблемы решены

Система ZFS появилась в 2005 году как компонент операционной системы OpenSolaris и быстро оказалась составной частью системы Solaris 10 и различных дистрибутивных пакетов, основанных на лицензии BSD. В 2008 году она уже могла использоваться как корневая файловая система и с тех пор стала основным выбором для операционной системы Solaris.

Несмотря на то что система ZFS обычно называется файловой, на самом деле она представляет собой глобальный механизм управления хранением данных, который включает в себя функции менеджера логических томов и контроллера RAID. Кроме того, он заново определяет многие аспекты управления хранением данных, делая их проще, легче и более логичными. Несмотря на то что текущая версия системы ZFS имеет несколько ограничений, большинство из них относится к категории "еще не реализовано", а не "невозможно по архитектурным причинам". Преимущества интегрированного подхода, реализованного в системе ZFS, очевидны. Даже если вы еще не знакомы с системой ZFS, мы уверены, что она вам понравится. Мы почти не сомневаемся, что эта система будет широко распространена в течение следующих десяти лет. Остается неясным лишь, как долго придется ждать реализации аналогичных функций в других системах. Несмотря на то что файловая система ZFS является открытым программным обеспечением, условия ее текущей лицензии запрещают включение в ядро Linux.

Проект файловой системы Btrfs компании Oracle ("B-tree file system", официальное название "butter FS," хотя это и вызывает ассоциации с выражением "butter face"[20]) пытается воспроизвести многие из достижений системы ZFS на платформе Linux. Эта файловая система уже включена в текущие ядра систем семейства Linux для предварительных испытаний. Пользователи систем Ubuntu и SUSE могут экспериментировать с ней, инсталлировав пакеты **btrfs-tools** или **btrfsprogs** соответственно. Однако файловая система Btrfs еще не готова к промышленному выпуску, и поскольку компания Oracle стала частью компании Sun, будущее систем Btrfs и ZFS остается неопределенным.

## Архитектура ZFS

На рис. 8.4 схематично изображены основные объекты системы ZFS и их взаимосвязи.

Пул системы ZFS аналогичен группе томов в других системах с механизмами управления логическими томами. Каждый пул состоит из виртуальных устройств, представ-

[20] "Butter face" на сленге означает "девушка с красивой фигурой, но некрасивым лицом". — Примеч. ред.

ляющих собой накопители (диски, разделы, устройства SAN и так далее), группы зеркал или массивов RAID.

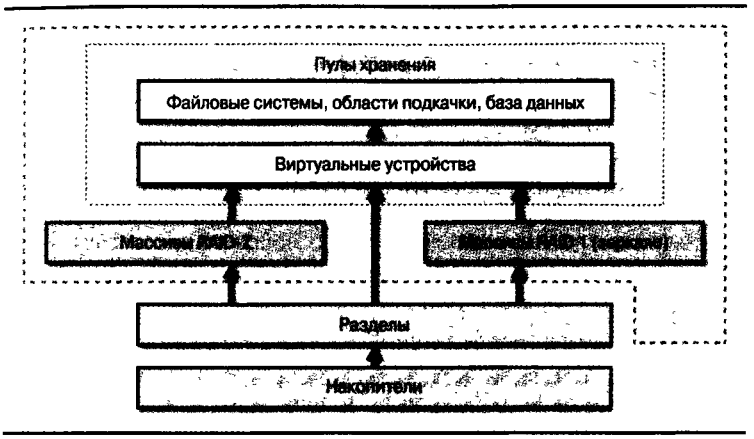



*Рис. 8.4. Архитектура ZFS*

Массив ZFS RAID по существу похож на массив RAID 5 тем, что он использует одно или несколько устройств четности для обеспечения избыточности массива. Однако в системе ZFS эта схема называется RAID-Z и использует последовательности дисков переменных объемов, чтобы исключить появление "дыры записи RAID 5". Все операции записи в пул хранения распределяются по виртуальным устройствам пула, поэтому пул, содержащий только отдельные накопители, по существу, представляет собой схему RAID 0, хотя накопители в этой конфигурации не обязаны иметь одинаковые объемы.

К сожалению, текущая версия массива ZFS RAID имеет недостатки. Например, в ней невозможно ни добавить новые устройства в массив после того, как он был определен, ни удалить устройство ZFS. Как и в большинстве реализаций массива RAID, накопители в нем должны иметь одинаковые объемы; вы можете заставить систему ZFS принять накопители переменных объемов, но общий размер массива будет определяться по наименьшему объему накопителя. Для того чтобы эффективно использовать диски переменных объемов в сочетании со схемой ZFS RAID, необходимо заранее разбить диски на разделы и определить оставшиеся области в качестве отдельных устройств.

Несмотря на то что существует возможность передать неразделенные диски под управление файловой системы ZFS, она скрытно записывает на них таблицу разделов, похожую на GPT, и выделяет все дисковое пространство на каждом диске их первым разделам.

Большая часть работы по настройке системы ZFS и управлению ею выполняется посредством двух команд: **`zpool`** и **`zfs`**. Команда **`zpool`** используется для создания пулов хранения и управления ими, а команда **`zfs`** предназначена для создания сущностей, возникших из пула, в основном файловых систем и томов, используемых в качестве области подкачки и баз данных, и управления ими.

## Пример: добавление диска в системе Solaris

Прежде чем погрузиться в детали системы ZFS, рассмотрим небольшой пример.

Допустим, что мы добавили новый диск в систему Solaris и он получил имя **`/dev/dsk/c8d1`**. (Для того чтобы определить требуемое устройство, следует выполнить коман-

ду **sudo format**. Команда **format** выведет меню системных дисков, из которого можно выбрать нужный диск, а затем нужно нажать комбинацию клавиш <Ctrl+C>.)

На первом этапе необходимо пометить диск и добавить его в новый пул хранения.

```
solaris$ sudo zpool create demo c8d1
```

На втором этапе ..., ну, на самом деле второго этапа нет. Система ZFS помечает диск, создает пул "demo", корневую файловую систему внутри этого пула и монтирует эту файловую систему как **/demo** за один этап. Файловая система будет демонтирована автоматически при загрузке системы.

```
solaris$ ls -a /demo
...
```

Пример был бы еще более впечатляющим, если бы мы могли просто добавить новый диск в существующий пул хранения корневого диска, который по умолчанию называется "rpool". (Эта команда могла бы выглядеть так: **sudo zpool add rpool c8d1**.) К сожалению, корневой пул может содержать только одно виртуальное устройство. Тем не менее остальные пулы допускают безболезненное расширение.

## Файловые системы и свойства

С помощью системы ZFS очень удобно автоматически создавать файловую систему в новом пуле хранения, потому что такие файловые системы не занимают какой-то конкретный объем памяти. Все файловые системы, существующие в пуле, могут занимать доступное пулу пространство.

В отличие от традиционных файловых систем, которые не зависят друг от друга, файловые системы ZFS являются иерархическими. Существует несколько способов взаимодействия файловых систем со своими родительскими и дочерними файловыми системами. Новая файловая система создается с помощью команды **zfs create**.

```
solaris$ sudo zfs create demo/new_fs
solaris$ zfs list -r demo
NAME          USED  AVAIL  REFER  MOUNTPOINT
demo          100K  488G   21K    /demo
demo/new_fs    19K  488G   19K    /demo/new_fs
```

Файл **-r** в команде **zfs list** означает, что она рекурсивно применяется ко всем дочерним файловым системам. Большинство остальных подкоманд **zfs** также понимает флаг **-r**. Еще более полезно то, что система ZFS автоматически монтирует новую файловую систему сразу после ее создания.

Для того чтобы имитировать традиционные файловые системы, имеющие фиксированный размер, к свойствам файловой системы можно добавить параметры **reservation** (объем дисковой памяти, зарезервированный для пула хранения и предназначенный для использования файловой системой) и **quota**. Эта настройка является ключевой в управлении системой ZFS и представляет собой определенную смену парадигмы для администраторов, работающих с другими системами. Для примера зададим оба значения равными 1 Гбайт.

```
solaris$ sudo zfs set reservation=1g demo/new_fs
solaris$ sudo zfs set quota=1g demo/new_fs
solaris$ zfs list -r demo
NAME          USED   AVAIL  REFER   MOUNTPOINT
demo          1.00G  487G     21K   /demo
demo/new_fs     19K  1024M    19K   /demo/new_fs
```

Новое значение параметра **quota** отражается в столбце AVAIL для файловой системы **/demo/new_fs**. Значение параметра **reservation** показано в столбце USED для файловой системы **/demo**. Это объясняется тем, что резерв для файловых систем, дочерних по отношению к системе **/demo**, уже учтен в ее размере.[21]

Изменения этих значений отражаются только на учетных записях. Единственное изменение, которое действительно касается пула хранения, — это изменение одного или двух блоков для записи новых параметров. Ни один процесс не запускается, чтобы отформатировать один гигабайт дискового пространства, зарезервированного для файловой системы **/demo/new_fs**. Большинство операций ZFS, включая создание нового пула хранения и новых файловых систем, выполняется так же легко.

Используя иерархическую систему управления дисковым пространством, можно легко группировать несколько файловых систем, чтобы гарантировать, что их общий объем не превысит заданного порога; нет необходимости задавать этот порог для каждой файловой системы отдельно.

Параметры **quota** и **reservation** следует задавать так, чтобы они правильно имитировали традиционную файловую систему фиксированного размера.[22] Сам по себе параметр **reservation** просто гарантирует, что файловая система будет иметь достаточно дисковой памяти, чтобы достичь *по крайней мере* указанного размера. Параметр **quota** ограничивает размер файловой системы *без гарантии*, что она будет иметь этот объем памяти для своего увеличения; другой объем может занять все свободное пространство пула, не оставив места для расширения файловой системы **/demo/new_fs**.

С другой стороны, в реальной жизни существует очень мало причин для такой настройки файловой системы. Эти свойства следует использовать только для демонстрации возможностей системы учета памяти в ZFS и чтобы подчеркнуть, что система ZFS при необходимости может быть совместимой с традиционной моделью.

## Наследование свойств

Многие свойства естественным образом наследуются дочерними файловыми системами. Например, если мы хотим смонтировать корень пула файловой системы **demo** в каталоге **/opt/demo**, а не **/demo**, можно просто задать соответствующий параметр **mountpoint**.

```
solaris$ sudo zfs set mountpoint=/opt/demo demo
solaris$ zfs list -r demo
NAME          USED   AVAIL   REFER   MOUNTPOINT
demo          1.00G   487G     21K   /optf
demo/new_fs     19K   1024M    19K   /opt/demo/new_fs
solaris$ ls /opt/demo
new_fs
```

[21] Столбец REFER содержит объем данных в активной копии каждой файловой системы. Файловые системы ***/demo*** и ***/demo/new_fs*** имеют близкие значения значений REFER, поскольку обе они пустые, а не потому что между ними существует какая-то связь.

[22] Параметры ***reservation*** и ***quota*** учитывают весь объем дисковой памяти, занимаемый файловой системой, включая пространство, занятое мгновенными копиями. Если вы хотите ограничить размер только активной копии файловой системы, следует использовать параметры ***refreservation*** и ***refquota***. Префикс ***ref*** означает "объем данных, на которые ссылается файловая система". Именно это значение показано в столбце REFER в результатах работы команды ***zfs list***.

Параметр **mountpoint** автоматически демонтирует файловые системы, а изменение точки монтирования влияет на дочерние файловые системы предсказуемым и очевидным образом. Обычные правила, касающиеся файловой системы, тем не менее, остаются в силе (см. раздел 6.2).

Для того чтобы увидеть действительное значение конкретного свойства, используется команда **zfs get**, а команда **zfs get all** выводит список значений всех параметров. Столбец SOURCE содержит информацию, объясняющую, почему каждое свойство имеет конкретное значение: слово `local` означает, что свойство было задано явно, а косая черта — что свойство предназначено только для чтения. Если значение свойства унаследовано от родительской файловой системы, то в столбце SOURCE будут указаны детали того наследования.

```
solaris$ zfs get all demo/new_fs
solaris$ zfs get all demo/new_fs
NAME         PROPERTY          VALUE                  SOURCE
demo/new_fs  type              filesystem             -
demo/new_fs  creation          Wed Mar 17 17:57 2010  -
demo/new_fs  used              19K                    -
demo/new_fs  available         1024M                  -
demo/new_fs  referenced        19K                    -
demo/new_fs  compressratio     1.00x                  -
demo/new_fs  mounted           yes                    -
demo/new_fs  quota             1G                     local
demo/new_fs  reservation       1G                     local
demo/new_fs  mountpoint        /opt/demo/new_fs       inherited from demo
… <остальные строки, всего около 40>
```

Внимательные читатели могут заметить, что свойства `available` и `referenced` подозрительно похожи на столбцы AVAIL и REFER, которые выводятся на печать командой **zfs list**. Фактически команда **zfs list** — это просто другой способ вывести на печать свойства файловой системы. Если бы выше мы привели полный список результатов, полученных с помощью команды **zfs get**, то среди них были бы и использованные свойства. Для того чтобы указать, какие именно свойства нас интересуют, их можно указать с помощью опции **-o** команды **zfs list**.

Поскольку нет никакого смысла задавать явно значения свойств `used` и других свойств, связанных с объемом, эти свойства предназначаются только для чтения. Если конкретные правила вычисления значения свойства `used` вас по каким-то причинам не устраивают, возможно, следует использовать другие свойства, такие как `usedbychildren` и `usedbysnapshots`, чтобы увидеть, как расходуется дисковая память. Полный список свойств можно найти в справочном руководстве администратора системы ZFS.

Для собственного использования и создания локальных сценариев можно задавать дополнительные, нестандартные свойства файловых систем. Процесс их задания ничем не отличается от задания стандартных свойств. Имена пользовательских свойств должны содержать двоеточие, чтобы отличать их от стандартных свойств.

## Одна файловая система на пользователя

Поскольку файловые системы не расходуют дисковую память и не требуют время на свое создание, их оптимальное количество скорее ближе к "много", чем к "мало". Если рабочие каталоги пользователей находятся в пуле хранения системы ZFS, рекомендуется создать отдельный рабочий каталог для отдельной файловой системы. Существует несколько причин для такого соглашения.

- Если необходимо задать квоты использования диска, то естественно в качестве единицы измерения применять рабочие каталоги. Квоты можно установить как для файловых систем для индивидуальных пользователей, так и для файловых систем, содержащих всех пользователей.
- Мгновенные копии соответствуют отдельным файловым системам. Если каждый рабочий каталог пользователя представляет собой отдельную файловую систему, то пользователь может получить доступ к старым мгновенным копиям с помощью каталога **~/.zfs**.[23] Одно это сэкономит администратору много времени, поскольку это значит, что пользователи могут самостоятельно обслуживать свои потребности в восстановлении файлов.
- Система ZFS позволяет делегировать разрешение выполнять разные операции, такие как поиск мгновенных копий и откат системы в предыдущее состояние. При желании можно передать пользователям контроль над этими операциями, которые выполняются в их собственных каталогах. В этой книге мы не будем описывать детали механизма управления разрешениями в системе ZFS; их можно найти в справочнике *ZFS Administration Guide*.

## Мгновенные копии и клоны

Система ZFS организована по принципу копирования при записи. Вместо перезаписывания блоков на месте, система ZFS выделяет новые блоки и обновляет указатели. Этот подход делает систему ZFS устойчивой к повреждениям, поскольку при сбоях или отказе электропитания и операции никогда не останутся выполненными наполовину. Корневой блок либо будет обновлен, либо нет; в любом случае файловая система останется согласованной (хотя некоторые недавно внесенные изменения могут быть отменены).

Как и менеджер логических томов, система ZFS обеспечивает копирование при записи на пользовательском уровне, разрешая создавать мгновенные копии. Однако есть важное отличие: мгновенные копии системы ZFS реализуются для файловых систем, а не для логических томов, поэтому они имеют произвольную глубину детализации. Система Solaris с огромным успехом использует эту функциональную возможность в приложении Time Slider для интерфейса GNOME. Аналогично приложению Time Machine в системе Mac OS, приложение Time Slider — это комбинация назначенных заданий, создающих мгновенные копии и управляющих ими через регулярные интервалы времени, и пользовательского интерфейса, который позволяет найти старые версии файлов.

Мгновенную копию можно создать с помощью командной строки **zfs snapshot**. Например, следующая последовательность команд иллюстрирует создание мгновенной копии, ее использование с помощью каталога **zfs/snapshot** и возвращение файловой системы в предыдущее состояние.

```
solaris$ sudo touch /opt/demo/new_fs/now_you_see_me
solaris$ ls /opt/demo/new_fs
now_you_see_me
solaris$ sudo zfs snapshot demo/new_fs@snap1
solaris$ sudo rm /opt/demo/new_fs/now_you_see_me
solaris$ ls /opt/demo/new_fs
```

[23] Этот каталог по умолчанию является скрытым; он не появляется среди результатов работы команды **ls -a**. Его можно сделать видимым с помощью команды **zfs set snapdir=visible** файловая_система.

```
solaris$ ls /opt/demo/new_fs/.zfs/snapshot/snap1
now_you_see_me
solaris$ sudo zfs rollback demo/new_fs@snap1
solaris$ ls /opt/demo/new_fs
now_you_see_me
```

Каждой мгновенной копии в момент ее создания можно присвоить имя. Полная спецификация мгновенной копии обычно записывается в виде *файловая_система@мгновенная_копия*.

Для рекурсивного создания мгновенных копий используется команда **zfs snapshot -r**. Ее эффект эквивалентен выполнению команды **zfs snapshot** над каждым объектом по отдельности: каждый подкомпонент получает собственную мгновенную копию. Все мгновенные копии имеют одинаковые имена, но с логической точки зрения они отличаются друг от друга.

Мгновенные копии в системе ZFS предназначены только для чтения, и хотя они имеют определенные свойства, все же не являются файловыми системами в подлинном смысле. Однако вы можете инициировать мгновенную копию как полноценную и допускающую запись файловую систему, "клонировав" ее.

```
solaris$ sudo zfs clone demo/new_fs@snap1 demo/subclone
solaris$ ls /opt/demo/subclone
now_you_see_me
solaris$ sudo touch /opt/demo/subclone/and_me_too
solaris$ ls /opt/demo/subclone
and_me_too now_you_see_me
```

Мгновенная копия, которая стала оригиналом для клона, остается нетронутой и предназначается только для чтения.

Однако новая файловая система (в нашем примере, **demo/subclone**) сохраняет связь как с мгновенной копией, так и с файловой системой, на которой она основана, и ни одну из них нельзя удалить, пока существует клон.

Операция клонирования применяется относительно редко, но это единственный способ создать новую ветвь на дереве эволюции файловой системы. Операция **zfs rollback**, продемонстрированная выше, может только восстановить файловую систему в состоянии, соответствующем самой последней по времени мгновенной копии, поэтому при ее использовании вы будете вынуждены постоянно удалять (**zfs destroy**) все мгновенные копии, сделанные за время, прошедшее после создания мгновенной копии, к которой хотите вернуться. Клонирование позволяет вам вернуться в прошлое без потери изменений, внесенных недавно.

Предположим, вы обнаружили дыру в системе безопасности, возникшую в течение прошедшей недели. Для обеспечения безопасности вы захотите вернуть файловую систему в состояние, в котором она находилась неделю назад, чтобы быть уверенным, что она не содержит лазеек, оставленных хакерами. В то же время вы не хотите потерять результаты работы за последнюю неделю или данные для аналитического отчета. Решением этой проблемы является клонирование мгновенной копии, сделанной неделю назад, в виде новой файловой системы, применение команды **zfs rename** к старой файловой системе и команды **zfs rename** — к клону, заменяющему исходную файловую систему.

Полезно также применить к клону команду **zfs promote**; эта операция инвертирует отношения между клоном и оригиналом файловой системы. После активации основная файловая система имеет доступ ко всем старым мгновенным копиям файловой системы, а старая файловая система становится "клонированным" ответвлением.

## Неразмеченные логические тома

Области подкачки и неразмеченные области для хранения создаются с помощью команды **zfs create**, точно так же как создаются файловые системы. Аргумент **-V** *размер* заставляет команду **zfs** обрабатывать новый объект как неразмеченный логический том, а не файловую систему. Параметр *размер* может использовать общую единицу измерения, например **128m**.

Поскольку том не содержит файловую систему, он не монтируется; вместо этого он демонстрируется в каталогах **/dev/zvol/dsk** и **/dev/zvol/rdsk**, и ссылаться на него можно так, будто он представляет собой жесткий диск или раздел. Система ZFS отражает иерархическую структуру пула хранения в этих каталогах, поэтому команда **sudo zfc create -V 128m demo/swap** создает том подкачки размером 128 Мбайт, размещенный в каталоге **/dev/zvol/dsk/demo/swap**.

Мгновенные копии неразмеченных томов можно создавать точно так же, как мгновенные копии файловых систем, но поскольку в данном случае нет иерархии файловой системы, в которой размещается каталог **.zfs/snapshot**, мгновенные копии оказываются в том же самом каталоге, что их исходные тома. Клоны функционируют точно так, как ожидается.

По умолчанию неразмеченные тома получают резерв дисковой памяти, равный его указанному объему. Можно уменьшить резерв или отказаться от него вообще, но в этом случае попытки записи на том могут вызвать ошибку "out of space". Клиенты неразмеченных томов могут не справиться с такими ошибками.

## Распределение файловой системы между NFS, CIFS и SCSI

Система ZFS не только переопределяет много аспектов управления традиционными файловыми системами, но и изменяет способ распределения файловой системы по сети. В частности, можно задать свойство **sharenfs** или **sharesmb** файловой системы равным **on** и сделать ее доступной с помощью протокола NFS или встроенного сервера CIFS операционной системы Solaris. Более подробно протокол NFS описан в главе 18, а сервер CIFS — в разделе 30.6.

Если оставить эти свойства равными **off**, то это не значит, что файловая система не станет общедоступной; просто пользователю придется использовать специальные инструменты для экспорта, например утилиты **sharemgr**, **share** и **unshare**, а не функциональные возможности системы ZFS. Свойства **sharenfs** и **sharesmb** могут принимать не только значения **on** и **off**. Если задать другое значение, то это будет означать, что вы хотите обеспечить совместное использование файловой системы, и это значение будет передаваться с помощью команды **zfs share** в виде аргументов командной строки.

Аналогично свойство **shareiscsi=on**, примененное, к неразмеченному тому, делает его доступным в рамках протокола iSCSI. Более подробно о протоколе рассказывается в разделе 8.10.

По умолчанию все свойства **share*** являются наследуемыми. Если вы делаете файловую систему **/home** совместно используемой в рамках протокола NFS, например, то автоматически становятся доступными все рабочие каталоги, расположенные в ней, даже если они были определены как самостоятельные файловые системы. Разумеется, это поведение можно отменить, задав явно параметр **sharenfs=no** для каждой вложенной файловой системы.

Для доступа к спискам управления доступом система ZFS использует стандарт NFSv4. Нюансы этого стандарта обсуждаются в главе 6. Подводя итог, скажем лишь, что

система ZFS обеспечивает прекрасную поддержку списков управления доступом для клиентов как системы Windows, так и протокола NFS.

## Управление пулом хранения

Рассмотрев свойства системы ZFS, относящиеся к файловой системе и уровню "блок-клиент", исследуем ее пулы хранения.

До сих пор мы использовали пул с именем "demo", который создали на отдельном диске. Команда **zpool list** выводит на печать следующие результаты.

```
solaris$ zpool list
NAME    SIZE   USED    AVAIL   CAP   HEALTH   ALTROOT
demo    496G   240K    496G    0%    ONLINE   -
rpool   748G   23.78G  724G    3%    ONLINE   -
```

Пул с именем **rpool** содержит загружаемую корневую файловую систему. На загружаемые пулы в настоящее время наложено несколько ограничений: они могут содержать только одно виртуальное устройство, и это устройство должно быть либо зеркальным массивом, либо отдельным диском; оно не может быть массивом RAID. Если устройство является диском, то оно не может иметь таблицы разделов GPT.

Команда **zpool status** добавляет несколько деталей о виртуальных устройствах, из которых состоит пул хранения, и сообщает об их текущем состоянии.

```
solaris$ zpool status demo
pool:    demo
state:   ONLINE
scrub:   none requested
config:
   NAME    STATE   READ   WRITE   CKSUM
   demo    ONLINE   0       0       0
   c8d1    ONLINE   0       0       0
```

Оставим пул **demo** и создадим нечто более сложное.

Мы присоединили к нашей системе пять накопителей SCSI емкостью 500 Гбайт. Сначала создадим пул с именем "monster", содержащий три из этих накопителя, в конфигурации RAID-Z с одним разрядом контроля четности (single-parity configuration).

```
solaris$ sudo zpool destroy demo
solaris$ sudo zpool create monster raidz1 c9t0d0 c9t1d0 c9t2d0
solaris$ zfs list monster
NAME      USED    AVAIL   REFER   MOUNTPOINT
monster   91.2K   981G    25.3K   /monster
```

Система ZFS также распознает схемы **raidz2** и **raidz3** для конфигурации с двумя и тремя разрядами контроля четности. Минимальное количество дисков всегда на единицу больше, чем количество устройств контроля четности. В данном случае один из трех накопителей предназначен для контроля четности, поэтому для файловой системы доступен примерно один терабайт памяти.

Для иллюстрации добавим оставшиеся два накопителя, сконфигурированных как зеркало.

```
solaris$ sudo zpool add monster mirror c9t3d0 c9t4d0
invalid vdev specification
use '-f' to override the following errors:
mismatched replication level: pool uses raidz and new vdev is mirror
solaris$ sudo zpool add -f monster mirror c9t3d0 c9t4d0
```

Команда **zpool** сначала "спотыкается" на этой конфигурации, потому что эти два виртуальных устройства имеют разные схемы избыточности. Данная конфигурация работает хорошо, поскольку оба виртуальных устройства обладают определенной избыточностью. В реальных условиях не следует смешивать виртуальные устройства, обладающие и не обладающие избыточностью, поскольку невозможно предсказать, какие блоки на каких устройствах будут находиться; частичная избыточность здесь бесполезна.

```
solaris$ zpool status monster
  pool: monster
  state: ONLINE
 scrub: none requested
config:

   NAME      STATE   READ  WRITE  CKSUM
   monster   ONLINE    0      0      0
   raidz1    ONLINE    0      0      0
   c9t0d0    ONLINE    0      0      0
   c9t1d0    ONLINE    0      0      0
   c9t2d0    ONLINE    0      0      0
   mirror    ONLINE    0      0      0
   c9t3d0    ONLINE    0      0      0
   c9t4d0    ONLINE    0      0      0
```

Система ZFS распределяет записи между всеми виртуальными устройствами пула. Как показано в этом примере, если виртуальные устройства имеют одинаковые объемы, это не обязательно.[24] Однако компоненты внутри избыточной группы должны иметь одинаковые размеры. Если это условие не выполняется, то каждый компонент будет иметь минимальный размер. При использовании нескольких простых дисков в пуле хранения важную роль играет конфигурация RAID 0.

Дополнительные виртуальные устройства можно добавить в пул в любое время. Однако существующие данные не будут перераспределены, чтобы не потерять преимущества параллелизма. К сожалению, в настоящее время невозможно добавлять дополнительные устройства в существующий массив RAID или зеркало.

Система ZFS имеет особенно хорошую реализацию кеширования чтения, которое обеспечивает эффективное использование накопителей SSD. Для того чтобы установить эту конфигурацию, достаточно просто добавить накопители SSDs в пул хранения как виртуальные устройства типа **cache**. Система кеширования использует алгоритм адаптивной замены, разработанный компанией IBM, который эффективнее обычного алгоритма кеширования LRU (least recently used — наиболее давно использовавшийся). Ей известна частота обращения к блокам и момент времени, когда они использовались в последний раз, поэтому чтение больших файлов не предполагает разрушения кеша.

Горячее резервирование реализуется с помощью виртуальных устройств типа **spare**. Один и тот же диск можно добавлять в несколько пулов хранения; диск из пула, который первый даст сбой, объявляется резервным.

## 8.11. Сеть хранения данных

Существует несколько способов присоединить ресурсы хранения к сети. В главе 18 описывается традиционный протокол NFS для системы UNIX, применяемый для со-

[24] В данном примере диски имеют одинаковые объемы, а виртуальные устройства нет (1 Тбайт против 500 Гбайт).

вместного использования файлов. Система Windows имеет аналогичный протокол, известный как CIFS или SMB. Основной реализацией протокола CIFS для систем UNIX и Linux является протокол Samba; подробно описан в разделе 30.6.

Протоколы NFS и CIFS — примеры систем для "сетевого хранилища" (NAS). Они относятся к высокоуровневым протоколам, а их основные операции сводятся к таким действиям, как "открыть файл X и послать мне первые 4 двоичных килобайта данных" или "настроить список управления доступом для файла Y, как указано в запросе". Эти протоколы хорошо работают в файловых системах, где многие клиенты хотят читать или записывать одновременно.

Сеть хранения данных (SAN) — это низкоуровневая система для абстрагирования операций хранения, при котором сетевое хранилище напоминает локальный жесткий диск. Операции SAN состоят, в основном, из инструкций чтения или записи конкретных дисковых "блоков" (хотя, разумеется, адресация этих блоков сервером определенным образом делается виртуальной). Если клиент хочет использовать сеть SAN для хранения файловой системы, он должен обеспечить собственную реализацию файловой системы. С другой стороны, тома SAN также можно использовать для хранения областей подкачки или других данных, не обязательно имеющих структуру файловой системы.

За исключением файловой системы VxFS в операционной системе HP, основные файловые системы не предназначены для работы с многочисленными клиентами, не знающими о существовании друг друга (по крайней мере, не на уровне неразмеченных блоков диска).[25] Следовательно, хранилище SAN обычно не рассматривают как способ совместного использования файлов. На самом деле оно представляет собой средство для замены локальных жестких дисков централизованными ресурсами хранения.

Зачем это нужно? Перечислим несколько причин.

- Каждый клиент имеет возможность получить преимущество благодаря использованию сложного хранилища, оптимизированного по производительности, устойчивого к сбоям и допускающего восстановление данных.
- Повышается эффективность использования, поскольку каждый клиент имеет именно столько места для хранения, сколько ему требуется. Несмотря на то что объем памяти на виртуальных дисках фиксирован, он не ограничен стандартными размерами физических жестких дисков. Кроме того, блоки виртуальных дисков, которые клиент никогда не записывал, на самом деле на сервере не хранятся.
- В то же время система SAN обеспечивает большую гибкость и простоту переконфигурации. "Модернизацию жесткого диска" теперь можно выполнить с помощью одной-двух команд, посланных из терминального окна администратора.
- Методы обнаружения дублированных блоков могут снизить стоимость хранения одинаковых файлов на нескольких компьютерах.
- Стратегия резервного копирования для предприятия может быть унифицирована посредством точного копирования резервных блоков на сервере SAN. В некоторых ситуациях каждый клиент получает доступ к расширенным возможностям

[25] Хотелось бы высказать свою точку зрения. На самом деле существует много таких файловых систем, которые называются кластерными (или кластеризованными). Для обеспечения кластеризации необходимо применять специальные алгоритмы блокировки и синхронизации, поэтому кластеризованные системы, как правило, работают медленнее, чем стандартные файловые системы на локальных дисках. Файловая система VxFS может работать в кластеризованном и некластеризованном режимах, поэтому она подходит для любых ситуаций.

мгновенного копирования, характерным для менеджеров логических томов, независимо от операционной системы или используемой файловой системы.

Производительность всегда интересует системных администраторов, но, не зная конкретной реализации, трудно делать общие выводы о влиянии сетей SAN на производительность ввода-вывода на сервере. Сети порождают скрытые затраты и ограничения пропускной способности, которых нет у локальных дисков. Даже при наличии сложного коммутирующего оборудования, сети представляют собой лишь наполовину совместно используемые ресурсы, которые зависят от пропускной способности линий связи между клиентами. К преимуществам сетей SAN следует отнести тот факт, что крупные серверы SAN могут быть объединены в группы с памятью и кешами SSD. Они используют первоклассные компоненты и разделяют физический ввод-вывод между многими дисками. В общем, правильно реализованная сеть SAN работает значительно быстрее, чем локальное хранилище.

Однако все это обходится недешево. Такое аппаратное обеспечение относится к категории специализированного промышленного оборудования, поэтому вы сразу можете забыть о цене 80 долл. за диск из магазина электроники. Основными игроками на рынке SAN являются компании EMC, NetApp, HP, IBM и, как ни странно, Dell.

## Сети SAN

Поскольку основным фактором, влияющим на производительность сетевого хранения, является сеть, серьезные компании обычно закладывают в основу своей инфраструктуры оптоволоконные кабели.

Типичная скорость передачи данных по таким кабелям колеблется от 4 до 8 Гбайт/с, в противоположность скорости передачи данных в проводных сетях Ethernet, в которых она составляет 1 Гбайт/с. Однако сети Ethernet представляют собой быстро растущий сегмент ранка. Этому есть несколько причин. Две главные причины заключаются в том, что появились недорогие варианты технологии Ethernet со скоростью 10 Гбайт/с и постоянно усиливается роль виртуальных серверов; системы виртуализации обычно лучше поддерживают технологию Ethernet, чем оптоволоконные кабели. Разумеется, свою роль играет то, что системы, основанные на технологии Ethernet, не требуют инсталляции дорогой вспомогательной физической сетевой инфраструктуры.

Функциональные возможности SAN на основе технологии Ethernet можно реализовать с помощью нескольких протоколов обмена данными. Общим свойством этих протоколов является возможность эмуляции конкретного аппаратного интерфейса, который многие системы уже понимают.

Основным протоколом является iSCSI, представляющий системе виртуальный накопитель так, будто он присоединен к локальной шине SCSI. Кроме него, используются протоколы ATA-over-Ethernet (AoE) и Fibre-Channel-over-Ethernet (FCoE). Эти протоколы ориентированы на технологию Ethernet (а значит, ограничены географическими факторами), в то время как протокол iSCSI работает на основе протокола IP. В настоящее время протокол iSCSI занимает около 20% рынка SAN, настоящие технологии оптоволоконной связи — около 60%, а остальные решения — оставшиеся 20%.

Детали реализации оптоволоконной инфраструктуры выходят за пределы нашей книги, поэтому мы рассмотрим подробно только протокол iSCSI. С точки зрения операционной системы компьютера, оптоволоконные накопители SAN обычно выглядят как ряд дисков SCSI, которыми можно соответственно управлять.

## Протокол iSCSI: интерфейс SCSI на основе протокола IP

Протокол iSCSI позволяет реализовать сетевое хранение на основе существующего недорогого сетевого аппаратного обеспечения, не используя специальную оптоволоконную сеть и дорогие адаптеры оптоволоконных шин. Серверы SAN в этом случае станут узкоспециализированными системами, но при этом появится возможность воспользоваться преимуществами общедоступного аппаратного обеспечения.

Используя традиционную терминологию накопителей SCSI, можно сказать, что протокол iSCSI ссылается на сервер, который создает виртуальные диски, доступные через сеть в качестве "цели" iSCSI. Клиент, монтирующий эти диски, называется "инициатором". Это важно, если вспомнить, что именно клиент инициирует команды SCSI, а сервер отвечает на них.

Компоненты программного обеспечения, реализующего цель, и инициатор, расположенные на обоих концах связи iSCSI, отделены друг от друга. Все современные операционные системы содержат в себе инициатор, хотя часто он представляет собой необязательный компонент. Кроме того, большинство систем имеет также стандартную реализацию цели.

Протокол iSCSI формально определен в документе RFC3720. В отличие от большинства документов RFC, эта спецификация состоит из нескольких сотен страниц, в основном из-за сложности базового протокола SCSI. В большинстве случаев управление протоколом iSCSI является простым, если только для структурного управления и распознавания ресурсов хранения не используется необязательная служба Internet Storage Name Service (iSNS). Протокол iSNS, определенный в документе RFC4171, представляет собой адаптацию протоколов управления и раскрытия оптоволоконной связи для протокола IP, поэтому он представляет интерес для организаций, желающих использовать и оптоволоконную инфрастуктуру, и протокол iSCSI.

В отсутствие службы iSNS достаточно просто указать инициатору на соответствующий сервер, задать имя устройства iSCSI, к которому необходимо получить доступ, а также имя пользователя и пароль для аутентификации. По умолчанию аутентификация iSCSI использует протокол Challenge Handshake Authentication Protocol (CHAP), который изначально предназначался для протокола Point-to-Point Protocol (PPP) (см. документ RFC1994), так что пароли через сеть не передаются открытым текстом. При желании инициатор может аутентифицировать цель, используя второй распределенный ключ.

Протокол iSCSI можно выполнять поверх протокола IPsec, хотя это и не требуется. Если вы не используете туннель IPsec, то блоки данных не шифруются. В соответствии с документом RFC3720, соединения, не использующие протокол IPsec, должны использовать ключи CHAP длиной не менее 12 символов.

Цели и иниииаторы имеют имена iSCSI, причем для них определено несколько схем именования. Наиболее широко используются имена iSCSI Qualified Names (IQNs), имеющие несколько странный формат.

```
iqn.yyyy-mm.reversed_DNS_domain:arbitrary_name
```

В большинстве ситуаций все, что предшествует двоеточию, является фиксированным (т.е. нерелевантным) префиксом, характеризующим сайт. Вы можете реализовать собственную схему именования для фрагмента *абстрактное_имя* имени IQN. Месяц и год (*mm* и *yyyy*), уточняющие домен DNS, указываются для того, чтобы обеспечить защиту от возможного изменения домена. Для этого следует использовать оригинальную дату регистрации DNS. Реальное имя выглядит следующим образом.

```
iqn.1995-08.com.example:disk54.db.engr
```

Несмотря на специфичный формат имени IQN, префикс, отражающий реальный домен DNS или дату его возникновения, никакой роли не играет. В большинстве реализаций протокола iSCSI по умолчанию в качестве имени IQN используется имя поставщика, и все работает прекрасно. Не требуется даже, чтобы имена IQN, взаимодействующие друг с другом, имели совпадающие префиксы.

## Загрузка с тома iSCSI

Если вы собираетесь размещать в сетевом хранилище SAN важные данные, то не было бы разумным вообще отказаться от использования локальных жестких дисков? Вы можете не только отказаться от специальных процедур, необходимых для управления локальными дисками, но и позволить администраторам "заменять" загрузочные накопители простыми незагрузочными накопителями, осуществлять постоянную модернизацию и обеспечивать альтернативные конфигурации загрузки, включая даже системы Windows.

К сожалению, использование тома iSCSI в качестве загрузочного накопителя не получило широкой поддержки. По крайней мере, эта функциональная возможность не реализуется непосредственно и не стала одной из основных. Разные проекты Linux пытались ее реализовать, но все эти реализации были жестко привязаны к конкретному аппаратному обеспечению и конкретному программному обеспечению для инициатора протокола iSCSI, так что ни один из проектов загрузки протокола iSCSI не взаимодействует с доминирующим в настоящее время программным обеспечением OpeniSCSI для инициатора. Аналогично поддержка загрузки протокола iSCSI в системах Solaris и OpenSolaris пока находится на стадии разработки, но окончательного промышленного решения пока нет.

Единственным исключением является система AIX, которая уже давно и хорошо поддерживает протокол iSCSI. Версии AIX 5.3 и более поздние, работающие на аппаратном обеспечении POWER, обеспечивают полную поддержку загрузки протокола iSCSI поверх протокола IPv4.

## Специфика инициаторов iSCSI от разных производителей

Было, по крайней мере, четыре реализации инициатора протокола iSCSI для системы Linux. Некоторые из них сошли с арены, а остальные объединились. Единственным выжившим в конкурентной борьбе стал протокол Open-iSCSI, который является стандартным инициатором, включенным во все дистрибутивы рассмотренных нами операционных систем семейства Linux distributions. Для того чтобы его установить и запустить, инсталлируйте пакет **`open-scsi`** в системах Ubuntu и SUSE или пакет **`iscsi-initiator-utils`** в системе Red Hat.

Домашняя страница проекта расположена по адресу `open-iscsi.org`, но документацию там искать не следует. Похоже, что, кроме справочных страниц для команд **`iscsid`** и **`iscsiadm`**, представляющих собой описание реализации и интерфейса системы, никакой документации больше нет. К сожалению, административная модель для системы Open-iSCSI лучше всего описывается словом "креативная".

В системе Open-iSCSI "узел" — это цель протокола iSCSI, т.е. сущность с именем IQN. Система Open-iSCSI ведет базу данных, содержащую информацию об известных ей узлах, которые расположены в иерархии ниже каталога **`/etc/iscsi/nodes`**. В этом дереве хранятся параметры конфигурации для отдельных узлов. Значения по умолча-

нию задаются в файле **etc/iscsi/iscsid.conf**, но иногда они копируются во вновь определенные узлы, так что их функционирование нельзя назвать полностью предсказуемым. Процесс установки параметров целей очень неприятен; команда **iscsiadm** постоянно вынуждает вас изменять по одному параметру за один раз и задавать имя IQN и сервер в каждой командной строке.

Ситуацию спасает лишь то, что файл **iscsid.conf** и все файлы базы данных являются обычными редактируемыми текстовыми файлами. Следовательно, целесообразно использовать команду **iscsiadm** для модификации немногочисленных параметров и избегать ее при внесении многочисленных изменений.

Для того чтобы настроить систему на выполнение простой статической операции с одним именем пользователя и паролем для всех целей iSCSI, сначала необходимо отредактировать файл **iscsid.conf** и убедиться, что следующие строки набраны так, как показано ниже.

```
node.startup = automatic
node.session.auth.authmethod = CHAP
node.session.auth.username = chap_name
node.session.auth.password = chap_password
```

Мы привели эти строки вместе, но на самом деле в файле они расположены в разных местах. Этот файл содержит много комментариев и много опций конфигурации. Убедитесь, что они не дублируют друг друга.

Затем сориентируйте команду **iscsiadm** на целевой сервер и предоставьте ей возможность создать записи для каждой цели, распознанной при чтении каталога сервера. В данном примере мы сконфигурировали цель с именем `test` на основе сервера с именем `iserver`.

```
ubuntu$ sudo iscsiadm -m discovery -t st -p iserver
192.168.0.75:3260,1 iqn.1994-11.com.admin:test
```

Для каждой цели команда **iscsiadm** создает подкаталог в каталоге **/etc/iscsi/nodes**. Если существуют цели, с которыми вы не хотите работать, то просто примените к их каталогам конфигурации команду **rm -rf**.

Если сервер предлагает много целей, а вас итересует только одна, то воспользуйтесь следующими командами.

```
ubuntu$ sudo iscsiadm -m node -o new -p iserver
  -T iqn.1994-11.com.admin:test
New iSCSI node [tcp:[hw=default,ip=,net_if=default,iscsi_if=default]
  iserver,3260,-1 iqn.1994-11.com.admin:test] added
```

Как ни странно, эти два метода дают одинаковые результаты, создавая при этом разные иерархии каталогов внутри каталога **/etc/iscsi/nodes**. Какую бы версию вы ни применили, проверьте текстовые файлы, существующие в иерархии, чтобы убедиться в правильности их параметров конфигурации.

Если вы зашли в цель самостоятельно, возможно, вам придется самому задать свойство `node.startup` равным `automatic`.

Затем вы можете соединиться с удаленными целями с помощью команды **iscsiadm -m node -l**.

```
ubuntu$ sudo iscsiadm -m node -l
Logging in to [iface: default, target: iqn.1994-11.com.admin:test, portal:
  192.168.0.75,3260]
Login to [iface: default, target: iqn.1994-11.com.admin:test, portal:
  192.168.0.75,3260]: successful
```

Для того чтобы проверить, видит ли система дополнительный диск, можно выполнить команду **fdisk -l**. (Файлы устройств для дисков iSCSI disks называются точно так же, как и любые другие диски named SCSI.) Если вы настроили файлы конфигурации так, как указано выше, то соединения будут автоматически восстановлены во время загрузки.

Для целевой службы iSCSI в системах семейства Linux предпочтительной реализацией является пакет iSCSI Enterprise Target, размещенный по адресу `iscsitarget.sourceforge.net`. Обычно он доступен для загрузки под именем **iscsitarget**.

Система Solaris содержит пакеты цели и инициатора; оба они являются необязательными. Все пакеты, связанные с протоколом iSCSI, в своих именах содержат слово "iscsi". Находясь на стороне инициатора, инсталлируйте пакет SUNWiscsi, а затем выполните перезагрузку системы.

Конфигурационного файла для этого пакета не существует; вся конфигурация создается командой **iscsiadm**, имеющей довольно странный синтаксис. Четыре главные операции (**add**, **modify**, **list** и **remove**) можно применить к разнообразным аспектам конфигурации инициатора. Основная конфигурация инициатора и соединение с целевым сервером **iserver** осуществляются с помощью следующих строк.

```
solaris$ sudo iscsiadm modify initiator-node -a CHAP -H testclient
solaris$ sudo iscsiadm modify initiator-node -C
Enter secret: <password for testclient>
Re-enter secret: <password for testclient>
solaris$ sudo iscsiadm modify discovery -s enable
solaris$ sudo iscsiadm add static-config iqn.1994-11.com.admin:test,iserver
solaris$ sudo iscsiadm list target -S
Target: iqn.1994-11.com.admin:test
   Alias: -
   TPGT:  1
   ISID:  4000002a0000
   Connections: 1
   LUN:  0
         Vendor:  IET
         Product: VIRTUAL-DISK
         OS Device Name: /dev/rdsk/c10t3d0s2
```

Теперь можно задать конфигурацию диска обычным способом (например, выполнив команду **zpool create iscsi c10t3d0**).

Первая команда задает для инициатора режим аутентификации CHAP, а имя пользователя задает равным `testclient`. Опция **-C** задает пароль; эту опцию нельзя комбинировать с другими. Кроме того, при желании можно задавать имя и пароль индивидуально для каждой цели.

Команда **modify discovery** позволяет использовать статически сконфигурированные цели, а команда **add** задает сервер и имя конкретной цели. Все эти конфигурации сохраняются во время перезагрузок.

Для обслуживания целей iSCSI в других системах необходимо инсталлировать пакет SUNWiscsitgt. Его механизм управления на стороне инициатора структурирован так же, но команда имеет имя **iscsitadm**, а не **iscsiadm**.

Для использования протокола iSCSI в системах семейства HP-UX, необходимо загрузить программное обеспечение для инициатора протокола iSCSI с сайта `software.hp.com` и инсталлировать с помощью инструмента Software

Distributor. При этом требуется перестройка и перезагрузка ядра. К счастью, система является хорошо документированной и сопровождается справочным руководством *HP-UX iSCSI Software Initiator Support Guide*, которое можно загрузить с сайта `docs.hp.com`.

В большинстве случаев конфигурацию инициатора можно задать с помощью команды **iscsiutil**, инсталлированной в каталоге **/opt/iscsi/bin**. Для задания имени IQN инициатора применяется команда **iscsiutil -l** *имя_iqn*, для задания глобального пользовательского имени для алгоритма аутентификации CHAP используется команда **iscsiutil -u -N** *пользователь* (ее также можно применять к отдельным серверам или отдельным целям), а для задания глобального пароля для алгоритма аутентификации CHAP — команда **iscsiutil -u -W** *пароль*.

Затем можно добавить цели на конкретном сервере с помощью команды **iscsiutil -a -I** *сервер*. Для того чтобы активировать серверное соединение и создать виртуальный дисковый накопитель, следует выполнить команду **ioscan -NH 64000**. Состояние системы проверяется с помощью команды **iscsiutil -p -o**.


Инициатор протокола iSCSI в системе AIX уже инсталлирован и готов к использованию. В типичном для системы AIX стиле большинство конфигураций реализуется с помощью системных баз данных ODM. Устройство iscsi0 представляет конфигурацию инициатора в целом, а индивидуальные целевые устройства можно определить с помощью записей ODM или в текстовых файлах конфигурации в каталоге **/etc/iscsi**. Текстовые файлы конфигурации, на первый взгляд, более надежны.

Система AIX не различает имя IQN инициатора и его пользовательское имя CHAP. Имя IQN задается на устройстве iscsi0, следовательно, на каждом сервере необходимо использовать одно и то же пользовательское имя CHAP. Для того чтобы получить возможность быстро изменять конфигурацию, сначала необходимо присвоить имени IQN соответствующее значение.

```
aix$ sudo chdev -l iscsi0 -a initiator_name='iqn.1994-11.com.admin:client'
```

В данном примере мы использовали пользовательское имя CHAP, которое отличалось от своих аналогов в других системах, поскольку "testclient", с формальной точки зрения, не является корректным именем IQN для инициатора (хотя на самом деле оно также прекрасно подходит).

В файл **/etc/iscsi/targets** добавим следующую запись.

```
iserver 3260 iqn.1994-11.com.admin:test "пароль-chap"
```

Параметр 3260 — это стандартный порт сервера для протокола iSCSI; мы включили его только потому, что этот порт предусмотрен форматом файла. Для того чтобы активировать новый диск iSCSI, необходимо всего лишь выполнить команду **cfgmgr -l iscsi0**. Команда **cfgmgr** не выводит никаких подтверждений, но убедиться в том, что новое устройство появилось, можно, просмотрев каталог **/dev** (в нашем примере новый диск имеет имя **/dev/hdisk2**) или выполнив команду **smitty devices**, перейдя в категорию Fixed Disk и просмотрев элементы списка. Вероятно, второй вариант более безопасный, поскольку команда **smitty** явно демонстрирует, что диск **hdisk2** является томом iSCSI.

Для того чтобы отсоединить устройство iSCSI, необходимо не только отредактировать файл конфигурации и перезагрузить его с помощью команды **cfgmgr**, но и удалить диск из списка Fixed Disk команды **smitty**.

## 8.12. Упражнения

8.1. Укажите, на что должен обратить особое внимание системный администратор при разработке архитектуры хранилища для каждого приложения.

а) сервер, на котором будут храниться рабочие каталоги около 200 пользователей

б) область подкачки для основного DNS сервера организации

в) хранилище для большого потока спама

г) крупная база данных InnoDB (MySQL)

8.2. Менеджеры логических томов — мощные механизмы, но их иногда трудно понять. Поэкспериментируйте с добавлением, удалением дисков, входящих в группу томов, и изменением их размеров. Покажите, как удалить устройство из одной группы томов и доьбавить его в другой. Что делать, если необходимо переместить логический том из одной группы томов в другую?

8.3. ★Используя информацию, опубликованную в Интернете, укажите накопители SCSI и SATA с наивысшей производительностью. Учитывают ли тестовые программы, используемые для оценки этих накопителей, что они могут применяться в качестве загрузочного диска на занятом сервере? Какие дополнительные расходы вы готовы нести, чтобы приобрести накопители SCSI, и какой прирост производительности вы хотите получить за эти деньги?

8.4. ★ Добавьте в вашу систему диск и настройте раздел или логический том на новом диске как резервный корневой раздел. Убедитесь, что вы можете запуститься с резервного корневого раздела и что система после этого работает нормально. Проследите по журналу все действия, которые необходимо выполнить, чтобы решить поставленную задачу. Возможно, вам поможет команда **`script`**. (Она требует прав суперпользователя.)

8.5. ★ Что такое суперблок и для чего он нужен? Посмотрите на определение структуры суперблока в системе ext4 в заголовочных файлах ядра и обсудите, что представляют собой поля в этой структуре.

8.6. ★★ Используя команду **`mdadm`** и ее опцию **`-f`**, смоделируйте сбойный диск в массиве RAID в системе Linux. Удалите этот диск из массива и добавьте обратно. Как выглядит диск **`/proc/mdstat`** на каждом из этих этапов?

8.7. ★★ Какие поля хранятся в индексном дескрипторе на файловой системе ext4? Перечислите содержимое индексного дескриптора, которое хранится в файле **`/etc/motd`**. Где хранится имя этого файла? (Вам могут помочь команды **`hexdump`** и **`ls -i`**.)

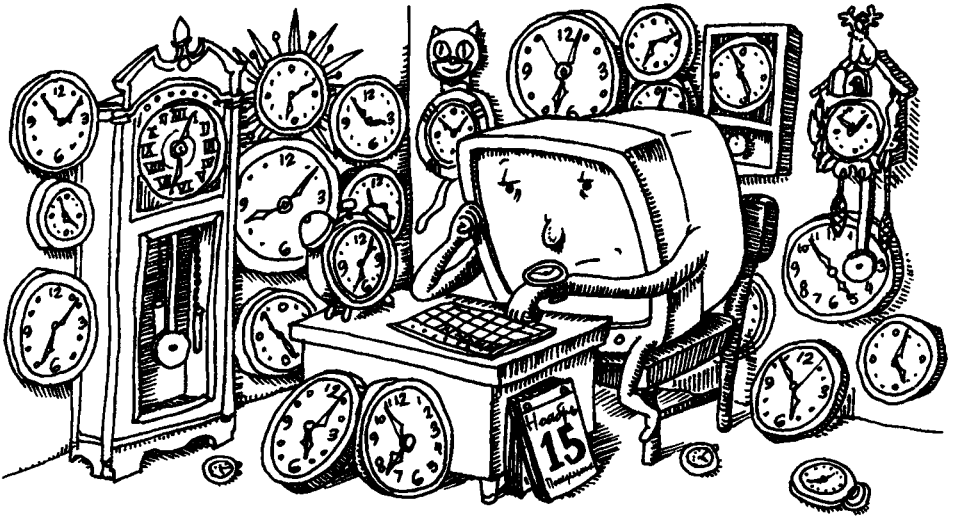
8.8. ★★ Проверьте содержимое каждого файла каталога с помощью программ **`od`** или **`hexdump`**. Каждая запись переменной длины представляет собой файл в заданном каталоге. Просмотрите дисковую структуру каталога и объясните каждое поле, используя в качестве примера реальный файл каталога. Затем загляните в каталог **`lost+found`** используемой вами файловой системы. Почему там так много имен, если каталог **`lost+found`** пуст?

8.9. ★★★★★ Напишите программу, которая отображает содержимое файлов **`/etc/motd`** и **`/etc/magic`**. Но не обращайтесь к файлам напрямую: откройте файл устройства, соответствующий корневому разделу, и воспользуйтесь системными вызовами **`seek`** и **`read`** для поиска нужных блоков данных. Файл **`/etc/motd`**

обычно невелик и содержит только "информационные" блоки. При работе с файлом **/etc/magic**, скорее всего, потребуется анализировать "ссылочные" блоки. (Если нет, выберите текстовый файл побольше.) **Подсказка:** Найдя в системных файлах заголовков описание структуры индексного дескриптора, убедитесь, что это дескриптор, хранящийся на диске, а не в ядре. (Необходим доступ с правами суперпользователя.)

# Глава 9

# Периодические процессы



Ключ к устойчивости и надежности систем лежит в автоматизации выполняемых в ней задач и успешном написании сценариев. Например, программа **adduser** может подключать новых пользователей быстрее, чем это сделает администратор вручную, причем с гораздо меньшей вероятностью ошибки. Практически любую задачу можно запрограммировать в сценариях интерпретатора команд либо на языках Perl или Python.

В некоторых случаях желательно, чтобы сценарий или команда выполнялись без вмешательства оператора. Можно, в частности, сделать так, чтобы сценарий проверял (скажем, каждые полчаса), как работают сетевые маршрутизаторы и мосты, и при обнаружении проблем посылал администратору сообщение по электронной почте[1].

## 9.1. Демон cron: системный планировщик

Демон **cron** — это стандартный инструмент для периодического выполнения команд. Он запускается на этапе начальной загрузки системы и выполняется до тех пор, пока система не будет выключена.

Демон **cron** читает файлы конфигурации, содержащие списки командных строк и расписание их вызова. Командные строки обрабатываются интерпретатором **sh**, поэтому почти все, что можно делать в данном интерпретаторе вручную, разрешается перепоручать демону **cron**[2].

Файл конфигурации демона **cron** называется **crontab** (сокращение от "cron table" — таблица демона **cron**). Пользовательские **crontab**-файлы хранятся в каталоге **/var/spool/cron**. Для каждого пользователя создается не более одного такого файла: один

[1] Многие организации идут еще дальше: в них сценарий отправляет текстовое сообщение на телефон администратора, уведомляя его о возникновении проблем. Подробнее читайте в главе 21.

[2] Можно настроить демон ***cron*** на использование других интерпретаторов.

для суперпользователя (**root**), один для пользователя **jsmith** и т.д. В качестве имени файла выбирается регистрационное имя пользователя, которому он принадлежит, и на основании этого имени демон **cron** выясняет, какое значение UID нужно использовать при выполнении команд, содержащихся в файле. Команда **crontab** передает **crontab**-файлы в каталог **/var/spool/cron** и из него.

Несмотря на различия в конкретных реализациях, все версии демона **cron** стремятся минимизировать время, затрачиваемое на синтаксический анализ файлов конфигурации и выполнение вычислений. Команда **crontab** помогает поддерживать эффективность работы демона **cron**, уведомляя его об изменениях **crontab**-файлов. Следовательно, вы не должны редактировать **crontab**-файлы сами, поскольку это может привести к тому, что демон **cron** не заметит ваших изменений. Если возникнет ситуация, когда **cron**, казалось бы, "не замечает" модификацию своих **crontab**-файлов, воспользуйтесь сигналом отбоя (**HUP**), который в большинстве систем заставляет его повторно прочитать их.

🕮 Подробнее о системе syslog написано в главе 11.

Демон **cron** обычно выполняет свою работу "молча", но в большинстве версий может вестись файл регистрации (как правило, это файл **/var/cron/log** или **/var/adm/cron/log**), в котором отражаются все выполняемые команды и время их запуска (см. табл. 9.2).

В одних системах при создании файла регистрации автоматически включается режим регистрации, а при удалении этого файла регистрация выключается. В других системах режим регистрации задается в файле конфигурации. Кроме того, демон **cron** может пользоваться услугами системы syslog. Файл регистрации быстро увеличивается в размерах и редко приносит пользу; лучше не включать регистрацию вообще, если только вы не занимаетесь устранением какой-то конкретной проблемы.

## 9.2. Формат crontab-файлов

Все **crontab**-файлы в системе имеют одинаковый формат. Комментарии начинаются со знака решетки в первой позиции строки. Каждая строка, не являющаяся комментарием, содержит шесть полей и представляет одну команду.

*минута час день месяц день_недели команда*

Первые пять полей (*минута*, *час*, *день*, *месяц* и *день_недели*) содержат информацию о времени запуска команды. Они отделяются друг от друга пробелами, но в поле *команда* пробел выполняет свою обычную функцию разделителя аргументов. Описание этих полей приведено в табл. 9.1.

**Таблица 9.1. Спецификации времени в crontab-файле**

| Поле | Описание | Диапазон |
|---|---|---|
| *минута* | Минута часа | от 0 до 59 |
| *час* | Час дня | от 0 до 23 |
| *день* | День месяца | от 1 до 31 |
| *месяц* | Месяц года | от 1 до 12 |
| *день_недели* | День недели | от 0 до 6 (0 = воскресенье) |

Каждое поле времени может содержать следующее:

- звездочку, которая обозначает любую цифру;

- целое число, трактуемое буквально;
- два разделенных дефисом целых числа, задающих диапазон значений;
- диапазон, за которым стоит косая черта и значение шага, например 1–10/2 (только для Linux);
- целые числа или диапазоны, разделенные запятыми (искомое время соответствует любому из указанных значений).

Например, строка

```
45  10  *   *  1-5
```

означает "10 часов 45 минут, с понедельника по пятницу". Совет: никогда не ставьте звездочку в первое поле, иначе команда будет выполняться каждую минуту.

С полями *день_недели* и *день* сопряжена потенциальная двусмысленность, которую необходимо учитывать. День можно рассматривать как день недели и число месяца. Если указаны оба поля, то искомому дню достаточно удовлетворять одному из этих требований, чтобы пройти отбор. Например, спецификация

```
0,30  *   13  *   5
```

означает "каждые полчаса по пятницам и каждые полчаса тринадцатого числа месяца", но не "каждые полчаса в пятницу, 13-го".

Поле *команда* содержит командную строку, выполняемую интерпретатором **sh**. Это может быть любая допустимая команда интерпретатора (брать ее в кавычки не нужно). Считается, что поле *команда* продолжается до конца строки и может содержать пробелы и символы табуляции.

Несмотря на то что интерпретатор **sh** включен в выполнение команды, он не действует как экземпляр интерпретатора, запускаемый при входе пользователя в систему (login shell), и поэтому не считывает содержимое файлов **~/.profile** или **~/.bash_profile**. В результате переменные среды указанной команды могут быть установлены несколько не так, как вы ожидали. Если вам кажется, что команда, выполняемая из оболочки, работает прекрасно, но "глючит" при вводе в файл **crontab**, значит, по всей вероятности, виновата среда. При необходимости можно всегда "обернуть" команду в сценарий, который устанавливает соответствующие переменные среды.

Знак процента (**%**) используется вместо символа новой строки в поле *команда*. В реальную команду включается только текст до первого знака процента, а остальные строки передаются команде в качестве стандартного входного потока.

Приведем несколько примеров допустимых команд **crontab**-файла.

```
echo The time is now `date` > /dev/console
mail -s Reminder evi@anchor % Don't forget to write your chapters.
cd /etc; /bin/mail -s "Password file" evi < passwd
```

А теперь рассмотрим полные примеры записей.

```
30  2  *  *  1 (cd /home/joe/project; make)
```

Эта строка будет активизироваться в 2:30 по понедельникам. Она инициирует выполнение утилиты **make** в каталоге **/home/joe/project**. Такую запись можно использовать для запуска длительного процесса компиляции в то время, когда в системе не работают другие пользователи. Вся выходная информация, выдаваемая командами **crontab**-файла, обычно посылается по электронной почте "владельцу" файла[3].

[3] Пользователю, от имени которого создан файл. Фактическим владельцем ***crontab***-файлов в общем случае является пользователь ***root***.

```
20 1 * * * find /tmp -atime +3 -type f -exec rm -f { } ';'
```

Эта строка будет активизироваться каждый день в 1:20. Соответствующая команда удаляет из каталога **/tmp** все файлы, к которым за последние 3 дня никто не обращался.

```
55 23 * * 0-3,6 /staff/trent/bin/checkservers
```

Эта строка запускает сценарий **checkservers** в 23:55 каждый день, кроме четверга и пятницы.

Демон **cron** не пытается компенсировать пропуск команд, время выполнения которых прошло, пока система не работала. Однако демоны **cron** в системах Linux и HP-UX вполне адекватно реагируют на небольшие смещения времени, связанные с переходами на летнее время и обратно. Другие версии демона **cron** могут пропустить команды или выполнить их дважды, если эти команды были назначены как раз на переходный период (например, обычно между 1:00 и 3:00 в США)[4].

## 9.3. Управление crontab-файлами

Команда **crontab** *имя_файла* инсталлирует указанный **crontab**-файл, заменяя его предыдущую версию, если таковая имеется. Команда **crontab -e** проверяет копию **crontab**-файла текущего пользователя, загружает ее в текстовый редактор (указанный в переменной среды **EDITOR**) для последующего изменения, а затем повторно записывает файл в системный каталог. Команда **crontab -l** отображает содержимое **crontab**-файла, а команда **crontab -r** удаляет этот файл.

Пользователю **root** разрешается задавать аргумент *имя_пользователя*, чтобы можно было просматривать или редактировать **crontab**-файлы других пользователей. Например, команда **crontab -r jsmith** удаляет **crontab**-файл, принадлежащий пользователю **jsmith**, а команда **crontab -e jsmith** редактирует этот файл. В системе Linux разрешается использование обоих аргументов *имя_пользователя* и *имя_файла* в одной команде, поэтому для устранения неоднозначности аргументу *имя_пользователя* должен предшествовать флаг **-u** (например, **crontab -u jsmith crontab.new**).

Будучи вызванными без аргументов, большинство версий команды **crontab** будут пытаться прочитать **crontab**-файл из своего стандартного входного потока. Если этот режим был активизирован случайно, не пытайтесь выйти из него, нажимая комбинацию клавиш <Ctrl+D>, так как весь **crontab**-файл будет удален. Необходимо нажать <Ctrl+C>. В Linux для того, чтобы команда **crontab** обратилась к своему стандартному входному потоку, в качестве аргумента *имя_файла* ей необходимо передать дефис. Коротко и ясно!

Два файла конфигурации **cron.deny** и **cron.allow** содержат информацию о том, кому из пользователей разрешено предоставлять **crontab**-файлы. В разных системах эти файлы расположены в разных каталогах (см. табл. 9.2).

Если файл **cron.allow** существует, то он содержит список пользователей, имеющих доступ к демону **cron** (по одному имени в строке). Пользователи, которые в списке отсутствуют, не имеют права выполнять команду **crontab**. Если файла **cron.allow** нет,

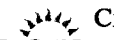
[4] Один из наших помощников зафиксировал случай, когда демон **cron** потреблял 100% времени центрального процессора, поскольку системная дата была установлена неправильно. Локальный часовой пояс имел отрицательное смещение относительно всемирного времени (GMT), поэтому местное время после вычисления давало отрицательную величину, и, как следствие, **cron** был "сбит с толку". В большинстве систем "бортовые" часы оснащены батарейками, поэтому переустановка часов — не такое уж необычное явление, как могло бы показаться. Нестабильность показателя времени — распространенный симптом "севшей" или "садящейся" батарейки.

проверяется файл **cron.deny**. Он также содержит список пользователей, но с противоположным назначением: доступ разрешен всем, кроме лиц, указанных в списке.

**Таблица 9.2. Файлы и каталоги демона cron в различных системах**

| Система | Каталог файлов allow и deny | Доступ | Журнальный файл |
|---|---|---|---|
| Linux | /etc | Все пользователи | Используется система **syslog** |
| Solaris | /etc/cron.d | Все пользователи | /var/cron/log |
| HP-UX | /usr/lib/cron | Только **root** | /var/adm/cron/log |
| AIX | /var/adm/cron | Все пользователи | /var/adm/cron/log |

Если нет ни одного из этих файлов, то, значит, нет единого правила для всех: в одних системах всем пользователям по умолчанию разрешается создавать **crontab**-файлы, а в других на это имеет право только пользователь **root**. На практике стартовый файл **cron.allow** или **cron.deny** часто включается в стандартную инсталляцию операционной системы, поэтому вопрос о **crontab**-поведении без файлов конфигурации остается спорным. Среди наших примеров систем только в HP-UX по умолчанию блокируется доступ к **crontab**-файлам для непривилегированных пользователей. Необходимо подчеркнуть, что права доступа регулируются командой **crontab**, а не демоном **cron**. Если пользователь найдет способ обходным путем поместить в системный каталог свой **crontab**-файл, демон **cron** будет благополучно выполнять указанные в нем команды.



Система Solaris в этом смысле ведет себя несколько по-другому. Ее демон **cron** удостоверяется в том, что учетная запись пользователя не заблокирована значением ***LK*** в файле **/etc/shadow**. Если окажется, что проверяемая учетная запись все же заблокирована, демон **cron** не выполнит задания этого пользователя. Важно предотвратить выполнение заданий, подготовленных заблокированными пользователями (причем не имеет значения, как они были заблокированы: случайно или намеренно). Если вы хотите, чтобы пользователь имел действующую учетную запись с точки зрения демона **cron**, но не действующий пароль, выполните команду **passwd -N** *пользователь*.

## 9.4. Использование версии демона Vixie-cron

Версия демона **cron**, включенная в дистрибутивы Linux (в том числе и наши три примера), обычно именуется как **ISC Cron** или **vixie-cron**; последнее название обязано имени автора Пола Викси (Paul Vixie). Эта версия представляет собой осовремененный вариант, который предоставляет дополнительные функции при меньшей неразберихе.

Основное отличие нового варианта состоит в том, что, в дополнение к поиску пользовательских **crontab**-файлов, версия **vixie-cron** также подчиняется системным **crontab**-записям, хранящимся в файле **/etc/crontab** и каталоге **/etc/cron.d**. Эти файлы имеют формат, который несколько отличается от формата пользовательских **crontab**-файлов тем, что они позволяют выполнение команд, выданных произвольным пользователем. Дополнительное поле *имя_пользователя* стоит перед названием команды. Поле *имя_пользователя* не представлено в обычных **crontab**-файлах, поскольку имя **crontab**-файла предоставляет ту же информацию (даже в системах Linux).

Демон **cron** точно так же интерпретирует записи из файла **/etc/crontab** и каталога **/etc/cron.d**. В общем файл **/etc/crontab** предназначен для системных администраторов, которые поддерживают его вручную, в то время как каталог **/etc/cron.d** играет

роль хранилища, куда пакеты программ могут инсталлировать любые **crontab**-файлы, которые им могут понадобиться. Файлы, хранимые в каталоге **/etc/cron.d**, называются (согласно действующему соглашению) по именам пакетов, которые их инсталлируют, но демона **cron** не заботит исполнение этого соглашения.

Временны*е* диапазоны в **crontab**-файлах версии **vixie-cron** могут включать любое значение шага. Например, ряд 0, 3, 6, 9, 12, 15, 18 кратко можно записать в виде 0–18/3. Вы можете также использовать трехбуквенное мнемоническое обозначение для названий месяцев и дней, но не в сочетании с диапазонами. Насколько нам известно, эта функция работает только с английскими названиями. В **crontab**-файле версии Vixie-**cron** можно задавать переменные среды и их значения. Подробнее см. **man**-страницу для **crontab(5)**.

Посредством системы **syslog** демон **vixie-cron** регистрирует свои действия, используя уровень "cron", при том, что большинство сообщений представлено на уровне "info". Стандартные конфигурации системы **syslog** отправляют **cron**-данные регистрационного характера в его собственный файл.

По непонятным причинам демон **cron** в системе Red Hat был переименован в **crond**. Но это все тот же демон **vixie-cron**, который все мы знаем и любим.

## 9.5. Стандартные применения демона cron

Существует ряд стандартных задач, решать которые удобнее всего с помощью демона **cron**. Соответственно, большинство записей **crontab**-файла пользователя **root** служит именно этим целям. В данном разделе рассмотрим подобные задачи и элементы **crontab**-файла, обеспечивающие их реализацию.

Системы часто поставляются с готовыми **crontab**-файлами. Если необходимо отключить стандартные записи, превратите их в комментарии, вставив в начало каждой строки знак решетки (#). Не удаляйте сами записи, возможно, они еще пригодятся.

Помимо файлов каталога **/etc/cron.d**, в дистрибутивах Linux есть также **crontab**-файлы, организующие запуск сценариев в ряде заранее известных каталогов. Это позволяет программам добавлять собственные периодические задания, не редактируя **crontab**-файлы. Например, сценарии, помещаемые в каталог **/etc/cron.daily**, запускаются раз в день, а сценарии каталога **/etc/cron.weekly** — раз в неделю. Вы можете также помещать файлы в эти каталоги вручную.

Многие организации сталкиваются с незначительными, но повторяющимися сетевыми проблемами из-за того, что администраторы сконфигурировали демон **cron** для выполнения одной и той же команды на сотнях компьютеров в одно и то же время. Синхронизация часов всех компьютеров посредством протокола NTP еще больше усугубляет проблему. Эту проблему легко устранить с помощью сценария выполнения команд со случайной задержкой или путем настройки файла конфигурации, однако она может с трудом поддаваться диагностированию, поскольку ее признаки быстро полностью исчезают.

### Простые напоминания

Не желая оставить Google Calendar без дела, демон **cron** тем не менее может вполне оказаться полезным для организации простых напоминаний: о днях рождения, ожидаемом времени (например, времени платежа), периодически повторяющихся задачах и пр.

Это особенно полезно в случае, когда процесс напоминания необходимо интегрировать с другими доморощенными программами.

Следующая **crontab**-запись реализует простое почтовое напоминание. (Приведенные строки в действительности представляют собой одну длинную строку.)

```
30 4 25 * * /usr/bin/mail -s "Time to do the TPS reports"
  owen@atrust.com%TPS reports are due at the end of the month! Get
  busy!%%Sincerely,%cron
```

Обратите внимание на то, что символ "**%**" используется как для отделения команды от входного текста, так и для обозначения конца строки в этом тексте. Эта запись однократно отправляет электронную почту 25-го числа каждого месяца.

## Чистка файловой системы

В любой системе есть ненужные файлы (естественно, речь идет не о системных файлах). Например, при аварийном завершении программы ядро создает файл (обычно он называется **core**, **core**.*pid* или *program*.**core**), содержащий образ адресного пространства программы. Такие файлы используются разработчиками программного обеспечения, но для администраторов это бесполезная трата дискового пространства. Пользователи часто вообще не знают о назначении **core**-файлов, поэтому предпочитают не брать на себя ответственность за их удаление[5].

🕮 NFS рассматривается в главе 18.

Еще один источник лишних файлов — это NFSv3 (Network File System — сетевая файловая система). Серверы NFSv3 сохраняют файлы, которые уничтожены в локальной системе, но продолжают использоваться удаленным компьютером. В большинстве реализаций таким файлам присваиваются имена вида **.nfs***xxx*, где *xxx* — числовой код. Иногда об этих файлах забывают и оставляют их на диске, думая, что они удалены.

Многие программы создают в каталоге **/tmp** или **/var/tmp** временные файлы, которые по той или иной причине не удаляются. Некоторые программы, особенно текстовые редакторы, делают резервные копии каждого файла, с которым они работают.

Частичное решение проблемы файлового "мусора" состоит в регулярном (каждую ночь) восстановлении свободного дискового пространства. В современных системах обычно имеются встроенные средства подобной очистки, поэтому необходимо убедиться в том, что они соответствуют потребностям организации.

Во всех показанных ниже примерах для удаления ненужных файлов применяется команда **find**.

```
find / -xdev -type f '(' -name core -o name 'core.[0-9]*' -o name '*.core' ')'
   -atime +7 -exec rm -f { } ';'
```

Эта команда удаляет файлы с именем **core**, которые не использовались в течение недели. Аргумент **-xdev** гарантирует, что команда **find** будет выполнена только в пределах корневой файловой системы, что очень важно в сетях, где возможно монтирование множества файловых систем[6]. Если необходимо очистить несколько файловых систем,

[5] Многие ядра систем можно сконфигурировать так, чтобы дампы памяти помещались в отдельный каталог или вообще не генерировались. Загляните в руководство, используя в Linux команду ***man core*** или в Solaris команду ***man coreadm***.

[6] Не все версии команды ***find*** поддерживают аргумент ***-xdev***. В некоторых системах он называется ***-x***.

задайте для каждой из них свою команду (помните, что каталог **/var** часто является отдельной файловой системой).

Аргумент **-type f** имеет большое значение, поскольку исходный код ядра Linux содержит каталог с именем **core**, удаление которого весьма нежелательно, не так ли?

```
find / -xdev -atime +3 '(' -name '#*' -o -name '.#*' -o -name '*.CKP' -o
        -name '*~' -o -name '.nfs*' ')' -exec rm -f { } ';'
```

Эта команда удаляет неиспользуемые в течение трех дней файлы, имена которых начинаются с префикса **#**, **.#** или **.nfs** либо заканчиваются расширением **~** или **.CKP**. Это типичные признаки временных и резервных файлов различных текстовых редакторов.

🕮 Подробнее об опциях команды **mount** написано в разделе 6.2.

В целях повышения производительности некоторые администраторы используют опцию **noatime** команды **mount**, чтобы предотвратить хранение файловой системой временных меток момента обращения к файлам. Подобная конфигурация помешает работе обеих приведенных выше команд **find**, поскольку файлы будут казаться такими, к каким не было обращений, даже если они недавно были активными. К сожалению, в случае ошибки команда удаляет файлы. Поэтому, прежде чем использовать приведенные команды, убедитесь, что система поддерживает временные метки доступа к файлам.

```
cd /tmp; find . ! -name . ! -name lost+found -type d -mtime +3
    -exec /bin/rm -rf { } ';'
```

Эта команда рекурсивно удаляет все подкаталоги каталога **/tmp**, не модифицируемые в течение трех дней. В большинстве систем обычные файлы в каталоге **/tmp** удаляются сценариями запуска системы, но в некоторых системах подкаталоги не удаляются. Если существует каталог **lost+found**, то он не включается в число удаляемых. Это важно, если каталог **/tmp** является отдельной файловой системой. Подробнее о каталоге **lost+found** рассказывалось в разделе 8.9.

Если в системе применяется одна из перечисленных выше команд, следует заранее уведомить пользователей о том, какими принципами руководствуется администратор при чистке файловой системы.

## Распространение конфигурационных файлов по сети

🕮 Подробнее о распространении конфигурационных файлов по сети рассказывается в главе 19.

При сетевом администрировании во многих случаях удобно пользоваться единой версией конфигурационных файлов, например базой почтовых псевдонимов. Обычно базовый механизм совместного использования ресурсов представляет собой некоторую форму опроса или периодического распространения ресурсов, что делает эту задачу идеальной для демона **cron**. Основные версии этих файлов можно распространять по сети каждую ночь с помощью утилиты **rsync**, **rdist**.

Иногда необходима последующая обработка файлов. Например, если в файле **sendmail.cf** не задана опция **AutoRebuildAliases**, может потребоваться, чтобы пользователь выполнял команду **newaliases** для преобразования текстового файла почтовых псевдонимов в хешированный формат, используемый программой **sendmail**. Может понадобиться и загрузка файлов в административную базу данных, например в NIS.

## Ротация журнальных файлов

Системы по-разному относятся к управлению своими журнальными файлами по умолчанию, поэтому вам, возможно, стоит изменить стандартные установки в этой области, чтобы согласовать их с вашими локальными стратегиями. Под "ротацией" журнального файла понимают разделение его на сегменты по размеру и дате с сохранением при этом нескольких старых версий каждого журнального файла постоянно доступными. Поскольку ротация представляет собой периодически повторяющееся событие, которое можно выполнять по расписанию, такую процедуру лучше всего планировать с помощью демона **cron**. Подробнее о ротации рассказывается в главе 11.

# 9.6. Упражнения

9.1. Локальный пользователь системы стал злоупотреблять своим правом создания **crontab**-файлов, периодически запуская ресурсоемкие задания. Вы несколько раз безуспешно просили его прекратить это делать и теперь вынуждены лишить его соответствующих привилегий. Перечислите действия, которые необходимо предпринять для того, чтобы удалить текущий **crontab**-файл пользователя и запретить ему впредь создавать такие файлы.

9.2. Придумайте три задания, которые требуется запускать регулярно (те, что были приведены в главе, не использовать). Составьте **crontab**-записи для каждого задания и укажите, в какие файлы их необходимо поместить.

9.3. Найдите **crontab**-файлы в своей системе и выберите из них три записи. Опишите, когда активизируется каждая из них, что она делает и зачем, по вашему мнению, она нужна. (Необходим доступ с правами суперпользователя.)

9.4. ☆ Напишите сценарий, который синхронизирует ваши конфигурационные файлы (**~/.[a-z]***) между всеми компьютерами, на которых у вас есть учетная запись. Сделайте так, чтобы этот сценарий регулярно запускался демоном **cron**. (Можно ли просто копировать все файлы, имена которых начинаются с точки? Как вы поступите с каталогами? Нужно ли создавать резервные копии файлов, заменяемых на целевом компьютере, перед их перезаписью?)

# Глава 10

# Резервное копирование


В большинстве коммерческих организаций информация, хранящаяся в компьютерном виде, стоит дороже самих компьютеров. Кроме того, ее гораздо труднее восстановить. Защита этой информации является одной из наиболее важных (и, к сожалению, самых трудоемких) задач системного администратора.

Потерять важную информацию можно как угодно. К порче файлов данных приводят ошибки в программном обеспечении. Пользователи случайно удаляют то, над чем работали всю жизнь. Хакеры и обозленные служащие стирают жесткие диски. Сбои в аппаратных средствах и стихийные бедствия выводят из строя целые машинные залы.

Резервные копии позволяют администратору восстанавливать файловую систему (или любую ее часть) в том состоянии, в каком она находилась на момент последнего копирования. Резервное копирование должно осуществляться тщательно и строго по графику. Кроме того, устройства резервного копирования и сами носители должны регулярно проверяться на предмет корректной работы.

Регулярность выполнения процедур резервирования данных, обеспечивающих целостность резервных копий, является важным элементом корпоративной стратегии. Ответственные руководители должны четко понимать, какой цели служат архивы данных и как они должны эксплуатироваться. Для университетской компьютерной лаборатории потерять результаты работы за один день не так страшно, как для торговой компании, регулярно принимающей заказы от десятков или сотен клиентов.

Начнем эту главу с описания общих принципов резервного копирования, после чего ознакомимся с наиболее распространенными устройствами создания резервных копий и их носителями (их преимуществами, недостатками, ценами на них). Затем речь пойдет о планировании процедуры резервного копирования и об особенностях работы популярных утилит **dump** и **restore**.

Далее будут рассмотрены некоторые дополнительные команды резервного копирования и архивирования и даны пояснения, в каких ситуациях предпочтительнее та или иная команда. В завершение главы читатели познакомятся с бесплатным пакетом сетевого резервного копирования Bacula и некоторыми другими свободно предоставляемыми и коммерческими вариантами.

# 10.1. Принципы резервного копирования

Прежде чем приступить к подробному рассмотрению резервного копирования, мы хотели бы поделиться с читателями опытом, который приобрели за долгие годы практической работы (иногда эти знания давались очень тяжело). Ни один из предлагаемых советов не является абсолютной истиной, но чем чаще вы будете им следовать, тем спокойнее будут протекать процессы резервного копирования данных и их восстановления.

## Создавайте резервные копии на центральном компьютере

Многие специализированные утилиты позволяют создавать резервные копии по сети. Производительность процесса несколько ухудшается, но это компенсируется легкостью администрирования. Если вы управляете только несколькими серверами, то проще всего выполнять сценарий на центральном компьютере, который запускает утилиту **dump** (посредством интерпретатора **ssh**) на всех компьютерах, где необходимо провести резервное копирование. Если у вас много серверов, мы рекомендуем использовать пакет программ (коммерческий или бесплатный), автоматизирующий этот процесс. Все архивы должны помещаться на одно устройство резервного копирования (в режиме без перемотки, естественно).

Если размер системы слишком велик и одного сервера недостаточно, все равно следует пытаться сделать систему резервного копирования как можно более централизованной. Это упростит администрирование и позволит восстанавливать данные на альтернативных серверах. Часто на сервере можно установить несколько устройств резервного копирования, и это не повлияет на его производительность.

Архивы, созданные с помощью утилиты **dump**, можно восстановить только на компьютерах с таким же порядком следования байтов, как у ведущего узла (и в большинстве случаев только на компьютерах с такой же операционной системой). Для решения задачи, связанной с простой перестановкой байтов, можно использовать программу **dd**, но она не устранит проблемы несовместимости различных версий утилиты **dump**.

Если вы резервируете большие объемы данных по сети, пропускная способность которой вызывает сомнения, рассмотрите как вариант создание выделенной локальной сети (LAN), предназначенной для передачи резервируемых данных. Многие организации уже оценили эффективность такого подхода, позволяющего преодолевать узкие места в сетях.

## Маркируйте носители

Маркируйте носители с резервными копиями системы максимально разборчиво и полно — неподписанная лента, по существу, бесполезна. Маркируйте каждый носитель уникальным образом в соответствии с его содержимым. На коробку следует нанести подробную информацию, включающую, например, список файловых систем, дату создания архивов, формат резервирования, точный синтаксис команд, с помощью которых

они были созданы, и другие сведения, необходимые для восстановления архива без обращения к документации.

Существуют бесплатные и коммерческие программы создания наклеек для различных носителей. Советуем приобрести одну из них, дабы избавиться от ненужной головной боли. Например, изготовители наклеек, предназначенных для печати на лазерных принтерах, обычно поставляют с ними и шаблоны для генерирования наклеек. Еще лучше приобрести специализированный принтер наклеек. Такие принтеры недорогие и прекрасно работают.

Ваша автоматизированная система архивирования должна регистрировать имя каждой файловой системы, которую она заархивировала. Если вы захотите восстановить некоторый файл, то хорошо организованная регистрация позволит вам быстро перейти к нужной файловой системе. При этом очень важно зафиксировать порядок размещения файловых систем на магнитной ленте или кассете.

По возможности приобретите механизм автоматической смены носителя или накопитель на магнитной ленте (НМЛ), который считывает штрих-коды. Тогда вы будете уверены, что ваши электронные метки магнитной ленты всегда будут совпадать с физическими.

## Правильно выбирайте периодичность резервного копирования

Чем чаще выполняется резервное копирование, тем меньше данных будет потеряно в случае сбоя системы. Процесс резервного копирования, однако, достаточно трудоемкий и ресурсоемкий. Системный администратор должен обеспечить приемлемую целостность данных при разумном уровне затрат времени и средств. В общем можно сказать, что расходы увеличиваются по мере использования средств восстановления с более мелкой модульностью.

В интенсивно эксплуатируемых системах наиболее удобный вариант — создавать резервные копии домашних каталогов пользователей каждый рабочий день. В системах, которые используются менее интенсивно или в которых данные изменяются не столь часто, можно делать копии несколько раз в неделю. В маленькой системе с одним пользователем достаточно делать резервную копию раз в неделю. В целом, периодичность определяется объемом данных, которые могут быть утеряны с момента последнего создания резервной копии.

## Будьте осмотрительны при выборе архивируемых файловых систем

Файловые системы, которые изменяются редко, не нужно архивировать столь же часто, как домашние каталоги пользователей. Если в файловой системе изменяется лишь несколько файлов (например, файл **/etc/passwd** в корневой системе), их можно каждый день копировать в другой раздел, резервные копии которого создаются регулярно.

Если каталог **/tmp** является отдельной файловой системой, то архивировать его не нужно. Он не содержит ничего существенного, поэтому нет оснований сохранять его. Если это вам кажется очевидным, то вас можно только поздравить (чего не скажешь о многих других организациях и их сотрудниках).

## Старайтесь умещать каждодневные архивы на одном носителе

🕮 Демон `cron` подробно описан в главе 9.

Было бы прекрасно иметь возможность создавать ежедневные резервные копии всех пользовательских файловых систем на одной ленте. Впрочем, при наличии устройств большой емкости, таких как накопители DLT, AIT и LTO, это вполне реально, но не каждая организация решится приобрести столь дорогие устройства. По мере того как привычки пользователей изменяются, а удаленный обмен данными становится все более популярным, "подходящий" временной интервал резервного копирования сокращается. Все больше и больше сетевых служб должно быть задействовано в течение суток, а на резервирование больших объемов данных тратится много времени.

Еще одна существенная проблема — быстрое увеличение дискового пространства, связанное со снижением цен на жесткие диски. Теперь вряд ли удастся найти настольный компьютер фабричной сборки, имеющий меньше 250 Гбайт дискового пространства. Зачем очищать диски и устанавливать квоты, когда с помощью небольшого капиталовложения можно безболезненно решить проблему? К сожалению, слишком часто объем сетевых хранилищ данных делает полное резервирование невозможным.

Утилиты резервирования прекрасно справляются с архивированием файловых систем на нескольких носителях. Но если архив занимает много резервных носителей (например, кассет), об их смене должен заботиться оператор или специальный робот, причем все носители должны быть снабжены наклейками, что впоследствии обеспечит нормальный процесс восстановления. Если у вас нет веской причины создавать большую файловую систему, то лучше не делайте этого.

Если ежедневные архивы невозможно разместить на одной ленте, есть несколько решений:

- купите ленточное устройство более высокой емкости;
- купите автозагрузчик или ленточную библиотеку, чтобы можно было вставлять несколько носителей в одно устройство;
- измените периодичность создания резервных копий;
- используйте несколько устройств резервного копирования.

## Храните носители вне рабочего помещения

Прежде всего, вы должны всегда иметь автономную копию своих данных, т.е. защищенную копию, которая не будет храниться вместе с оригиналом на жестком диске того же компьютера. Настоящее резервирование не заменят никакие моментальные снимки и дисковые массивы (RAID arrays)!

В большинстве организаций резервные копии хранят вне рабочего помещения во избежание их порчи вместе с основными данными в случае пожара или стихийного бедствия. Фраза "вне рабочего помещения" означает как сейф в банке, так и полку в доме у президента компании. Фирмы, специализирующиеся на безопасном хранении резервных носителей, обеспечивают для них надлежащий температурный режим и гарантируют их сохранность. Всегда следует выбирать фирму, пользующуюся хорошей репутацией. Существует множество фирм, специализирующихся на хранении данных, которые обеспечивают передачу данных по сети, но хранят их в надежном месте.

Выбор внешнего хранилища определяется тем, как часто нужно восстанавливать файлы и какие потери времени при этом допустимы. В некоторых организациях создают по две резервные копии в день на разных носителях: одна остается в рабочем помещении, а вторую сразу же выносят[1].

## Защищайте резервные копии

Дэн Гир (Dan Geer), специалист по вопросам безопасности, однажды сказал: "Что делает резервная копия? Гарантированно нарушает права доступа к файлам на расстоянии". Ни больше, ни меньше!

Защищайте свои резервные носители. Один из способов защиты — шифрование резервного носителя. Это сравнительно простой процесс, который тем не менее требует соблюдения таких стандартов обеспечения защиты, как PCI DSS (Payment Card Industry Data Security Standard — стандарт защиты информации в индустрии платежных карт, разработанный международными платежными системами Visa и MasterCard). Существует множество утилит резервного копирования, которые облегчают жизнь системному администратору. Вместе с тем, вам следует обеспечить сохранность и целостность ключей шифрования, а также их доступность при возникновении аварийных ситуаций.

Резервные носители необходимо защищать и на физическом уровне. Носители нужно хранить не просто вне помещения, а под замком, ключ от которого надежно спрятан. Если соответствующие услуги предоставляет коммерческая компания, она должна гарантировать конфиденциальность лент.

Некоторые компании настолько заботятся о сохранности резервных копий, что даже создают их дубликаты.

## Активность файловой системы во время создания архива должна быть низкой

Снятие резервных копий лучше проводить в периоды низкой активности системы, потому что изменения файлов могут привести к ошибкам в работе команды **dump**. Архивы следует создавать, когда в системе работает мало пользователей (ночью или в выходные дни). Чтобы автоматизировать этот процесс, устанавливайте свои резервные носители каждый день перед уходом домой и позвольте демону **cron** выполнять эту работу за вас. В этом случае архивирование будет происходить при минимальной вероятности изменения файлов, а сам процесс архивирования будет оказывать на пользователей минимальное влияние.

В наши дни практически невозможно выполнить резервное копирование при полном отсутствии каких-либо дисковых операций. Пользователям зачастую нужен постоянный и круглосуточный доступ к системе, службы работают сутки напролет, а для резервирования баз данных необходимо использовать специальные процедуры. В большинстве случаев работу баз данных нужно временно приостанавливать или переводить в специальный "ослабленный" режим (с ухудшенными характеристиками), чтобы утилиты резервного копирования могли аккуратно собирать данные в определенное время суток. Узлы с большими объемами данных могут и не "выдержать" время простоя, необходимое для выполнения традиционных операций резервирования их базы данных. В такие дни оста-

[1] Крупная финансовая компания, размещавшаяся во Всемирном торговом центре, хранила свои "внешние" резервные копии одним или двумя этажами ниже своих офисных помещений. Когда здание подверглось первой же террористической атаке, резервные ленты (как и сами компьютеры) были уничтожены. Поэтому "внешнее" хранилище должно быть действительно внешним.

ется только один способ обеспечить архивирование данных в отсутствие дисковой активности — сначала создавать "снимок", т.е. копию состояния каждого объекта (snapshot).

О сетях хранения данных (storage area network — SAN) рассказывается в разделе 8.11.

В большинстве контроллеров SAN (и во всех наших примерах операционных систем) предусмотрен тот или иной способ создания "снимка" файловой системы. Это средство позволяет относительно безопасно создавать резервные копии любой активной файловой системы (даже при открытых файлах). В Linux "снимки" создаются с помощью менеджера логических томов (см. раздел 8.8), а в остальных примерах операционных систем — посредством файловой системы.

"Снимки" можно создавать практически мгновенно благодаря "интеллектуальной" схеме копирования при записи. В процессе создания "снимка" на самом деле никакие данные не копируются и не перемещаются. Если "снимок" существует, то изменения, вносимые в файловую систему, записываются в новые области диска. В этом случае можно поддерживать два (или больше) образа при минимальном использовании дополнительной памяти. Концептуально создание "снимков" можно сравнить с инкрементным резервным копированием, различие только в уровне процесса: не на уровне блоков, а на уровне файловой системы.

В этом контексте "снимки" представляют собой основное средство создания "реальных" резервных копий файловой системы. Но они никогда не заменяют автономные резервные копии. "Снимки" позволяют упростить резервирование баз данных, поскольку для создания "снимка" работу базы данных необходимо приостановить лишь на секунду. Позже, используя "снимок", можно выполнить более медленный процесс резервирования на магнитную ленту, никоим образом не мешая базе данных обслуживать запросы.

## Проверяйте состояние своих носителей

Известно немало ужасных историй о системных администраторах, которые не замечали проблем, связанных с созданием архивов, до тех пор, пока в системе не происходил серьезный сбой. Важными задачами администратора являются непрерывный контроль над соблюдением установленного порядка резервного копирования и проверка качества его выполнения. Ошибка со стороны оператора приводит к повреждению большего числа архивов, чем любая другая ошибка.

На первом этапе проверки следует заставить программу резервного копирования по окончании работы повторно прочесть все ленты[2]. Неплохо также просмотреть ленты на предмет наличия ожидаемого количества файлов. В идеале должна быть проконтролирована каждая лента, но это едва ли практически возможно в крупной организации, где каждый день задействуются сотни лент. Благоразумнее в подобной ситуации осуществлять выборочную проверку.

Более подробно команда **restore** рассматривается в разделе 10.4.

Часто имеет смысл создать таблицу содержимого каждой файловой системы (для этого можно использовать команду **restore -t** утилиты **dump**) и сохранить результат (в виде списков) на диске. Полученные списки нужно называть так, чтобы их имена отражали связь с соответствующими лентами, например **okra:usr.Jan.13**. База данных с такими записями помогает быстро определить, на каком носителе находится пропавший файл. Для этого достаточно выполнить команду **grep** с именем файла в качестве аргумента и выбрать самый последний его экземпляр.

[2] Чтобы сравнить содержимое ленты с исходным деревом каталогов, в GNU-версиях можно использовать команду **restore -C**.

Успешное чтение таблицы содержимого свидетельствует о том, что архив создан нормально и что в случае необходимости наверняка можно будет восстановить файлы с носителя. Пробное восстановление файла, выбранного случайным образом, послужит еще более надежным доказательством возможности восстановления данных с этого носителя[3].

Периодически следует выполнять восстановление с произвольно выбранного резервного носителя, чтобы убедиться в том, что оно по-прежнему возможно. Иногда стоит читать файлы со старых резервных лент (месячной и даже годичной давности)[4], ведь у накопителей временами нарушается центровка, и они утрачивают способность читать ими же записанные ленты. Некоторые компании выполняют восстановление старых носителей, но их услуги стоят достаточно дорого.

Еще один вид проверки — попробовать восстановить данные с носителя в другой аппаратной среде. Если машинный зал сгорит, пользы от осознания того, что резервную копию можно было бы прочитать на расплавившемся накопителе, будет мало. В прошлом цифровые аудиокассеты особенно часто страдали от этой проблемы, но теперь технология усовершенствована и носители стали значительно более надежными.

## Определите жизненный цикл носителя

Срок службы всех носителей ограничен. Их можно и нужно использовать повторно, но не забывайте о предельном сроке эксплуатации, определяемом изготовителем носителя. Как правило, этот срок задается в виде количества проходов, которые способна выдержать лента. Каждая такая операция, как архивирование, восстановление и команда **`mt fsf`** (перемотка на один файл вперед), представляет собой один проход. Неленточные носители обладают значительно более длительным сроком службы, который иногда выражают *средним временем до отказа* (mean-time-to-failure — MTTF), но в любом случае все аппаратные средства и носители имеют ограниченный срок службы. Поэтому имеет смысл оценивать его не в реальных годах, а с точки зрения продолжительности жизни, например, собаки.

Но прежде чем выбрасывать старые ленты, не забудьте стереть с них всю информацию или же сделать ее недоступной для считывания. В этом может помочь устройство для размагничивания магнитной ленты (большой электромагнит), но его следует держать вдали от компьютеров и активных носителей. Разрезание или вытягивание ленты из кассеты — не слишком надежная защита данных, поскольку ленту сравнительно легко склеить или намотать заново. Компании, специализирующиеся на уничтожении документов, за сравнительно невысокую плату предоставляют услуги по уничтожению магнитных лент.

Если в качестве резервных носителей вы используете жесткие диски, помните, что службы восстановления дисков стоят меньше тысячи долларов. И прежде чем с вашими дисками случится нечто непоправимое, подумайте заранее о средствах, обеспечивающих безопасное удаление (см. раздел 8.5), или о команде SCSI-форматирования.

---

[3] Команда **`restore -t`** читает только таблицу содержимого архива, хранящуюся в начале ленты. При реальном восстановлении конкретного файла проверяется более крупная область носителя.

[4] К пользователям, которые просят восстановить случайно удаленные файлы, полезно относиться как к коллегам, которые просто помогают вам выполнить выборочный контроль вашей системы архивации, а не как к источнику раздражения, не способному даже позаботиться о сохранности своих файлов. Позитивное отношение к пользователям делает вашу работу более приятной как для вас самих, так и для системы в целом (поскольку такие ситуации увеличивают количество выборочных проверок).

## Компонуйте данные с учетом резервного копирования

Когда есть столь дешевые и надежные дисковые устройства, возникает соблазн отказаться от архивирования всех имеющихся данных, сосредоточив усилия лишь на критически важных файлах, тем более что многие системы заполняются практически бесконтрольно. Но если разумно распланировать схему дискового хранения, то можно существенно упростить процесс резервного копирования.

Начать необходимо с оценки потребностей.

- С данными каких типов придется иметь дело?
- Как часто будут изменяться данные различных типов?
- Насколько часто нужно выполнять резервное копирование, чтобы возможные потери причинили минимальный ущерб?
- Какие сетевые и административные ограничения должны быть определены для данных?

Используя эту информацию, следует продумать схему хранения данных с учетом резервного копирования и возможного расширения системы в будущем. Лучше размещать каталоги проектов и домашние каталоги пользователей на выделенных файловых серверах, что упростит управление данными и позволит гарантировать их безопасность.

С появлением убедительных решений по созданию образов системы зачастую проще заново сгенерировать сломанную систему, чем диагностировать неисправности и восстанавливать поврежденные или недостающие файлы. Одни администраторы конфигурируют рабочие станции пользователей в расчете на сохранение всех данных на централизованном сервере. Другие управляют группами серверов, которые характеризуются почти идентичными конфигурациями и данными (такими, как содержимое веб-сайтов занятости). В такой среде не разумно резервировать объемные массивы однотипных систем. Однако те, кто "поведен" на безопасности систем, настаивают на интенсивном создании резервных копий, чтобы в случае аварии данные всегда были доступны для судебного исследования.

## Будьте готовы к худшему

После разработки схемы резервного копирования изучите самый худший сценарий: система полностью уничтожена. Определите, сколько информации будет потеряно и сколько времени уйдет на восстановление системы (включая время на закупку нового оборудования). Затем посмотрите, удовлетворяют ли вас полученные результаты.

Более формальные организации для информации на конкретных серверах или файловых системах часто определяют такие характеристики, как *допустимое время восстановления* (Recovery Time Objective — RTO) и *допустимый уровень восстановления* (Recovery Point Objective — RPO). С помощью этих значений можно обеспечить полноценное управление системой.

Показатель RTO представляет собой максимальное время, которое бизнес может "вытерпеть" в ожидании восстановления после возникновения аварийной ситуации. Обычно для пользовательских данных значения RTO лежат в пределах от нескольких часов до нескольких дней, а для рабочих серверов — от нескольких секунд до нескольких часов.

Показатель RPO означает, какая по давности резервная копия требуется для восстановления данных, поврежденных по определенной причине (т.е. за какой срок вы

готовы потерять данные). Это значение влияет на уровень, на котором должны сохраняться резервные копии. В зависимости от того, как часто изменяется набор данных и насколько он важен, значение RPO может колебаться от недель до часов и даже секунд. Очевидно, что резервные копии на магнитной ленте не могут удовлетворять требованиям, выраженным близкими к нулю значениями RPO, и поэтому такие требования обычно предполагают наличие больших инвестиций и специализированных устройств хранения данных, расположенных в нескольких информационных центрах.

Несмотря на то что процесс определения этих показателей может показаться несколько произвольным, он все же позволяет "владельцам" данных и техническому персоналу одинаково понимать задачу сохранности данных. Этот процесс требует уравновешивания стоимости и затрат в зависимости от конкретных требований бизнеса к восстанавливаемости. Это трудная, но важная задача, которую необходимо решать.

## 10.2. Устройства и носители, используемые для резервного копирования

Поскольку многие неисправности могут приводить к одновременному выходу из строя сразу нескольких аппаратных устройств, резервные копии следует помещать на съемные носители. Поэтому важно создавать автономные резервные копии, которые не мог бы повредить ни один "самый сердитый" системный администратор.

Например, копирование содержимого одного жесткого диска на другой на одном и том же компьютере или в одном и том же центре обработки данных, конечно, лучше, чем ничего, но оно обеспечивает весьма незначительный уровень защиты на случай отказа сервера. Хотя компании, организующие резервное копирование через Интернет, становятся все более популярными, все же большинство резервных копий хранится локально.

В следующих подразделах описываются различные виды носителей для хранения резервных копий. Они перечислены в порядке возрастания емкости.

Производители устройств резервного копирования любят указывать емкость, отталкиваясь от объема сжатых данных, оптимистично предполагая коэффициент сжатия 2:1 или выше. Ниже мы будем приводить реальное число байтов, которое физически можно записать на тот или иной носитель без учета сжатия.

Коэффициент сжатия влияет также на пропускную способность устройства. Если накопитель позволяет записывать данные на ленту со скоростью 1 Мбайт/с, а производитель заявляет о коэффициенте 2:1, то пропускная способность волшебным образом возрастает до 2 Мбайт/с. Как и в случае емкости, мы проигнорируем рекламную скорость записи и будем учитывать реальные показатели.

### Оптические носители: CD-R/RW, DVD±R/RW, DVD-RAM и Blu-ray

При цене около 30 центов за штуку, диски CD и DVD хорошо подходят для резервного копирования небольших, изолированных систем. Емкость компакт-дисков CD составляет около 700 Мбайт, а дисков DVD — около 4,7 Гбайт. Емкость двухслойных DVD доходит до 8,5 Гбайт.

Дисководы с возможностью записи на эти носители выпускаются для всех стандартных шин (SCSI, IDE, USB, SATA и другие) и во многих случаях настолько дешевы, что,

по существу, на их стоимость можно не обращать внимание. Теперь, когда цены CD и DVD практически сравнялись, нет никаких причин использовать CD, а не DVD (если, конечно, не учитывать, что устройства чтения CD все еще более распространены).

Запись на оптические носители осуществляется фотохимическим способом посредством лазерного луча. Хотя достоверных данных нет, считается, что оптические носители долговечнее, чем магнитные. Однако даже диски с однократной записью (CD-R, DVD-R и DVD+R) уступают по долговечности фабричным (штампованным) компакт-дискам CD и DVD.

Современные записывающие приводы DVD не уступают по скорости ленточным накопителям (а некоторые даже и превосходят их). Версии DVD-R и DVD+R предназначены для одноразовой записи. Носители DVD-RW, DVD+RW и DVD-RAM допускают многократную запись. Система DVD-RAM имеет встроенные средства обработки дефектов и поэтому более надежна, чем другие типы носителей. В то же время эти носители значительно дороже остальных.

По прогнозам изготовителей потенциальный срок службы этих носителей при правильном их хранении должен составлять сотни лет. Изготовители рекомендуют хранить диски в отдельных коробках при постоянной температуре в диапазоне 41–68°F (5°–20°C) и относительной влажности 30–50% в защищенном от прямых солнечных лучей месте. Для маркировки дисков следует использовать только водорастворимые маркеры. Вероятно, в общем случае более реальным можно считать срок службы, равный 1–5 годам.

Как показывают многочисленные исследования, выполненные независимыми экспертами, надежность оптических носителей, в основном, определяется фирмой-изготовителем. Это как раз тот случай, когда приобретение первоклассных носителей полностью окупается. К сожалению, даже изделия одного и того же изготовителя различаются по качеству.

Недавно на рынке оптических носителей появились диски типа Blu-Ray, различные версии которых позволяют хранить 25–100 Гбайт данных. Их высокая емкость обеспечивается более короткой длиной волны (405 нм) лазера, используемого для чтения и записи (отсюда и название этого типа носителей "blue" — голубой). Поскольку цена на эти носители падает, технология Blu-Ray обещает стать хорошим решением для резервного копирования.

## Переносные и съемные жесткие диски

Внешние накопители, подключаемые через порты USB 2.0 или eSATA, распространены повсеместно. Обычно в качестве основного устройства хранения используется тот или иной жесткий диск, но флеш-память (вездесущие "портативные диски") находит все большее распространение. Емкости этих устройств варьируются от менее 250 Гбайт до 2 Тбайт и более. В настоящее время используются также полупроводниковые (твердотельные) диски (Solid state drives — SSD), в основе которых лежит флеш-память емкостью до 160 Гбайт. В настоящее время максимальная емкость устройств флеш-памяти встречающихся на рынке, составляет около 64 Гбайт.

В основном, долговечность устройств флеш-памяти определяется числом циклов записи. Предположительно устройства среднего класса рассчитаны на 100 000 циклов.

Основным ограничением устройств, используемых в качестве резервных носителей является то, что они обычно работают в реальном масштабе времени и поэтому весьма чувствительны к скачкам напряжения, перегреву и злонамеренным манипуляциям пользователей. Для того чтобы жесткие диски эффективно работали резервными носителя

ми, они должны быть вручную размонтированы или отсоединены от сервера. Съемные устройства облегчают эту задачу. Специализированные "безленточные" системы резервирования, которые используют диски для эмулирования автономной природы ленточных устройств, также присутствуют на рынке.

## Магнитные ленты

Многие виды носителей информации сохраняют данные за счет специального ориентирования магнитных частиц. Такие носители подвержены разрушительной силе со стороны электрических и магнитных полей. Поэтому вы должны остерегаться таких источников магнитного поля, как акустические колонки, трансформаторы, источники питания, неэкранированные электродвигатели, дисковые вентиляторы и ЭЛТ-мониторы, а также длительного воздействия фонового излучения Земли.

Все магнитные ленты рано или поздно (спустя годы) становятся нечитабельными. Большинство ленточных носителей прослужат верой и правдой по крайней мере три года, но если вы планируете хранить свои данные больше трех лет, должны использовать носитель, который сертифицирован для более долгого хранения информации, или периодически переписывать данные на другие носители.

## Малые лентопротяжные устройства: 8-миллиметровые и DDS/DAT

Различные варианты лентопротяжных устройств, как 8-миллиметровых, так и работающих в стандарте DDS (Digital Data Storage — цифровое хранение данных) и DAT (Digital Audio Tape — цифровая аудиолента), образуют дешевый сегмент рынка ленточных накопителей. Еще не так давно наибольшей популярностью пользовались 8-миллиметровые лентопротяжные устройства компании Exabyte, но в них наблюдалась тенденция к нарушению центровки каждые 6–12 месяцев, что вело к необходимости их регулировки в ремонтной мастерской. Часто лентопротяжный механизм растягивал ленты, и они становились нечитаемыми. Емкость этих лент, равная 2–7 Гбайт, делает их малопригодными для резервирования даже современных настольных систем, не говоря уже о серверах.

Накопители DDS/DAT — это устройства, в которых запись данных осуществляется на 4-миллиметровую ленту методом спиральной развертки (наклонно-строчная запись). В действительности DAT-устройства относятся к стандарту DDS, но это непринципиальное различие. Исходный формат обеспечивал емкость около 2 Гбайт, но в следующих версиях стандарта DDS емкость существенно возросла. Последняя версия (DAT 160) позволяет хранить до 80 Гбайт данных при скорости их передачи, равной 6,9 Мбайт/с. По заявлениям изготовителей ленты должны выдерживать до 100 циклов резервной записи, а их срок службы составляет до 10 лет.

## Устройства DLT/S-DLT

Устройства DLT/S-DLT (Digital Linear Tape/Super Digital Linear Tape) представляют основную разновидность носителей резервного копирования. Они надежны и позволяют хранить большие объемы данных. Их родоначальниками являются кассетные накопители TK-50 и TK-70 компании DEC. Впоследствии компания DEC продала технологию фирме Quantum, которая увеличила скорость записи и емкость носителей и снизила на них цены. В 2002 году Quantum приобрела лицензию на технологию Super DLT, разрабо-

танную компанией Benchmark Storage Innovations. При использовании этой технологии записывающая головка наклоняется вперед и назад, что позволяет снизить перекрестные искажения между соседними дорожками.

В настоящее время компания Quantum предлагает две серии устройств: одна ориентирована на достижение максимальной производительности, а вторая — на максимальную емкость. Пользователь имеет возможность выбрать то, что ему наиболее подходит. Емкости ленты варьируются от 800 Гбайт в устройствах серии DLT-4 до 160 Гбайт в устройствах DLT-4 серии повышенной емкости при скоростях передачи данных 60 и 10 Мбайт/с соответственно. По заявлениям производителей ленты прослужат от 20 до 30 лет. Но будут ли к тому времени существовать устройства, способные их прочесть? Много ли вам известно устройств чтения 9-дорожечных лент, которые до сих пор функционируют?

Недостатком технологии S-DLT является цена носителей, которая достигает 90–100 долларов (для 800-гигабайтовых лент). Для какой-нибудь инвестиционной компании с Уолл-стрит это, может быть, нормальная цена, но для университета она несколько высоковата.

## Устройства AIT и SAIT

Технология AIT (Advanced Intelligent Tape — усовершенствованная лента со встроенной логикой) — это линия усовершенствованных устройств записи на 8-миллиметровую ленту, выпускаемых компанией Sony. В 1996 году компания отказалась от поддержки устройств Exabyte и представила собственный стандарт AIT-1, в котором также использовался метод наклонно-строчной записи (спиральная развертка), но емкость накопителей была в два раза выше. В настоящее время Sony предлагает версию AIT-4 (с емкостью кассеты 200 Гбайт при максимальной скорости передачи 24 Мбайт/с) и свою последнюю разработку AIT-5 (с удвоенной емкостью кассеты при той же максимальной скорости передачи).

Предлагаемые компанией Sony устройства SAIT половинной высоты используют более широкий носитель и обладают большей емкостью, чем AIT. Ленты SAIT позволяют хранить до 500 Гбайт данных при скорости их передачи 30 Мбайт/с. Эти устройства наиболее популярны в качестве ленточных библиотек.

В AIT- и SAIT-устройствах используются ленты AME (Advanced Metal Evaporated — усовершенствованная технология напыления металла) с большим сроком эксплуатации. В ленточных картриджах имеется встроенное ЭСППЗУ (электрически стираемое программируемое ПЗУ), содержащее микрокод носителя, но для использования этого микрокода требуется программная поддержка. Цена устройств и лент сопоставима с ценой DLT-аналогов.

## Устройства VXA/VXA-X

В 2006 году компания Tandberg Data купила технологии VXA и VXA-X, разработанные компанией Exabyte. В лентопротяжных устройствах VXA использована технология, которая в компании Exabyte получила название пакетной технологии передачи данных. В устройствах VXA-X по-прежнему используются ленты AME производства компании Sony. Емкость устройств серии V возрастает с появлением носителей более высокой емкости. Заявленная емкость устройств серий VXA и X колеблется в диапазоне 33–160 Гбайт при скорости передачи 24 Мбайт/с.

## Устройства LTO

Открытый стандарт ленты с линейной записью (Linear Tape-Open — LTO) был разработан компаниями IBM, HP и Quantum в качестве альтернативы запатентованному формату DLT. Последняя версия этого формата, LTO-4, позволяет хранить 800 Гбайт данных при скорости передачи 120 Мбайт/с. Предполагаемый срок эксплуатации лент LTO — 30 лет, но они чувствительны к влиянию магнитных полей. Устройства предыдущего поколения (LTO-3) продаются гораздо дешевле, что вполне ожидаемо, и при этом они вполне пригодны для использования во многих средах. Цена картриджей формата LTO-3 (емкостью 400 Гбайт) составляет около $25, а формата LTO-4 — $40.

## Системы с автоматической загрузкой носителей (автозагрузчики, ленточные массивы и библиотеки)

Стоимость современных жестких дисков столь мала, что многие организации могут позволить себе использовать диски огромной емкости. Поэтому для резервного копирования всех имеющихся данных иногда требуется несколько лент, даже если емкость каждой из них составляет 800 Гбайт. В таких случаях можно порекомендовать приобрести автозагрузчик, ленточный массив или библиотеку.

*Автозагрузчик* — это простое устройство для автоматической смены лент, используемое со стандартным накопителем. Он оснащен съемным магазином, в который помещаются ленты. Заполненная лента изымается из накопителя и заменяется пустой лентой, которую автозагрузчик берет из магазина. В магазинах автозагрузчиков обычно помещается до десяти лент.

*Ленточный массив* — это устройство, которое автоматически меняет съемные носители в нескольких накопителях. Выпускаются массивы для различных типов носителей. Часто они поставляются вместе со специальными программами создания резервных копий, которые манипулируют устройством смены носителей. Такие программные продукты выпускаются, в частности, компаниями Storage Technology (теперь это часть Oracle) и Sony.

*Ленточные библиотеки* (также известные как *библиотекари с автоподачей картриджей*) — это устройства для хранения огромных (терабайтовых) объемов данных. Они представляют собой аппаратные комплексы размером со шкаф, в которых имеется манипулятор типа "рука", обслуживающий многочисленные полки с ленточными носителями или оптическими приводами. Несложно догадаться, что эти устройства очень дороги как при покупке, так и в обслуживании, ведь им требуется специальное электропитание, помещение и особый режим вентиляции.

Как правило, вместе с библиотекой заказываются услуги оператора, который отвечает за установку и наладку комплекса. С библиотекой поставляется также программный пакет, управляющий ее работой. Ведущими производителями ленточных библиотек являются компании Storage Technology (Oracle), Spectra Logic и HP.

## Жесткие диски

Стремительное снижение стоимости жестких дисков делает их вполне достойными кандидатами на роль устройств резервного копирования. И хотя мы не рекомендуем копировать один диск на другой на том же самом компьютере, можно создавать резервные копии по сети; их стоимость при этом будет очень невелика, а времени, требуемого для восстановления больших наборов данных, уйдет достаточно мало.

Конечно, емкость жесткого диска ограничена, и со временем он будет занят полностью. Однако резервные копии на жестких дисках — прекрасное средство защиты от случайного удаления файлов. Когда есть вчерашний образ диска, доступный по сети через NFS или CIFS, пользователи получают возможность исправлять свои ошибки без вмешательства администратора.

Помните, что оперативная память, как правило, недостаточно защищена от злонамеренных взломщиков или отказов оборудования информационных центров. Если вы не можете сохранять свои резервные копии автономно, то, по крайней мере, подумайте об их географической диверсификации.

## Интернет и службы облачного резервирования данных

В последнее время поставщики услуг начали предлагать решения, обеспечивающие сохранение данных в Интернете. Вместо использования хранилища в собственном информационном центре, вы арендуете хранилище у так называемого облачного поставщика услуг (cloud provider). Этот подход не только обеспечивает доступ по запросу к хранилищу практически неограниченного размера, но и предоставляет простой способ сохранения данных в нескольких географических местах.

Цены на услуги интернет-служб хранения информации составляют от 10 центов за гигабайт в месяц и возрастают по мере добавления функциональных возможностей. Например, некоторые провайдеры позволяют выбрать количество сохраняемых резервных копий данных. Такая оплата по факту потребления позволяет вам выбрать уровень надежности, соответствующий вашим данным и бюджету.

Интернет-резервирование действует только в случае, если ваше интернет-соединение достаточно быстрое для передачи копий изменений в течение ночи (т.е. без погрязания в "реальном" трафике). Если в вашей организации обрабатываются очень большие объемы данных, то вряд ли вы сможете резервировать их с помощью Интернета. Но для небольших организаций "облачное" резервирование может оказаться идеальным решением, поскольку оно не требует никакой предоплаты и покупки оборудования. Однако следует помнить, что любые данные, передаваемые через Интернет или сохраняемые в "облачных" копиях, должны быть зашифрованы.

## Типы носителей

Как мы видим, существует множество возможностей. В табл. 10.1 приведены характеристики рассмотренных выше носителей и соответствующих им устройств.

**Таблица 10.1. Сравнительные характеристики носителей, применяемых для резервного копирования**

| Носитель | Емкость*, Гбайт | Скорость*, Мбайт/с | Цена накопителя, долл. | Цена носителя, долл. | Цена в расчете на 1 Гбайт, долл. | Многократное использование? | Произвольный доступ? |
|---|---|---|---|---|---|---|---|
| CD-R | 0,7 | 7 | 15 | 0,15 | 0,21 | Нет | Да |
| CD-RW | 0,7 | 4 | 20 | 0,3 | 0,42 | Да | Да |
| DVD±R | 4,7 | 30 | 30 | 0,3 | 0,06 | Нет | Да |
| DVD+R DL[a] | 8.5 | 30 | 30 | 1 | 0,12 | Нет | Да |
| DVD±RW | 4,7 | 10 | 30 | 0,4 | 0,09 | Да | Да |

*Окончание табл. 10.1*

| Носитель | Емкость[а], Гбайт | Скорость[а], Мбайт/с | Цена накопителя, долл. | Цена носителя, долл. | Цена в расчете на 1 Гбайт, долл. | Многократное использование? | Произвольный доступ? |
|---|---|---|---|---|---|---|---|
| Blu-ray | 25 | 30 | 100 | 3 | 0,12 | Нет | Да |
| DDS-4 (4 мм) | 20 | 30 | 100 | 5 | 0,25 | Да | Нет |
| DLT/S-DLT | 160 | 16 | 500 | 10 | 0,06 | Да | Нет |
| DLT-S4 | 800 | 60 | 2500 | 100 | 0,13 | Да | Нет |
| AIT-4 (8 мм) | 200 | 24 | 1200 | 40 | 0,2 | Да | Нет |
| AIT-5 | 400 | 24 | 2500 | 50 | 0,13 | Да | Нет |
| VXA-320 | 160 | 12 | 800 | 60 | 0,38 | Да | Нет |
| LTO-3 | 400 | 80 | 200 | 25 | 0,06 | Да | Нет |
| LTO-4 | 800 | 120 | 1600 | 40 | 0,05 | Да | Нет |

[а] Емкость и скорость указаны без учета сжатия данных.

[б] Допускает произвольный доступ к любому фрагменту данных.

[в] Двухслойный.

## Что покупать

В табл. 10.1 достаточно точно отражено состояние рынка оборудования для выполнения резервного копирования. Все устройства резервного копирования работают достаточно хорошо, и среди систем одной ценовой категории обычно трудно отдать какой-либо предпочтение. Покупайте устройство, которое отвечает требованиям вашей организации и имеет приемлемую цену. Если вы разрабатываете новое оборудование, убедитесь в том, что оно поддерживается вашей операционной системой и программными средствами резервирования данных.

Несмотря на то что цена и емкость носителя — важные факторы, не менее важно учесть и пропускную способность. Со скоростными носителями приятнее иметь дело, но вы должны внимательно отнестись к покупке накопителя на магнитной ленте, чтобы он не перегружал сервер. Если сервер не в состоянии передавать данные на накопитель с приемлемой скоростью, нужно останавливать процесс записи на накопитель на время, пока сервер его не "догонит". Конечно, низкие скорости никого не могут устроить, но и слишком высокие тоже вам ни к чему.

Выбирайте резервный носитель в соответствии с объемами ваших данных. Нет смысла тратить деньги на устройства DLT-S4, если вам необходимо защищать лишь несколько сотен гигабайт данных. Ваш накопитель в этом случае будет оставаться полупустым.

Оптические носители, устройства DDS и LTO лучше всего подходят для небольших рабочих групп и отдельных компьютеров, в которых установлены диски высокой емкости. Начальная цена этих систем относительно умеренна, а их носители широко распространены, к тому же за каждым стандартом стоит целый ряд производителей. Все системы обладают достаточным быстродействием, что позволяет архивировать большие объемы данных за разумное время.

Устройства DLT, AIT и LTO имеют сопоставимые характеристики. Трудно сделать выбор в пользу одной из этих технологий, тем более что ситуация может измениться с появлением новых спецификаций и моделей устройств. Все эти форматы тщательно про-

думаны и могут быть успешно интегрированы в любую систему — будь то среда учебного заведения или среда крупной коммерческой организации.

В следующих разделах базовый термин "лента" используется в качестве универсального обозначения носителей, выбранных для резервного копирования. Примеры команд, выполняющих резервное копирование, даются в контексте ленточных накопителей.

## 10.3. Экономия пространства и времени с помощью инкрементного архивирования

Почти все инструменты резервирования поддерживают, по крайней мере, два разных вида архивирования: полное резервное (страховое) копирование и инкрементное. Полное копирование включает все содержимое файловой системы, а инкрементное — только файлы, которые были изменены с момента предыдущего резервирования. Инкрементное резервное копирование способствует минимизации объема ленточного накопителя и пропускной способности сети, требуемых для ежедневного создания резервных копий. Поскольку большинство файлов никогда не меняется, даже в простейшей схеме инкрементного архивирования многие файлы не включаются в ежедневный архив.

Многие инструменты резервирования поддерживают дополнительные виды архивирования (помимо основных процедур выполнения полного и инкрементного резервного копирования). Стоит отметить, что все эти дополнения являются более сложными вариациями на тему инкрементного копирования. Единственный способ уменьшить объем резервируемых данных — воспользоваться тем, что многие данные уже сохранены на какой-нибудь архивной ленте.

Существуют программы резервирования, которые распознают идентичные копии данных, даже если они находятся в различных файлах на различных компьютерах. При этом гарантируется, что на ленту записывается только одна копия данных. Эта функция называется *дедупликацией* (разделение на несколько частей), и она может быть очень полезной для ограничения размера резервных копий.

Выбор схемы зависит от следующих факторов:

- активность файловых систем;
- емкость устройства резервного копирования;
- необходимая степень избыточности;
- число лент, которые нужно приобрести.

При создании архива ему присваивается номер уровня (целое число от 0 до 9). В архив уровня *N* копируются все файлы, которые изменились с момента создания последнего архива уровня ниже *N*. В архив нулевого уровня включается вся файловая система. В режиме инкрементного архивирования файловую систему можно вернуть в то состояние, в котором она находилась в момент последнего резервного копирования, хотя для этого может потребоваться восстановление файлов из нескольких лент[5].

Ранее команда **dump** поддерживала уровни 0–9, но более новые ее версии поддерживают тысячи уровней архивирования. При введении дополнительных уровней небольшое число регулярно изменяющихся файлов разбивается на более мелкие сегменты. Сложная схема резервного копирования обеспечивает следующие преимущества:

[5] Большинство версий команды **dump** не отслеживает, какие файлы были удалены с момента создания архива. При восстановлении информации из инкрементных резервных копий удаленные файлы снова появятся в системе.

- данные будут архивироваться чаще, что позволит снизить возможные потери;
- ежедневно будет использоваться меньшее число лент (или же все поместится на одну ленту);
- в целях защиты от ошибок будет создаваться несколько копий каждого файла;
- пропускная способность сети и время, необходимое для резервирования, существенно сокращаются.

Эти преимущества необходимо соизмерить со сложностью, связанной с поддержкой системы и восстановлением файлов. Мы опишем несколько возможных схем и приведем аргументы в пользу каждой из них. Иногда можно использовать эти схемы без изменений, а иногда приходится разрабатывать совершенно другие схемы.

## Простая схема

Если общий размер дискового пространства меньше емкости ленты, можно предложить простой график архивирования. Архивы нулевого уровня каждой файловой системы должны создаваться ежедневно. Используйте многократно один набор лент, но через каждые *N* дней (где *N* определяется потребностями организации) откладывайте ленты навсегда. Стоимость такой схемы составит (365/*N*) * (цена ленты) в год. При создании очередного архива нельзя использовать предыдущую ленту. Лучше осуществлять ротацию лент, чтобы в случае уничтожения одного из архивов можно было восстановить архив предыдущего дня.

Подобная схема обеспечивает высокую избыточность и значительно упрощает восстановление данных. Она хорошо подходит для организации, имеющей много денег, но очень занятого (или неопытного) администратора. С точки зрения надежности и удобства это идеальная схема. От нее не следует отступать без особой на то причины (например, в целях экономии лент или снижения трудоемкости).

## Умеренная схема

В большинстве организаций придерживаются более разумного подхода: выделяется по одной ленте на каждый день недели, каждую неделю месяца (понадобится пять таких лент) и каждый месяц года. Ежедневно создается дневной архив девятого уровня, еженедельно — недельный архив пятого уровня, ежемесячно — месячный архив третьего уровня. Архив нулевого уровня нужно создавать всякий раз, когда инкрементные копии становятся слишком большими для одной ленты. Чаще всего это происходит с лентами месячного архива. В любом случае архивирование нулевого уровня необходимо проводить хотя бы раз в год.

Выбор уровней 3, 5 и 9 сделан произвольно. С таким же успехом можно использовать уровни 1, 2 и 3, но интервалы между ними дают определенную свободу для маневра на случай, если впоследствии потребуется добавить еще один уровень. (В других программах резервирования вместо уровней, выраженных числами, используются подходы, определяемые такими терминами, как полное, дифференциальное и инкрементное архивирование.)

Такой график требует наличия двадцати четырех основных лент, а также какого-то количества лент для архивов нулевого уровня. Общее число необходимых лент невелико, но и избыточность невысока.

## 10.4. Команда dump: настройка режима резервирования

Самые распространенные программные средства создания резервных копий и восстановления данных — команды **dump** и **restore**. Они существуют уже длительное время и хорошо изучены. В большинстве организаций команды **dump** и **restore** используются автоматизированными системами резервного копирования.

Наличие команд **dump** и **restore** в системе зависит от того, какие опции были заданы при установке Linux, поэтому не исключено, что их придется инсталлировать явно. Для каждой системы существует специальный программный пакет, упрощающий процедуру установки. Текущая версия ядра системы Red Hat во время установки предлагает установить административный пакет, который включает в себя команду **dump**.

В системах Solaris команды **dump** и **restore** называются **ufsdump** и **ufsrestore** соответственно. Команда **dump** существует, но она не связана с резервированием данных. Как можно понять из названий этих команд, **ufs**-команды работают только со старыми файловыми системами UFS; они не работают в файловых системах ZFS. О вариантах ZFS-резервирования написано в разделе 10.6.

Команда **ufsdump** принимает те же флаги и аргументы, что и традиционная команда **dump** в других системах, но обрабатывает аргументы по-другому. Команда **ufsdump** ожидает приема всех флагов, содержащихся в первом аргументе, и требует, чтобы аргументы флагов следовали в определенном порядке. Например, в случае, когда большинство команд принимает последовательность аргументов **-a 5 -b -c 10**, Solaris-команда **ufsdump** примет такую последовательность: **abc 5 10**.

Предполагается, что команда **ufsdump** используется только в немонтируемых файловых системах. Если вам нужно выполнить резервирование активной файловой системы, запустите сначала Solaris-команду **fssnap**, а затем для получения "снимка" системы выполните команду **ufsdump**.

В системе AIX команда **dump** называется **backup**, хотя команда **restore** сохранила прежнее имя **restore**. Команда **dump** существует, но она не связана с резервированием данных.

Для простоты мы будем называть команды резервирования "общими" именами **dump** и **restore**, отображая их традиционные флаги командной строки. Несмотря на другие названия в некоторых системах, эти команды функционируют аналогичным образом. Учитывая важность надежного архивирования, имеет смысл проверить назначение флагов в man-страницах на компьютере, данные которого вы резервируете; поскольку многие изготовители имеют обыкновение изменять значение хотя бы одного флага.

### Архивирование файловых систем

Команда **dump** формирует перечень файлов, которые модифицировались с момента предыдущего архивирования, а затем упаковывает эти файлы в один большой архив, подлежащий записи на внешнее устройство. Команда **dump** обладает рядом преимуществ, по сравнению с другими утилитами, описанными в этой главе.

- Резервные копии могут быть записаны на несколько лент.

- Можно выполнять резервное копирование и восстановление файлов любого типа (даже файлов устройств).
- Права доступа, информация о владельцах и даты модификации файлов сохраняются.
- Обеспечивается правильная обработка файлов с "дырами"[6].
- Резервное копирование может выполняться в инкрементном режиме (на ленту записываются только модифицированные версии файлов).

GNU-версия команды **tar**, используемая в Linux, также реализует все эти возможности, но средства инкрементного архивирования у команды **dump** лучше.

К сожалению, так сложилось, что версия команды **tar**, входящая в состав большинства основных UNIX-систем, не обладает перечисленными выше особенностями GNU-версии. Если с резервными копиями нужно работать не только в Linux, но и в других UNIX-системах, то у команды **dump** нет конкурентов. Она одинаково ведет себя на разных платформах, поэтому нет смысла осваивать две команды, когда достаточно одной. Команда **dump** предоставляет меньше возможностей, но команда **tar** разборчива!

В системах Linux команда **dump** поддерживает файловые системы в расширенном составе. Возможно, для поддержки других файловых систем вам придется загрузить и инсталлировать другие версии команды **dump**.

Команда **dump** понимает структуру исходных файловых систем и непосредственно читает таблицы индексных дескрипторов, чтобы определить, какие файлы подлежат архивированию. Такое знание файловой системы делает команду **dump** весьма эффективной, но вместе с тем налагает на нее определенные ограничения[7].

🕮 Подробная информация об NFS приведена в главе 18.

Одно из ограничений заключается в том, что каждая файловая система должна архивироваться в индивидуальном порядке. Другое ограничение: разрешается копировать только файловые системы локального компьютера, а файловую систему NFS, смонтированную по сети, архивировать нельзя. Тем не менее можно создать резервную копию локальной файловой системы на удаленном ленточном накопителе[8].

Еще одна приятная особенность команды **dump** связана с тем, что она не обращает внимания на длину имен файлов. Иерархии каталогов могут быть произвольно глубокими, и длинные составные имена обрабатываются корректно.

Первым аргументом команды **dump** должен быть уровень архива. Для того чтобы определить, насколько давно создавался последний архив, команда **dump** проверяет файл **/etc/dumpdates**. Флаг **-u** заставляет команду по завершении копирования автоматически обновить файл **/etc/dumpdates**. При этом регистрируются дата, уровень архива и имя файловой системы. Если флаг **-u** ни разу не использовался, всем архивам будет присваиваться уровень 0, поскольку факт предыдущего резервного копирования файло-

[6] "Дыры" — это блоки, которые никогда не содержали данных. Если открыть файл и записать в него один байт, а затем переместить указатель текущей позиции на 1 Мбайт вперед и записать еще один байт, то полученный "разреженный" файл будет занимать всего два дисковых блока, хотя его логический размер гораздо больше. Много "дыр" обычно содержат файлы, созданные в СУБД Berkeley DB или ndbm.

[7] Команда **dump** требует доступа к неструктурированным разделам диска. Каждый, кому разрешено выполнять резервное копирование, может, приложив определенные усилия, прочесть все файлы в системе.

[8] Унаследованные системы для создания резервной копии системы на удаленном ленточном накопителе могут использовать отдельную команду **rdump**. Современные версии команды **dump** принимают аргумент **-f** *имя_узла:ленточное_устройство*.

вой системы нигде не отражался. Если имя файловой системы изменилось, можно отредактировать файл **/etc/dumpdates** вручную.

О номерах устройств рассказывается в разделе 13.2.

Свою выходную информацию команда **dump** посылает на устройство, заданное по умолчанию. Как правило, это основной ленточный накопитель. Для указания другого устройства необходимо использовать флаг **-f**. Когда на одну ленту помещается несколько архивов, задайте имя ленточного устройства, не поддерживающего режим обратной перемотки (т.е. укажите файл устройства, который не вызывает перемотку ленты по окончании записи, — для большинства ленточных устройств такой файл создается наряду со стандартным)[9]. Прочитайте man-страницу, посвященную вашему ленточному устройству, чтобы определить точное имя соответствующего файла. В табл. 10.2 приведено несколько советов для наших четырех примеров систем.

**Таблица 10.2. Файлы устройств для стандартного ленточного накопителя SCSI**

| Система | С перемоткой | Без перемотки |
|---|---|---|
| Linux | **/dev/st0** | **/dev/nst0** |
| Solaris | **/dev/rmt/0** | **/dev/rmt/0n** |
| HP-UX | **/dev/rmt/0m** | **/dev/rmt/0mn** |
| AIX | **/dev/rmt0** | **/dev/rmt0.1** |

Если случайно выбран файл устройства с обратной перемоткой, все закончится сохранением только последней заархивированной файловой системы. Поскольку команда **dump** не проверяет, перемотана лента в начало или нет, эта ошибка не повлечет за собой других ошибок и станет очевидной лишь при попытке восстановить файлы.

Чтобы заархивировать удаленную систему, удаленное устройство нужно задать в формате *имя_компьютера:устройство*.

```
$ sudo dump -0u -f anchor:/dev/nst0 /spare
```

Права доступа к удаленному ленточному устройству определяются посредством канала SSH. Подробнее о SSH рассказывается в разделе 22.10.

Раньше команде **dump** нужно было указывать точную длину ленты, чтобы запись прекращалась при достижении конца ленты. Современные накопители самостоятельно обнаруживают конец ленты и сообщают об этом команде **dump**, которая перематывает ленту в начало, выталкивает носитель из устройства и запрашивает новую ленту. Поскольку применение различных аппаратных алгоритмов сжатия не позволяет определить длину ленты на основании емкости носителя, лучше ориентироваться на сигнал конца ленты (EOT — End Of Tape).

Все версии команды **dump** воспринимают опции **-d** и **-s**, которые задают плотность записи информации на магнитную ленту (tape density) в байтах на дюйм и длину ленты в футах соответственно. Чуть более чувствительные версии позволяют указывать размеры в килобайтах (с помощью опции **-B**). Для версий, которые "не понимают" эти опции, для выражения желаемых размеров вам придется предварительно произвести некоторые арифметические вычисления.

Например, предположим, что вы хотите создать архив уровня **5** файловой системы **/work** на ленточном накопителе DDS-4 (DAT), собственная емкость которого состав-

[9] Для всех файлов ленточного устройства задан один и тот же старший номер устройства. Младший номер устройства сообщает драйверу о наличии особых свойств (режим обратной перемотки, режим перестановки байтов и т.д.).

ляет 20 Гбайт (при обычной емкости сжатых данных около 40 Гбайт). DAT-устройства могут сообщать о конце ленты (EOT), поэтому нам нужно "обмануть" команду **dump** и установить размер ленты равным значению, намного превышающему 40 Гбайт, скажем, 50 Гбайт. Тем самым мы "выжмем" около 60 000 футов при плотности 6 250 байт на дюйм.

```
# dump -5u -s 60000 -d 6250 -f /dev/nst0 /work
   DUMP: Date of this level 5 dump: Wed Nov 18 14:28:05 2009
   DUMP: Date of last level 0 dump: Sun Nov 15 21:11:05 2009
   DUMP: Dumping /dev/hda2  (/work) to /dev/nst0
   DUMP: mapping (Pass I)   [regular files]
   DUMP: mapping (Pass II)  [directories]
   DUMP: estimated 18750003 tape blocks on .23 tape(s)
   ...
```

За флагами **-5u** следуют параметры **-s** (размер: 60 000 футов), **-d** (плотность. 6 250 байт на дюйм) и **-f** (файл устройства: **/dev/nst0**). В конце указывается обязательный аргумент: имя файловой системы (**/work**). Одни версии (таких большинство) команды **dump** позволяют задавать файловую систему с помощью ее точки монтирования, как в приведенном выше примере. Другие требуют указывать исходный файл устройства.

Последняя строка результатов выполнения приведенной выше команды подтверждает, что команда **dump** не будет пытаться менять ленты по собственной инициативе, поскольку она "считает", что для данного архива понадобится только около четверти ленты. Хорошо, если количество ожидаемых лент больше 1, так как заданный размер ленты больше реального. Команда **dump** достигнет реального конца ленты (EOT) раньше, чем она достигнет собственного вычисленного предела.

## Команда restore: восстановление файлов из резервных копий

Команда, восстанавливающая данные с резервных лент, называется **restore**. Вначале рассмотрим восстановление отдельных файлов (или небольших групп файлов), а затем — файловых систем целиком.

Обычно используется динамически связанная команда **restore**, поэтому для успешной работы вам нужен доступ к библиотекам системы. Построение статически связанной версии команды **restore** потребует дополнительных усилий, но упростит восстановление системы после аварии, поскольку в этом случае команда совершенно автономна.

Первое, что нужно сделать, узнав о пропаже файла, — выяснить, на каких лентах есть его версии. Чаще всего пользователи просят найти самую последнюю версию файла, но так бывает не всегда. Например, пользователь, потерявший файл из-за случайного копирования поверх него другого файла, будет искать ту версию, которая существовала до этой неприятности. Желательно, чтобы пользователи сообщали не только о том, какие файлы пропали, но и когда они пропали и когда модифицировались последний раз.

Если не ведутся оперативные таблицы архивов, придется монтировать ленты одну за другой и пытаться восстанавливать пропавшие файлы, пока не будет найдена нужная лента. Если пользователь помнит примерную дату последнего изменения файла, можно с достаточной степенью вероятности предположить, на какой ленте он находится.

Определив ленты, с которых будет производиться восстановление, создайте временный каталог, к примеру, **/var/restore**, где будет сформирована большая иерархия подкаталогов, и перейдите в него с помощью команды **cd**. Команда **restore** обычно создает каталоги, ведущие к файлу, прежде чем восстанавливать его. Не используйте для этих

целей каталог **/tmp**: в случае системного сбоя и последующей перезагрузки содержимое каталога может быть стерто.

У команды **restore** много опций. Самые важные из них — **-i**, которая позволяет восстанавливать отдельные файлы и каталоги в интерактивном режиме, и **-r**, задающая полное восстановление всей файловой системы. Опция **-x** запрашивает автоматическое восстановление указанных файлов — будьте осторожны, чтобы не затереть существующие файлы.

Команда **restore -i** читает с ленты таблицу содержимого, а затем позволяет перемещаться по архиву, как по обычному дереву каталогов, с помощью команд **ls**, **cd** и **pwd**. Найдя файл, который нужно восстановить, выполните команду **add** *имя_файла*. Когда все необходимые файлы выбраны, скопируйте их с ленты командой **extract**.

🕮 Команда **mt** описана в разделе 10.7.

Если на одной ленте находится много архивов, то перед выполнением команды **restore** необходимо перемотать ленту на соответствующий архив с помощью команды **mt**. Не забудьте выбрать файл устройства без режима обратной перемотки!

Например, для восстановления файла **/users/janet/iamlost** с удаленного ленточного накопителя необходимо задать показанную ниже последовательность команд. Предполагается, что нужная лента найдена и смонтирована в точке **tapehost:/dev/nst0**, а файловая система, содержащая начальный каталог пользователя **janet**, — четвертая на ленте.

```
# mkdir /var/restore
# cd /var/restore
$ sudo mkdir /var/restore
$ cd /var/restore
$ sudo ssh tapehost mt -f /dev/nst0 fsf 3
$ sudo restore -i -f tapehost:/dev/nst0
restore> ls
.:
janet/  garth/  lost+found/ lynda/
restore> cd janet
restore> ls
afile bfile cfile iamlost
restore> add iamlost
restore> ls[10]
afile bfile cfile iamlost*
restore> extract
You have not read any volumes yet.
Unless you know which volume your files are on you should
start with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
```

Ленточные тома нумеруются с единицы, а не с нуля, поэтому для архива, который умещается на одной ленте, нужно указать значение 1. Спрашивая, следует ли задать имя владельца и режим доступа, команда **restore** "интересуется", должен ли текущий каталог соответствовать корневому каталогу ленты. Обычно в этом нет необходимости, если только не восстанавливается вся файловая система целиком.

После того как команда **restore** завершит работу, передайте извлеченный файл пользователю **janet**.

[10] Звездочка рядом с именем ***iamlost*** означает, что этот файл помечен для восстановления.

```
$ cd /var/restore
$ ls janet
iamlost
$ ls ~janet
afile bfile cfile
$ sudo cp -p janet/iamlost ~janet/iamlost.restored
$ sudo chown janet ~janet/iamlost.restored
$ sudo chgrp staff ~janet/iamlost.restored
$ cd /; sudo rm -rf /var/restore
$ mail janet
Your file iamlost has been restored as requested and has
been placed in /users/janet/iamlost.restored.
Your name, Humble System Administrator
```

Некоторые администраторы восстанавливают файлы в специальный каталог и дают пользователям возможность самостоятельно копировать их в свои каталоги. В этом случае необходимо обеспечить защиту файлов, назначив им соответствующих владельцев и нужные права доступа. Не забывайте очищать этот каталог, чтобы файлы не "засоряли" систему.

Если вы сначала сохранили резервную копию на удаленном ленточном накопителе и не можете из нее восстановить файлы локально, попробуйте читать ленту на том же удаленном компьютере, где она записывалась.

Команда **restore -i** — самый простой способ восстановления нескольких файлов или каталогов из архива. Но она не будет работать, если ленту нельзя перематывать по одной записи назад (такая проблема существует у некоторых накопителей на 8-миллиметровых лентах). В таком случае попробуйте использовать команду **restore -x**. Она требует указания полного имени восстанавливаемого файла (относительно корневого каталога архива) в командной строке. Следующая группа команд повторяет показанный выше пример.

```
$ sudo mkdir /var/restore
$ cd /var/restore
$ sudo ssh tapehost mt -f /dev/nst0 fsf 3
$ sudo restore -x -f tapehost:/dev/nst0 ./janet/iamlost
```

## Восстановление файловых систем

Есть счастливчики, которым никогда не приходилось после серьезного сбоя восстанавливать всю файловую систему. Другим везет меньше. Прежде чем пытаться восстанавливать файловую систему, убедитесь, что проблема, которая привела к ее разрушению, устранена. Глупо часами перематывать ленты лишь для того, чтобы тут же потерять файловую систему еще раз.

Перед началом процедуры восстановления нужно создать и смонтировать целевую файловую систему. О том, как это делается, подробно рассказывалось в главе 8. Далее перейдите в каталог монтирования новой файловой системы, вставьте в накопитель первую ленту самого последнего архива нулевого уровня и введите **restore -r**.

Команда **restore** будет сама подсказывать, когда нужно вставить следующую ленту. Восстановив архив нулевого уровня, повторяйте процедуру для инкрементных архивов в том порядке, в каком они создавались. Поскольку всегда существует определенная избыточность, то, как правило, нет необходимости восстанавливать все инкрементные архивы. Вот примерная последовательность действий.

- **Шаг 1**: восстановите самый последний архив нулевого уровня.
- **Шаг 2**: восстановите тот из оставшихся архивов, у которого наименьший уровень; если на данном уровне было создано несколько архивов, восстановите новейший из них.
- **Шаг 3**: если это оказался самый последний архив, процедуру можно считать завершенной.
- **Шаг 4**: в противном случае вернитесь к шагу 2.

Приведем несколько примеров архивных последовательностей. Восстанавливать нужно только те архивы, номера которых выделены полужирным шрифтом.

```
0 0 0 0 0 0
0 5 5 5 5
0 3 2 5 4 5
0 9 9 5 9 9 3 9 9 5 9 9
0 3 5 9 3 5 9
```

Давайте проанализируем все используемые команды. Например, если последним был создан ежемесячный архив, а перед ним создавался ежегодный архив нулевого уровня (см. раздел "Умеренная схема" выше), то для восстановления файловой системы **/home**, находящейся в логическом томе **/dev/vg01/lvol5**, понадобятся следующие команды.

```
$ sudo mkfs /dev/vg01/lvol5
$ sudo mount /dev/vg01/lvol5 /home
$ cd /home
/* Монтируем первую ленту архива уровня 0 для каталога /home. */
$ sudo restore -r
/* Монтируем ленты, запрашиваемые командой restore. */
/* Монтируем первую ленту ежемесячного архива уровня 3. */
$ sudo restore -r
```

Если на одной резервной ленте находится несколько файловых систем, то перед обращением к команде **restore** необходимо с помощью команды **mt** перемотать ленту на нужную файловую систему. Команда **mt** описана в разделе 10.7.

Приведенная последовательность команд позволит восстановить файловую систему в том состоянии, в котором она находилась, когда был создан архив третьего уровня, с одной особенностью: заодно будут "воскрешены" все удаленные с тех пор файлы. Эта проблема особенно неприятна, когда восстанавливается активная файловая система или диск практически заполнен. Не исключено, что во втором случае команда **restore** завершится неудачно[11].

## Восстановление системы на новом оборудовании

При отказе всей системы вы должны выполнить то, что называется "восстановлением системы с нуля". Прежде чем выполнить описанные выше пошаговые действия по восстановлению файловой системы, вам, как минимум, необходимо сделать следующее:

- обеспечить оборудование замены;
- инсталлировать свежую копию операционной системы;
- инсталлировать утилиты резервирования (такие, как **dump** и **restore**);

[11] Говорят, что некоторые версии команд ***dump*** и ***restore*** отслеживают "судьбу" удаляемых файлов. Будем надеяться, что это справедливо и для систем Solaris и Linux.

- сконфигурировать локальное ленточное устройство или доступ к ленточному серверу.

После этого можно переходить к процессу восстановления.

## 10.5. Архивирование и восстановление при модификации операционной системы

В процессе обновления операционной системы необходимо путем создания архива нулевого уровня получить резервные копии всех файловых систем, причем иногда их приходится сразу восстанавливать. Последнее требуется лишь в том случае, если в новой ОС используется другой формат файловой системы или изменяется структура разделов диска. В то же время резервное копирование нужно обязательно предусматривать как меру предосторожности на случай проблем, которые могут возникнуть в процессе модификации. Кроме того, наличие полного комплекта резервных копий даст возможность переинсталлировать старую операционную систему, если новая версия не подойдет. К счастью, благодаря применению систем последовательного обновления, используемых в большинстве современных дистрибутивов, эти ленты вряд ли потребуются.

Не забудьте сделать резервную копию всей системы и разделов пользователей. В зависимости от вашего пути обновления, вы можете выбрать восстановление только пользовательских данных и системно-зависимых файлов, которые расположены в корневой файловой системе или в таких **/usr**-каталогах, как **/etc/passwd**, **/etc/shadow** или **/usr/local**. Это довольно трудная задача, ибо организация каталогов, принятая в UNIX, приводит к перемешиванию файлов, созданных на компьютере, с файлами, входящими в системный дистрибутив.

Сразу после завершения модификации нужно еще раз создать полный набор архивов нулевого уровня. Процедуры модернизации, принятые в большинстве операционных систем, предполагают установку даты модификации системных файлов по времени их создания производителем системы, а не по текущему времени. Это означает, что в случае сбоя инкрементные архивы, которые были созданы относительно архива нулевого уровня, записанного до модернизации, окажутся недостаточными для восстановления системы в модернизированном состоянии.

## 10.6. Другие архиваторы

Команда **dump** является наиболее эффективным средством архивирования файловых систем, хотя это не единственная команда, которую можно использовать для записи файлов на ленты. Команды **tar** и **dd** тоже умеют перемещать файлы с одного носителя на другой.

### Команда tar: упаковка файлов

Команда **tar** объединяет несколько файлов или каталогов в один файл, часто записываемый прямо на ленту. Это удобный инструмент создания резервных копий файлов, которые предполагается восстанавливать в ближайшем будущем. Например, если у вас есть несколько старых файлов, а в системе мало места на диске, вы можете воспользоваться командой **tar** и перенести эти файлы на ленту, после чего удалить их с диска.

Команда **tar** хорошо подходит для перемещения дерева каталогов, особенно если файлы копируются от имени пользователя **root**. Команда **tar** сохраняет информацию о

принадлежности объектов и времени их создания, но только если это указано в ее параметрах. Например, команда

```
sudo tar -cf - исходный_каталог | ( cd целевой_каталог ; sudo tar -xpf - )
```

создает копию дерева исходного каталога в целевом каталоге. В аргументе *целевой_каталог* следует избегать использования символов "..", поскольку символьные ссылки и команды автоматического монтирования могут трактовать эту последовательность несколько иначе. Нам доводилось сталкиваться с подобной ситуацией.

По умолчанию большинство версий команды **tar** не выполняет интерпретацию символьных ссылок, но может делать это по указанию. Имеет смысл обратиться к man-странице этой команды, чтобы уточнить использование флага, который изменяется при переходе от системы к системе. Самый большой недостаток команды **tar** состоит в том, что ее версии, не соответствующие лицензии GNU, не позволяют использовать несколько ленточных томов. Если данные, которые вы хотите архивировать, не помещаются на одной ленте, вам, возможно, нужно обновить свою версию команды **tar**.

Для некоторых версий команды **tar** (не GNU) характерна еще одна проблема: длина имени файла по умолчанию ограничена 100 символами. Это не позволяет использовать команду для архивирования глубоких иерархий каталогов. Если в Linux создаются **tar**-архивы, которыми предполагается делиться с пользователями других систем, помните о том, что обладатели стандартной команды **tar** могут не суметь прочитать их[12].

Опция **-b** команды **tar** позволяет задать коэффициент объединения блоков, который должен учитываться при записи информации на ленту. Этот коэффициент указывается в виде числа 512-байтовых фрагментов и определяет, какой объем данных команда помещает во внутренний буфер перед выполнением записи. Некоторые DAT-устройства начинают работать некорректно, если используется нестандартный коэффициент объединения, тогда как другие накопители его игнорируют.

В ряде систем при определенных значениях коэффициента объединения блоков производительность работы с лентой повышается. Оптимальный коэффициент зависит от конкретного компьютера и ленточного накопителя, хотя во многих случаях разница в быстродействии незаметна. Если есть сомнения, задайте коэффициент 20.

Команда **tar** заполняет "дыры" в файлах и плохо справляется с наличием ошибок на лентах[13].

## Команда dd: манипулирование битами

Команда **dd** предназначена для копирования и преобразования файлов. При отсутствии иных указаний команда просто копирует информацию из входного файла в выходной. Если пользователь принес ленту, которая была записана не в UNIX, использование команды **dd** может оказаться единственным способом ее прочитать.

Одним из первоначальных применений команды **dd** было создание копии всей файловой системы. Сегодня есть более эффективный вариант: создать с помощью команды **mkfs** целевую файловую систему, а затем выполнить команду **dump**, соединив ее каналом с командой **restore**. При неправильном использовании команда **dd** может искажать информацию о структуре разделов. Она копирует файловые системы только между разделами одинакового размера.

[12] GNU-реализация в качестве одного из файлов архива включает таблицу привязки файлов по их именам. Пользователи стандартной команды **tar** могут извлечь содержимое архива и восстановить его вручную, но, конечно, этот процесс довольно утомительный.

[13] GNU-версия команды **tar** обрабатывает "дыры" только в том случае, если при создании исходного архива была указана опция **-S**.

Команду dd можно также применять для создания копий магнитных лент. При наличии двух ленточных накопителей (к примеру, **/dev/st0** и **/dev/st1**) используйте следующую команду.

```
$ dd if=/dev/st0 of=/dev/st1 cbs=16b
```

Если есть только один накопитель (**/dev/st0**), введите такую последовательность.

```
$ dd if=/dev/st0 of=tfile cbs=16b
/* Меняем ленты */
$ dd if=tfile of=/dev/st0 cbs=16b
$ rm tfile
```

Конечно, когда имеется всего один накопитель, на диске должно быть достаточно места для сохранения образа ленты.

Команда dd также является популярным инструментом у судебных специалистов. Поскольку команда dd создает побитовую (подлинную) копию тома, ее можно использовать для дублирования свидетельских показаний с последующим применением в суде.

## Резервирование файловых систем ZFS

📖 Общее представление о файловых системах ZFS можно получить в разделе 8.10.

Файловая система ZFS, первоначально разработанная для Solaris, включает интегрированные средства управления логическими томами и может функционировать как RAID-контроллер. Во многих отношениях это мечта системного администратора, но резервирование представляет собой некоторую смесь.

ZFS упрощает создание снимков файловой системы. Старые версии файловой системы доступны через каталог **.zfs**, расположенный в корневом каталоге, поэтому пользователи могут легко восстанавливать собственные файлы из прошлых снимков без участия администратора. С точки зрения оперативного управления версиями ZFS заслуживает "золотой звезды".

Однако снимки, сохраняемые на том же носителе в качестве активной файловой системы, не должны быть вашей единственной стратегией резервирования. Это мнение "разделяет" и ZFS: она включает прекрасное средство **zfs send**, которое переводит снимок файловой системы в линейный поток. Вы можете сохранить этот поток в файле или перенаправить его в удаленную систему. Этот поток можно записать на магнитную ленту. Вы можете даже переслать этот поток в удаленный процесс **zfs receive**, чтобы реплицировать файловую систему на другом компьютере (даже со всеми ее снимками и предысторией состояний системы). При желании процесс **zfs send** может выделить между двумя снимками лишь инкрементные изменения. За эту функцию мы даем еще одну "золотую звезду", а еще одну — за то, что полная документация ZFS занимает только одну-две страницы (см. man-страницу для zfs).

Нельзя не сказать и о ложке дегтя в бочке меда: процесс **zfs receive** работает только с полными файловыми системами. Чтобы восстановить несколько файлов из набора упорядоченных образов **zfs send**, придется восстановить все систему целиком, а затем выбрать из нее желаемые файлы. Будем надеяться, что у вас есть море времени и свободного места на диске и что данный ленточный накопитель не требуется для других целей. Однако у нас есть аргументы, которые могут компенсировать этот недостаток. Файловые системы ZFS характеризуются "легким весом", поэтому вы можете создавать их в большом количестве. Восстановление всего содержимого из каталога **/home** может показаться вам тяжелым случаем, но восстановление из каталога **/home/ned** — вполне

приемлемо[14]. Важнее то, что оперативная ZFS-система снимков устраняет 95% случаев, в которых вам обычно приходилось обращаться к архивной ленте.

Оперативные снимки системы не заменяют архивных лент или не сокращают частоту, с которой эти ленты должны записываться. Однако снимки действительно уменьшают частоту считывания этих лент.

## 10.7. Запись нескольких архивов на одну ленту

Магнитная лента фактически содержит одну длинную строку данных. Но очень часто возникает необходимость поместить на ленту несколько архивов, поэтому ленточные накопители и их драйверы на логическом уровне реализуют более сложную структуру хранения. Когда команда **dump** (или ее аналог) записывает поток байтов на ленту, а затем закрывает файл устройства, на ленту автоматически помещается маркер конца файла (EOF — end of file). Этот маркер отделяет один поток от другого. При извлечении потока данных чтение автоматически прекращается в случае обнаружения маркера EOF.

Чтобы перемотать ленту для записи определенного потока, используется команда **mt**. Она очень удобна при размещении на одной ленте нескольких архивов. Кроме того, ее сообщения об ошибках — одни из самых интересных среди утилит UNIX. Базовый формат команды таков.

**mt** [**-f** *имя_ленты*] *инструкция* [*счетчик*]

Аргумент *инструкция* может иметь множество значений. От платформы к платформе они меняются, поэтому мы рассмотрим лишь те варианты, которые важны для создания резервных копий и их восстановления.

**rew** Перемотать ленту к началу.

**offl** Перевести ленту в автономный режим. В большинстве накопителей это приводит к перемотке ленты в начало и выталкиванию ее из приемника. Сценарии архивирования применяют эту инструкцию для извлечения ленты по окончании операции, что должно свидетельствовать об успешном ее завершении.

**status** Вывести информацию о текущем состоянии ленточного накопителя (вставлена ли лента и т.д.).

**fsf** [*счетчик*] Перемотать ленту вперед. Если аргумент *счетчик* отсутствует, лента перематывается на один файл. Если указан числовой аргумент, лента перематывается на соответствующее количество файлов. Эту инструкцию можно использовать для перехода к нужной файловой системе на ленте, содержащей несколько архивов.

**bsf** [*счетчик*] Перемотать ленту назад на указанное число файлов. Точная интерпретация инструкции зависит от типа накопителя и его драйвера. Иногда текущий файл учитывается, иногда — нет. В некоторых случаях эта инструкция не делает ничего (и не сообщает об этом). Если лента перемотана слишком далеко, лучше всего ввести инструкцию **mt rew** и начать сначала.

[14] Применяя вариант **zfs send -R** для файловой системы **/home** и ее потомков, помните, что пока нет способа восстановливать только содержимое каталога **/home/ned**; вам придется восстанавливать все содержимое каталога **/home**. Как администратору, вам, вероятно, не захочется планировать резервное копирование для каждого домашнего каталога.

Для ознакомления с перечнем всех поддерживаемых команд обратитесь к man-странице команды **mt**.

Те, кому повезло иметь в своем распоряжении роботизированную ленточную библиотеку, могут воспользоваться пакетом **mtx**, который представляет собой улучшенную версию команды **mt** и позволяет манипулировать устройством смены лент. Мы применяли его для автоматизированного управления лентами в библиотеке картриджей Dell PowerVault LTO-3. Устройства смены лент со считывателями штрихового кода отображают даже метки магнитных лент, сканируемые с помощью **mtx**-интерфейса.

# 10.8. Программа Bacula

Bacula — это программа резервного копирования систем "клиент/сервер" на уровне организации, которая управляет резервным копированием, восстановлением и проверкой целостности файлов по сети. Компоненты сервера Bacula работают в системах Linux, Solaris и FreeBSD. Клиентская часть программы Bacula может выполнять резервное копирование других операционных систем, включая Microsoft Windows.

В предыдущем издании этой книги в качестве некоммерческого средства резервного копирования мы рекомендовали использовать программу Amanda. Информацию о ней можно найти в предыдущем издании этой книги или по адресу **amanda.org**.

Ниже приведены аргументы в пользу выбора программы Bacula.

- Имеет модульную структуру.
- Позволяет выполнять резервное копирование файловых систем UNIX, Linux, Windows и Mac OS.
- В качестве вспомогательной системы управления базой данных поддерживает MySQL, PostgreSQL и SQLite.
- Поддерживает простую в использовании, управляемую с помощью меню консоль командной строки.
- Ее исходный код соответствует лицензии программного обеспечения с открытым исходным кодом.
- Ее резервные копии могут быть записаны на несколько ленточных томов.
- Серверы этой программы могут работать на нескольких платформах.
- Для каждого копируемого файла программа может создавать сигнатуры SHA1 или MD5.
- Программа позволяет шифрование как сетевого трафика, так и данных, хранимых на ленте.
- Программа может копировать файлы, размер которых превышает 2 Гбайт.
- Поддерживает ленточные библиотеки и устройства автоматической замены лент.
- До и после выполнения задач резервного копирования может выполнять сценарии и команды.
- Осуществляет централизованное управление резервным копированием для всей сети.

## Модель, используемая программой Bacula

Для того чтобы использовать программу Bacula, необходимо понимать работу ее основных компонентов. Общая архитектура системы Bacula показана на рис. 10.1.

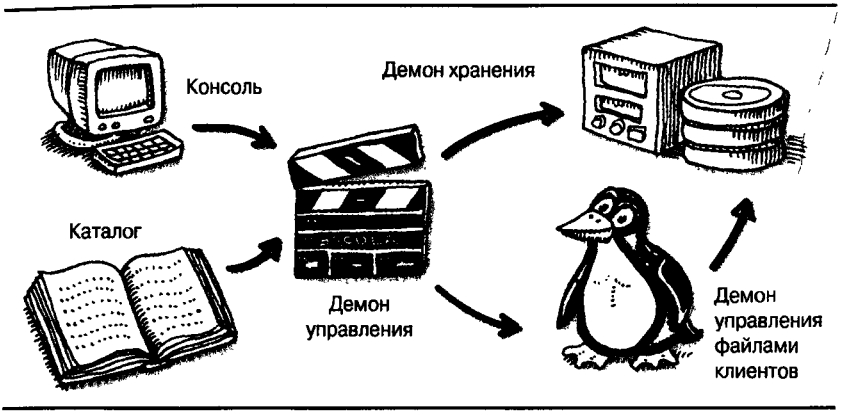



*Рис. 10.1. Компоненты системы Bacula и их взаимосвязь*

Демон управления Bacula — это демон, который координирует операции резервного копирования, восстановления и проверки. Передача заданий резервного копирования или восстановления демону управления осуществляется посредством консоли Bacula. Демон управления может также запрашивать демон хранения или демоны файловой службы, размещенные на компьютерах клиентов.

Обмен данными с демоном управления выполняется посредством консоли Bacula, которая может функционировать в качестве графического интерфейса пользователя GNOME, MS Windows либо в качестве средства командной строки. Консоль может быть запущена на любом компьютере, а не только на том, где действует демон управления.

Демон хранения — компонент системы Bacula, который выполняет чтение и запись лент и других носителей резервного копирования. Эта служба должна быть запущена на компьютере, к которому подключено ленточное устройство или устройство хранения, используемое для хранения резервных копий. Но ее установка на том же сервере, на котором установлен демон управления, не обязательна (хотя и допускается).

Демон управления файлами действует в каждой системе, файлы которой требуется копировать. Во время выполнения заданий резервного копирования реализации этого демона каждой из поддерживаемых операционных систем отправляют демону хранения данные и атрибуты соответствующих файлов.

Последний компонент системы Bacula — каталог, представляющий собой реляционную базу данных, в которой Bacula хранит информацию о каждом копируемом файле и томе. Он позволяет программе Bacula быстро и эффективно выполнять восстановление, поскольку ретроспективная информация о всех операциях резервного копирования доступна в оперативном режиме; программе известно, какие тома с резервными копиями требуются для восстановления конкретного набора файлов, еще до того, как она выполнит чтение хотя бы одной ленты. В настоящее время Bacula поддерживает три базы данных: MySQL, PostgreSQL и SQLite. Базе данных каталога не обязательно размещаться на том же сервере, на котором установлен демон управления.

Дополнительный, необязательный, компонент — Bacula Rescue CD-ROM (аварийный компакт-диск Bacula). Он представляет собой отдельно загружаемый пакет, который создает индивидуальные загрузочные компакт-диски восстановления систем Linux с нуля. Эти компакт-диски содержат статически скомпонованную копию демона файлов системы и специальный сценарий интерпретатора с информацией о конфигурации

дисков системы, ядра и сетевых интерфейсов. При возникновении фатального сбоя системы Linux ее компакт-диски аварийного восстановления позволят выполнить загрузку системы, разбиение диска на логические разделы и подключение демона управления Bacula для полного восстановления системы по сети.

## Настройка программы Bacula

Из-за сложности программы Bacula, связанной с поддержкой расширенного набора функций и модульной структурой, существует множество способов настройки схемы резервного копирования в рамках той или иной организации. В этом разделе рассмотрим основные особенности конфигурации Bacula.

В общем случае для запуска системы Bacula требуется выполнение пяти шагов:

- установка поддерживаемой внешней базы данных и демонов Bacula;
- конфигурирование демонов Bacula;
- установка и конфигурирование демонов управления файлами клиентов;
- запуск демонов Bacula;
- добавление носителей в пул носителей с помощью консоли Bacula;
- выполнение тестового резервирования и восстановления.

Мы рассмотрим минимальный вариант настройки системы, состоящей из сервера и одного или нескольких клиентов. На компьютерах-клиентах установлен только демон управления файлами. Остальные четыре компонента системы Bacula (демон управления, демон хранения, каталог и консоль) действуют на сервере. В более обширных средах рекомендуется распределять серверные Bacula-компоненты среди нескольких компьютеров, но система с минимальной конфигурацией прекрасно работает с точки зрения резервного копирования на нескольких десятках систем.

## Установка базы данных и демонов Bacula

Важно использовать одну и ту же версию версию программы Bacula в каждой системе. Раньше некоторые версии были несовместимы друг с другом.

Прежде чем установить систему Bacula, необходимо установить базу данных для хранения ее каталога. Для узлов, выполняющих резервное копирование всего нескольких систем, база данных SQLite предоставляет самый простой вариант инсталляции. Если вы резервируете больше систем, рекомендуется использовать более масштабируемую базу данных. Наш опыт использования MySQL в этой роли был положительным, и мы предполагаем использование MySQL в следующих примерах.

Стабильность и надежность — обязательные требования к платформе резервного копирования, поэтому сразу после установки базы данных рекомендуем загрузить с веб-сайта Bacula и установить самую последнюю стабильную версию исходного кода. Подробные пошаговые инструкции по установке программы приведены в каталоге **docs** пакета с исходным кодом. Документацию в форматах HTML и PDF можно также найти по адресу `bacula.org`. Там же приведены полезные учебники и руководства для разработчиков.

После распаковки исходного кода необходимо выполнить команду

```
./configure --with-mysql,
```

а затем команды **make** для компиляции двоичных файлов и **make install**, чтобы закончить инсталляцию.

По завершении установки программы Bacula необходимо создать базу данных MySQL и таблицы данных внутри нее. Bacula включает три сценария интерпретатора, которые подготавливают MySQL к сохранению каталога. Сценарий **grant_mysql_privileges** устанавливает соответствующие MySQL-разрешения для пользователя Bacula. Сценарий **create_mysql_database** создает базу данных Bacula, и, наконец, сценарий **make_mysql_tables** заполняет эту базу данных требуемыми таблицами. Аналогичные сценарии включены для баз данных PostgreSQL и SQLite. Сценарии предварительного создания баз данных можно найти в каталоге **src/cats** дистрибутива, содержащего исходный код программы Bacula.

Bacula сохраняет элемент таблицы для каждого файла, резервируемого из каждого клиента, поэтому ваш сервер базы данных должен иметь значительный объем памяти и дискового пространства. Таблицы базы данных для сети среднего размера могут легко вырасти до миллионов записей. Используя MySQL, вам следует сосредоточиться, по крайней мере, на ресурсах, определенных в файле **my-large.cnf**, включенном в дистрибутив. Если вы обнаружите, что ваша база данных каталога выросла так, что вот-вот может стать неуправляемой, вы всегда можете настроить второй экземпляр MySQL и использовать отдельные каталоги для различных групп клиентов.

## Конфигурирование демонов Bacula

После настройки базы данных, предназначенной для хранения каталога, вы должны сконфигурировать еще четыре компонента Bacula. По умолчанию все файлы конфигурации расположены в каталоге **/etc/bacula**. Bacula имеет отдельный файл конфигурации для каждого компонента. В табл. 10.3 перечислены имена файлов и компьютеры, на которых должен находиться каждый файл конфигурации.

**Таблица 10.3. Имена файлов конфигурации Bacula (в каталоге /etc/bacula)**

| Компонент | Файл | Компьютер |
|---|---|---|
| Демон управления | **bacula-dir.conf** | Сервер, который запускает демон управления |
| Демон хранения | **bacula-sd.conf** | Каждый сервер, который содержит демон хранения |
| Демон файлов | **bacula-fd.conf** | Каждый клиент, который будет создавать архивы |
| Консоль (пульт) управления | **bconsole.conf** | Каждый компьютер, используемый в качестве пульта управления |

Может показаться неразумной необходимость независимо конфигурировать каждый компонент программы Bacula при наличии единственного сервера, но такое модульное проектирование позволяет программе Bacula очень хорошо масштабироваться. Сервер ленточного накопителя работает на полную мощность? Добавьте второй сервер с собственным демоном хранения. Хотите сбрасывать резервные копии в сторонний компьютер? Инсталлируйте демон хранения на том сервере. Нужно заархивировать данные новых клиентов? Установите и сконфигурируйте демоны файлов этих клиентов. У вас появился новый администратор по вопросам резервирования? Инталлируйте консоль (пульт) управления на его (или ее) компьютере.

Файлы конфигурации — это текстовые файлы, предназначенные для чтения человеком. Образцы файлов конфигурации, включенные в дистрибутив Bacula, хорошо документированы и могут послужить прекрасной стартовой площадкой для типичной конфигурации.

Прежде чем приступить к подробному описанию процесса установки, необходимо дать определение ряду основных терминов, используемых в системе Bacula.

- "Задания" — это фундаментальные составляющие активности программы Bacula. Существует две их разновидности: архивирование и восстановление. Здание включает такие элементы: клиент, набор файлов, пул памяти и схему.
- "Пулы" — это группы физических носителей, на которых хранятся задания. Например, мы будем использовать два пула — для полных архивов и для инкрементных.
- "Наборы файлов" — это списки файловых систем и отдельных файлов. Наборы файлов могут быть явно включены в задания архивирования или восстановления либо исключены из них.
- "Сообщения" — это информация о состоянии демонов и заданий (фактически журнальные записи), которой обмениваются демоны. Они могут также передаваться по электронной почте или записываться в файлы журналов.

Мы не будем рассматривать все возможные параметры конфигурирования. Вместо этого в начале каждого раздела приведем общий обзор файла конфигурации, а затем подробнее остановимся на некоторых параметрах, которые, по нашему мнению, наиболее полезны или трудны для понимания.

## Разделы конфигурации

Файлы конфигурации Bacula состоят из разделов, обычно именуемых "ресурсами". Раздел каждого ресурса заключен в фигурные скобки. Некоторые ресурсы входят в несколько файлов конфигурации. Во всех файлах конфигурирования программы Bacula комментарии начинаются с символа **#**.

Все четыре файла конфигурации содержат ресурс **Director**.

```
# Пример файла конфигурации демона управления Bacula, /etc/bacula-dir.conf
Director {
   Name = bull-dir # a canonical name for our Bacula director
   DIRport = 9101
   Query File = "/etc/bacula/query.sql"
   Working Directory = "/var/Bacula/working"
   Pid Directory = "/var/run"
   Maximum Concurrent Jobs = 1
   Password = "zHpScUnHN9"
   Messages = Standard
}
```

Ресурс **Director** фактически служит истоком всех компонентов Bacula. Его параметры определяют имя и основы поведения демона управления. Они определяют коммуникационный порт, через который остальные демоны осуществляют обмен данными с демоном управления, каталог, в котором он хранит свои временные файлы, и количество заданий, которые демон управления может выполнять.

Пароли разбросаны по всем файлам конфигурации Bacula, и все они имеют разные цели. На рис. 10.2 показано, как должны соотноситься пароли на разных компьютерах в разных файлах конфигурации.

Несмотря на то что пароли записываются в файлах конфигурации открытым (незашифрованным) текстом, они никогда не передаются по сети в такой форме.

В данном примере и демон управления, и консоль расположены на одном компьютере. Тем не менее пароль должен присутствовать в обоих файлах конфигурации.

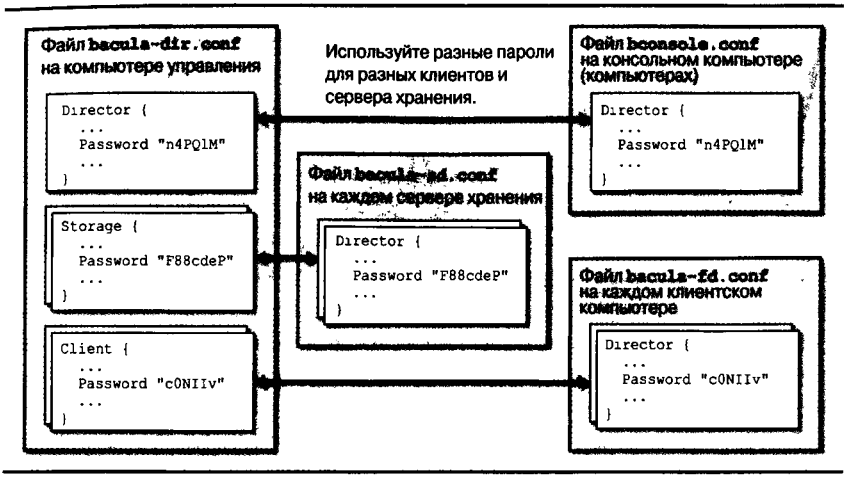



*Рис. 10.2. Пароли в файлах конфигурации Bacula*

Все демоны управления, хранения и файлов имеют ресурс **Messages**, который сообщает программе Bacula, как обрабатывать специальные типы сообщений, сгенерированные каждым Bacula-демоном. В типичной конфигурации демоны хранения и файлов отправляют свои сообщения демону управления.

```
Messages {
   Name = Standard
   director = bull-dir = all
}
```

В файле конфигурации демона управления ресурс **Messages** более сложный. Следующий код предписывает программе Bacula сохранять сообщения в журнале регистрации и отправлять их по электронной почте.

```
Messages {
  Name = Standard
  mailcommand = "/sbin/bsmtp -h localhost -f \"\(Bacula\)
  bacula@admin.com\" -s \"Bacula: %t %e of %c %l\" %r"
  operatorcommand = "/sbin/bsmtp -h localhost -f \"\(Bacula\)
  bacula@admin.com\" -s \"Bacula: Intervention needed for %j\" %r"
  mail = backups-admins@admin.com = all, !skipped
  operator = backups-tapeadmins@admin.com = mount
  console = all, !skipped, !saved
  append = "/var/log/bacula.log" = all, !skipped
}
```

Вы можете определить несколько ресурсов **Messages** для демона управления, а затем назначить их на специальные задания в ресурсах **Job**. Этот тип ресурса характеризуется перестраиваемой конфигурацией; полный список переменных и команд можно найти во встроенной оперативно доступной документации.

## Конфигурирование демона управления: файл bacula-dir.conf

Файл **bacula-dir.conf** — самый сложный файл конфигурации Bacula. Помимо описанных выше ресурсов **Director** и **Messages**, он требует минимум семь типов опре-

делений ресурсов: **Catalog**, **Storage**, **Pool**, **Schedule**, **Client**, **FileSet** и **Job**. Здесь мы проиллюстрируем определение каждого ресурса небольшим примером, но вы начинайте создание собственных файлов конфигурации с редактирования файлов-образцов, включенных в состав Bacula.

### Ресурсы каталогов

Ресурс **Catalog** указывает программе Bacula на базу данных конкретного каталога. Он включает имя каталога (чтобы вы могли определить несколько каталогов), имя базы данных и мандат базы данных.

```
Catalog {
  Name = MYSQL
  dbname = "bacula"; dbuser = "bacula"; dbpassword = "9p6NLm6CnQ"
}
```

### Ресурсы хранения

Ресурс **Storage** описывает, как общаться с конкретным демоном хранения, который в свою очередь несет ответственность за согласование с его локальными устройствами резервирования. Ресурсы хранения являются аппаратно-независимыми; демон хранения имеет собственный файл конфигурации, который описывает оборудование более детально.

```
Storage {
  Name = TL4000
  Address = bull
  SDPort = 9103
  Password = "jiKkrZuE00"
  Device = TL4000
  Autochanger = yes
  Maximum Concurrent Jobs = 2
  Media Type = LTO-3
}
```

### Ресурсы накопителей

Ресурс **Pool** группирует устройства резервирования (обычно это ленты) в наборы, которые используются специальными заданиями резервирования. Эта операция может быть полезной для отделения лент, которые служат для стороннего хранения архивов, от лент, которые используются для еженощных инкрементных архивов. Каждый накопитель назначается одному ресурсу **Pool**, поэтому нетрудно обеспечить автоматическую рециркуляцию лент разных наборов.

```
Pool {
      Name = Full_Pool
      Pool Type = Backup
      Recycle = yes
      AutoPrune = yes
      Storage = TL4000
      Volume Retention = 365 days
}
```

### Ресурсы схем резервирования

Ресурсы **Shedule** определяют временные таблицы (расписания) заданий архивирования. Обязательными являются только параметры, задающие имя, дату и время, но, как

видно из следующего примера, вы можете вставлять дополнительные значения. Эти значения переопределяют стандартные параметры, указанные в спецификации ресурса **Job**.

В данном случае полные архивирования выполняются в 20:10 по первым вторникам каждого месяца, а инкрементные архивирования (использующие другой ленточный накопитель) — еженедельно в 20:10 во все дни с четверга по понедельник.

```
Schedule {
     Name = "Nightly"
     Run = Level=Full Pool=FullPool 1st tue at 20:10
     Run = Level=Incremental Pool=IncrementalPool wed-mon at 20:10
}
```

### Ресурсы клиентов

Ресурсы **Client** определяют компьютеры, файлы которых должны быть архивированы. Каждый ресурс имеет уникальное имя, IP-адрес и пароль. Для каждого клиента требуется отдельный ресурс. Также необходимо указать каталог для сохранения метаданных резервирования. Параметры **File Retention** и **Job Retention** определяют длительность хранения в каталоге записей файлов и заданий для данного клиента. Если параметр **AutoPrune** установлен, дата истечения срока хранения удаляется из каталога. Эта опция влияет только на записи в каталоге, но не реальные файлы, хранящиеся на архивных лентах. Рециркуляция лент конфигурируется в ресурсе **Pool**.

```
Client {
  Name = harp
  Address = 192.168.1.28
  FDPort = 9102
  Catalog = MYSQL
  Password = "TbEpJqrcqy"
}
```

### Ресурсы файловых наборов

Ресурс **FilSet** определяет файлы и каталоги, которые должны быть либо включены в задания архивирования, либо исключены из них. Если у вас нет систем с идентичными схемами разбиения, вам, скорее всего, для каждого клиента необходимо использовать свой файловый набор. Ресурсы могут определять несколько параметров **Include** и **Exclude** и индивидуальные параметры **Options**, задаваемые в виде регулярных выражений. По умолчанию программа Bacula рекурсивно архивирует каталоги, но не включает в задание архивирования несколько логических разделов. Все логические разделы, которые нужно архивировать, должны быть указаны в отдельном параметре **File**.

В приведенном ниже примере используем такие дополнительные параметры, как **compression** (для сжатия данных перед их записью на ленту) и **signature** (для вычисления хеш-значения для каждого архивируемого файла). Эти параметры увеличивают загрузку процессора во время архивирования, но могут сэкономить емкость накопителей или при возникновении подозрения о нарушении безопасности позволяют идентифицировать файлы, которые были модифицированы.

```
FileSet {
    Name = "harp"
    Include {
       Options {
          signature=SHA1
          compression=GZIP
```

```
        }
        File = "/"
        File = "/boot"
        File = "/var"
    }
    Exclude = { /proc /tmp /.journal /.fsck }
}
```

### Ресурсы заданий

Ресурс **Job** определяет предельные характеристики конкретного задания архивирования посредством связывания воедино ресурсов **Client**, **FileSet**, **Storage**, **Pool** и **Schedule**. В общем случае для каждого клиента создается одно определение **Job**, хотя вы можете легко установить несколько заданий, если хотите архивировать различные файловые наборы (ресурсы **FileSets**) с различной частотой.

При желании для установки стандартных значений параметров для всех заданий архивирования можно предоставить ресурс **JobDefs**. Использование этого ресурса позволит упростить данные конфигурации, относящиеся к каждому заданию.

```
Job {
    Name = "Harp"
    JobDefs = DefaultJob
    Level = Full
    Write Bootstrap = "/bacula/bootstraps/harp.bsr"
    Client = harp
    FileSet = harp
    Pool = Full_Pool
    Incremental Backup Pool = Incremental_Pool
    Schedule = "Nightly"
    Prefer Mounted Volumes = no
    Max Run Time = 36000
}
```

Файл "самозагрузки" — это специальный текстовый файл (создаваемый программой Bacula), который содержит информацию о файлах, подлежащих восстановлению. Список файлов начальной загрузки содержит перечень файлов и томов, которые требуются для выполнения задания восстановления и чрезвычайно полезны для восстановления системы с нуля. Наличие этих файлов начальной загрузки не обязательно, но весьма желательно.

Файлы "самозагрузки" создаются во время операций восстановления или резервирования, если вы определили параметр **Write Bootstrap** для данного задания. Параметр **Write Bootstrap** указывает программе Bacula, куда записывать информацию о начальной загрузке, которая будет использована во время восстановления. Файлы "самозагрузки" перезаписываются в процессе полного архивирования и дописываются во время инкрементного архивирования.

## Конфигурирование демона хранения: файл bacula-sd.conf

Демоны хранения принимают данные от демонов управления файлами и передают их на реальный носитель хранения (и в обратном направлении при восстановлении). В файле **bacula-sd.conf** необходимо определить ресурсы **Storage**, **Device** и **Autochanger**, не говоря уже о ресурсах **Messages** и **Director**. Ниже приведен полный пример файла конфигурации.

### Ресурс Director

Ресурс **Director** управляет тем, какие управляющие устройства разрешены для контакта с демоном хранения. Пароль в ресурсе **Storage** файла **bacula-dir.conf** должен соответствовать паролю в ресурсе **Director** файла **bacula-sd.conf**.

```
Director {
   Name = bull-dir
   Password = "jiKkrZuE00"
}
```

### Ресурсы Storage

Ресурс **Storage** сравнительно прост. Он определяет ряд основных рабочих параметров, таких как сетевой порт демона и рабочий каталог.

```
Storage {
   Name = bull-sd
   SDPort = 9103
   WorkingDirectory = "/var/bacula/working"
   Pid Directory = "/var/run"
   Maximum Concurrent Jobs = 2
}
```

### Ресурсы Device

Каждое устройство архивирования получает собственное определение ресурса **Device**. Этот ресурс характеризует физическое устройство архивирования, будь то магнитная лента, оптический накопитель или оперативная память. В состав характеристик входит тип устройства, производительность и данные о механизме автоматической смены носителя для устройств, которые управляются таким механизмом смены или роботом.

В приведенном ниже примере определяется накопитель LTO-3 (Linear Tape-Open) с автоматической сменой лент. Обратите внимание на то, что файл устройства **/dev/nst0** соответствует устройству без перемотки, которое требуется практически во всех ситуациях. Параметр **Name** определяет символическое имя, которое используется для связывания накопителя с соответствующим ресурсом **Autochanger**. Параметр **Name** также позволяет запомнить, какую единицу оборудования описывает данный ресурс.

Параметр **AlwaysOpen** указывает о необходимости сохранения устройства в открытом состоянии, если только администратор специально не потребует его размонтирования. Этот параметр экономит время и уберегает ленту от дополнительных нагрузок, поскольку исключает выполнение команд перемотки и позиционирования между заданиями.

```
Device {
   Name = TL4000-Drive0
   Media Type = LTO-3
   Archive Device = /dev/nst0
   AutomaticMount = yes
   AlwaysOpen = yes
   RemovableMedia = yes
   RandomAccess = no
   Autochanger = yes
}
```

### Ресурсы Autochanger

Необязательное определение ресурса **Autochanger** требуется только в случае, если вы — счастливый обладатель механизма смены носителя. Он связывает устройства хранения данных с механизмом автоматической смены носителя и определяет команду, которая обеспечивает смену лент.

```
Autochanger {
   Name = TL4000
   Device = TL4000-Drive0,TL4000-Drive1
   Changer Command = "/etc/bacula/mtx-changer %c %o %S %a %d"
   Changer Device = /dev/changer
}
```

## Конфигурирование консоли: файл bconsole.conf

Программа консоли служит для обмена данными с демоном управления при планировании заданий, проверки их состояния или восстановления данных. Файл **bconsole.conf** указывает консоли, как следует осуществлять обмен данными с демоном управления Bacula. Параметры в этом файле должны соответствовать параметрам, определенным в ресурсе **Director** файла конфигурации демона управления (**bacula-dir.conf**), за исключением параметра **Address**.

```
# Файл конфигурации консоли, bconsole.conf
Director {
   Name = bull-dir
   DIRport = 9101
   Address = bull
   Password = "zHpScUnHN9"
}
```

## Установка и конфигурирование демона управления файлами клиента

Демон управления файлами клиентов архивирования обменивается данными с демоном хранения во время выполнения заданий архивирования и восстановления. Он должен быть установлен и сконфигурирован на каждом компьютере, файлы которого нужно архивировать с помощью программы Bacula.

Программа Bacula распространяется в двоичной форме и подходит для многих платформ, включая Windows и различные версии Linux. В системах UNIX установку демона можно выполнить, копируя исходное дерево Bacula на каждый компьютер клиента и выполняя команду **./configure --enable-client-only**. Затем необходимо выполнить команды **make** и **make install**. Вы можете жестко закодировать стандартные значения для многих параметров демона файлов во время конфигурации, но поддержка будет легче, если вы просто перечислите их в своем файле **bacula-fd.conf**. По умолчанию двоичные файлы инсталлируются в каталог **/sbin**, а файлы конфигурации — в каталог **/etc/bacula**.

После установки демона файлов сконфигурируйте его, отредактировав файл **bacula-fd.conf**, который разделен на три части. Первая часть включает ресурс **Director**, который сообщает демону файлов, какое устройство управления разрешается использовать для планирования операций архивирования для данного клиента. Ресурс **Director** также включает параметр **Password**, который должен совпадать с паролем, указанным

в ресурсе **Client** собственного файла конфигурации демона управления. Вторая часть файла **bacula-fd.conf** представляет собой ресурс **FileDaemon**, который определяет имена клиентов и задает порт, прослушиваемый демоном управления файлами на предмет команд от демона управления.

Последний компонент программы — ресурс **Messages** (см. раздел "Разделы конфигурации" выше в этой главе), который определяет, как должны быть обработаны локальные сообщения.

## Запуск демонов Bacula

После того как демоны сервера установлены, а клиент сконфигурирован, необходимо запустить демоны с помощью сценария запуска, размещенного в каталоге установки сервера (**./bacula start**). Эта же команда используется на каждом клиентском компьютере для запуска демона управления файлами.

Сценарий запуска программы **bacula** должен быть сконфигурирован для выполнения во время начальной загрузки. Может ли это быть сделано за вас или нет — зависит от вашей системы и метода инсталляции. О том, как запускать службы во время загрузки системы, показано на примерах систем в главе 3.

После того как демоны Bacula запущены, программу консоли (**bconsole** в каталоге установки) можно использовать для проверки их состояния, добавления носителей в пулы и для выполнения заданий архивирования и восстановления. Программу **bconsole** можно запускать с любого компьютера, если она корректно установлена и сконфигурирована.

```
$ sudo ./bconsole
Connecting to Director bull:9101
1000 OK: bull-dir Version: 2.4.4 (23 December 2009)
Enter a period to cancel a command.
*
```

Для ознакомления с полным списком поддерживаемых ею команд используйте команду **help** консоли.

## Добавление носителей в пулы

Прежде чем выполнять задания архивирования, необходимо пометить ленту и включить ее в пулы носителей, определенные в файле конфигурации демона управления. Чтобы записать программную "метку" на пустую ленту и назначить ее пулу Bacula, необходимо использовать команду **label** консоли.

Всегда сопоставляйте свою метку Bacula с физической меткой на вашей ленте. Если у вас есть механизм автоматической смены носителя, который поддерживает штрихкоды, вы можете использовать команду **label barcode**, чтобы автоматически помечать любые пустые ленты читабельными значениями с их этикеток со штриховым кодом.

Для проверки того, что лента была добавлена в нужный пул и помечена как допускающая добавление данных, следует использовать команду **list media**.

## Выполнение архивирования вручную

Чтобы выполнить архивирование вручную, необходимо использовать команду **run** консоли. Эта команда не требует никаких аргументов. При ее выполнении консоль отображает все задания, определенные в файле конфигурации демона управления. Любые параметры команды **run** можно изменять, следуя подсказкам, подобным меню, на консоли.

Приведенный ниже пример демонстрирует выполнение вручную полной архивации файлов сервера **harp** с использованием стандартных параметров, указанных в файлах конфигурации.

```
* run harp
Run Backup job
JobName: harp
Level: Full
Client: harp
FileSet: harp
Pool: FullPool
Storage: SureStore
When: 2009-10-08 10:56:41
Priority: 10
OK to run? (yes/mod/no): yes
Run command submitted.
```

После успешной передачи задания архивирования демону управления его состояние можно отслеживать с помощью команды **status** консоли. Для систематичного получения информации обо всех обновлениях при ее поступлении можно использовать также команду **messages** консоли. В зависимости от настройки ресурсов **Messages** системы, подробный итоговый отчет может отправляться также администратору системы Bacula.

## Выполнение задания восстановления

Для того чтобы восстановить файлы, необходимо запустить консоль и выполнить команду **restore**. Как и команда **run**, команда **restore** выбирается из меню. Ее выполнение начинается с вывода информации о том, какие задания должны быть прочитаны для восстановления целевых файлов. Команда **restore** предоставляет несколько методов указания необходимых идентификаторов заданий. После того как набор заданий выбран, из них можно выбрать файлы, которые нужно восстановить.

```
* restore
To select the JobIds, you have the following choices:
1: List last 20 Jobs run
2: List Jobs where a given File is saved
3: Enter list of comma separated JobIds to select
4: Enter SQL list command
5: Select the most recent backup for a client
6: Select backup for a client before a specified time
7: Enter a list of files to restore
8: Enter a list of files to  restore before a specified time
9: Find the JobIds of the most recent backup for a client
10: Find the JobIds for a backup  for a client before a specified time
11: Enter a list of directories to restore for found JobIds
12: Cancel
Select item:  (1-12):
```

Вероятно, три наиболее полезных запроса — "Select the most recent backup for a client" (Выберите самую последнюю резервную копию клиента) (№5), "Select backup for a client before a specified time" (Выберите резервную копию клиента до заданного времени) (#6) и "List Jobs where a given File is saved" (Перечислите задания, содержащие данный файл) (№2). Первые две опции предоставляют оболочку для выбранных файлов, по аналогии с командой **restore -i**. Последняя опция пригодится тем рассеянным пользователям, которые не в состоянии запомнить точное местонахождение случайно удаленного фай-

ла. Еще одна полезная команда — опция №4, "Enter SQL list command" (Введите SQL-команду), которая позволяет вводить любой надлежащим образом сформатированный SQL-запрос. С помощью этой опции можно находить идентификаторы заданий. (Безусловно, вам должна быть известна логическая структура используемой базы данных.)

Предположим, что пользователю необходимо скопировать свой восстановленный сценарий **pw_expire.pl**. Однако он не помнит, в каком именно каталоге хранился этот файл. Кроме того, пользователю желательно восстановить файлы в каталог **/tmp** на исходном компьютере. Многих системных администраторов подобный запрос может повергнуть в шоковое состояние, но для администратора Bacula выполнение этой задачи не представляет никакой сложности. (К сожалению, формат вывода результатов поиска, используемый программой Bacula, столь подробен, что нам пришлось максимально уменьшить масштаб выводимой информации.)

```
* restore
…
To select the JobIds, you have the following choices:
1: List last 20 Jobs run
2: List Jobs where a  given File is saved
…
Select item:  (1-12): 2
Defined Clients:
1: bull
2: harp
…
Select the Client (1-12): 2
Enter Filename (no path): pw_expire.pl
+-------+-----------------------------------+---------------------+--------
| JobId | Name                              | StartTime           | JobType…
+-------+-----------------------------------+---------------------+--------
| 4484  | /home/jim/development/pw_expire.pl | 2009-11-03 20:11:35 | B…
| 4251  | /home/jim/development/pw_expire.pl | 2009-10-21 18:03:01 | B…
| 4006  | /home/jim/development/pw_expire.pl | 2009-10-06 20:10:02 | B…
+-------+-----------------------------------+---------------------+-----------
```

Из списка экземпляров файла **pw_expire.pl**, созданного программой Bacula, мы видим несколько недавних резервных копий и соответствующие им идентификаторы заданий. Затем программа Bacula возвращает нас к меню восстановления, где можно использовать опцию №3 ("Enter job IDs" (Введите идентификаторы заданий)), чтобы выбрать нужное задание.

```
Select item: (1-12): 3
Enter JobId(s), comma separated, to restore: 4484
You have selected the following JobId: 4484
Building directory tree for  JobId 4484 ... 1 Job, 779,470 files
You are now entering file selection mode where you add (mark) and
remove (unmark) files to be restored. No files are initially added, unless
you used the "all" keyword on the command line.
Enter "done" to leave this mode.
cwd is: /
$ cd /home/jim/development
cwd is: /home/jim/development
$ dir
…
-rwxr-xr-x 1 jim atrust 923 2009-10-25 12:05:43 /home/jim/development/pw_exp…
```

```
$ mark pw_expire.pl
1 files marked.
$ done
```

Хотя в этом примере и не показано, но иногда программа Bacula отображает идентификационные номера заданий с запятой (4,484). Вы должны опускать запятую при повторном вводе. В противном случае программа интерпретировала бы это значение как список идентификаторов заданий, разделенных запятыми.

```
Bootstrap records written to /var/bacula/working/restore.bsr
The restore job will require the following Volumes:
000879L3
1 file selected to be restored.
Defined Clients:
1: bull
2: harp
...
Select the Client (1-2): 2
```

Затем программа записывает файл начальной загрузки, который она будет использовать для выполнения восстановления, отображает имена необходимых ей ленточных томов и предлагает выбрать клиентский компьютер, на котором нужно восстановить файлы. В данном примере восстановление было выполнено на исходном компьютере **harp**.

```
Run Restore Job
JobName: RestoreFiles
Bootstrap: /bacula/bacula/working/jf-dir.restore.4.bsr
Where: /var/restore
Replace: always
FileSet: Full Set
Backup Client: harp
Restore Client: harp
Storage: LTO3-TL4000
When: 2009-11-23 15:13:05
Catalog: MYSQL
Priority: 10
OK to run? (yes/mod/no):
```

Перед выполнением данного конкретного задания мы хотим изменить параметры, заданные по умолчанию. В частности, в соответствии с пожеланием пользователя нам нужно изменить место назначения этого восстановления на **/tmp**.

```
OK to run? (yes/mod/no): mod
Parameters to modify:
1: Level
2: Storage
3: Job
4: FileSet
5: Restore Client
6: When
7: Priority
8: Bootstrap
9: Where
10: File Relocation
11: Replace
12: JobId
Select parameter to modify (1-12): 9
```

```
Please enter path prefix for restore (/ for none): /tmp
Run Restore job
JobName: RestoreFiles
Bootstrap:  /var/Bacula/working/restore.bsr
Where:      /tmp
...
OK to run? (yes/mod/no): yes
Run command submitted.
Restore command done.
```

После внесения изменений мы передаем задание демону управления, который его выполняет. Затем можно воспользоваться командой **messages** для просмотра журнала выполнения задания.

## Архивирование клиентов Windows

Вы можете загрузить предварительно созданные двоичные файлы для клиентов Windows из файла **bacula.org**. Программа Bacula прекрасно справляется с резервированием Windows-файлов данных, но ей приходится выполнить дополнительное действие для создания надежной копии системы Windows уровня 0. К сожалению, программа Bacula не имеет никакого понятия о системном реестре Windows или состоянии системы; она также не понимает блокирования открытых файлов в Windows. Как минимум, вам следует сконфигурировать сценарий предварительного выполнения, который бы запускал встроенную в Windows функцию восстановления системы (System Restore). Эта функция создает локальную копию состояния системы, а программа Bacula затем получит текущее состояние системы в заархивированной форме. Чтобы корректно собрать заблокированные файлы, вам, возможно, стоит использовать Windows-службу фонового копирования тома (Volume Shadow Copy Service — VSS).

Просмотрите каталог **examples** в исходном дереве программы Bacula на предмет других Windows-ориентированных предложений. Например, вы найдете интеллектуальный сценарий, который может "осчастливить" клиента обновлениями Windows-демона управления файлами посредством встроенной Bacula-системы восстановления. Windows-клиент в последней версии программы Bacula поддерживает не только клиенты Windows 7 и 64-битовые клиенты, но и (пока в качестве эксперимента) архивы Microsoft Exchange.

## Мониторинг конфигураций программы Bacula

Системное администрирование требует бдительности, и создание резервных копий — не исключение. Если в процессе создания резервных копий возникнет сбой, который не будет быстро устранен, важные данные будут стопроцентно утеряны.

Задания Bacula создают отчет, который направляется, согласно ресурсу **Message**, в файл конфигурации демона управления. Отчет содержит основную информацию об использованных томах, размере и количестве заархивированных файлов и о любых возникших ошибках. Как правило, он предоставляет достаточный объем информации для решения любых не очень серьезных проблем. Рекомендуем задать такую конфигурацию, чтобы важные сообщения приходили на ваш ящик входящих сообщений электронной почты или на ваш пейджер.

Команду **status** консоли можно использовать для запроса различной информации о демонах Bacula. Следующий вывод содержит информацию о запланированных, выполняющихся и прерванных заданиях. Команда **messages** — простой способ обзора недавних записей журнала.

Программа Bacula включает плагин Nagios, который упрощает интеграцию архивов в существующую инфраструктуру мониторинга. Общую информацию о плагине Nagios можно получить из раздела 21.12.

## Советы по использованию программы Bacula

Две наиболее часто встречающиеся проблемы — невозможность запуска демонов управления файлами клиентов и неспособность демонов хранения обнаружить какие-либо ленточные тома, доступные для добавления данных. В приведенном ниже примере демон управления сообщает, что задание архивирования было прервано с фатальной ошибкой вследствие невозможности осуществления обмена данными с демоном управления файлами компьютера **harp**. Сообщение об этой ошибке может многократно повторяться в конце итогового отчета.

```
...
SD termination status:     Waiting on FD
Termination:               *** Backup Error ***

11-Nov 21:06 bull-dir JobId 259: Warning: bsock.c:123 Could not connect to
    Client: harp on 192.168.1.3:9102. ERR=Connection refused Retrying ...
11-Nov 21:31 bull-dir JobId 259: Warning: bsock.c:123 Could not connect to
    Client: harp on 192.168.1.3:9102. ERR=Connection refused
```

В следующем примере демон хранения сообщает, что для запрошенного задания архивирования отсутствуют доступные ленточные тома соответствующего пула. Проблему можно устранить, либо добавив в пул новый том, либо очистив и повторно используя существующий том. Задание не нужно запускать снова — программа Bacula должна продолжить его выполнение, если только оно не будет прервано явно.

```
06-May 23:01 bull-sd JobId 1545: Job  Southernfur.2009-05-06_19.03.00.07
    waiting. Cannot find any appendable volumes.
Please use the "label"  command to create a new Volume for:
    Storage:     "TL4000-Drive0" (/dev/nst0)
    Pool:        Full_Pool
    Media type: LTO-3
```

Для того чтобы ознакомиться с более подробной информацией о действиях, выполняемых демонами, их можно вынудить отсылать отладочную информацию на консоль, добавив опцию **-d***nnn* к команде запуска. Например, можно использовать такую команду.

```
$ sudo ./bacula start -d100
```

Значение *nnn* представляет уровень отладки. Типичные значения лежат в диапазоне от 50 до 200. Чем выше это значение, тем больший объем информации отображает команда. Отладку можно также активизировать из консоли с помощью команды **setdebug**.

## Альтернатива программе Bacula

В Интернете доступно несколько других бесплатных или условно бесплатных пакетов резервного копирования. Следующие пакеты заслуживают особого внимания, хотя они все еще находятся в состоянии активной разработки.

- Amanda: очень популярная и проверенная система, которая выполняет архивирование файлов систем UNIX и Linux на одном ленточном устройстве. См. **amanda.org**.

- **rsync**: бесплатная утилита, входящая в состав многих пакетов Linux (работает на всех наших примерах систем). Она может выполнять синхронизацию файлов с одного компьютера на другой и ее можно использовать совместно с SSH для безопасной передачи данных по Интернету. Утилита **rsync** настолько интеллектуальна, что передает по сети только различия в файлах, что говорит о ее высокой эффективности. Дополнительная информация о программе **rsync** приведена в разделе 19.2. Обратите внимание на то, что одной утилиты **rsync** обычно не достаточно для архивов, поскольку она не сохраняет несколько копий. И не создает автономных архивов.
- **star**: более производительная реализация программы **tar**. Эта программа входит в состав Linux и всех типов систем UNIX.
- Mondo Rescue: утилита, которая выполняет архивирование файлов систем Linux на CD-R, DVD-R, ленту или жесткий диск. Этот пакет особенно удобен, если необходимо выполнить восстановление в системе, в которой нет какого-либо программного обеспечения. Подробнее о нем можно узнать по адресу: **mondorescue.org**.

## 10.9. Коммерческие системы резервного копирования

Нам бы хотелось, чтобы UNIX была единственной операционной системой в мире, но, к сожалению, это не так. Анализируя различные коммерческие системы резервного копирования, необходимо учитывать их способность взаимодействовать с другими операционными системами, которые используются в организации. Большинство современных продуктов позволяет включать рабочие станции UNIX, Windows и Mac OS в схему резервного копирования. Следует также учитывать наличие массивов хранилищ и файловых серверов, работающих не под управлением UNIX.

Защищать от сбоев нужно также портативные компьютеры пользователей и другие компьютеры, которые не имеют постоянного подключения к сети. Коммерческая программа должна правильно обрабатывать ситуации, когда архивируются идентичные файлы с каждого портативного компьютера. В конце концов, сколько резервных копий файла **command.com** вам нужно?

Поскольку система Bacula идеально подходит для наших потребностей, мы не приобрели богатого опыта работы с другими коммерческими пакетами. Мы опросили нескольких знакомых администраторов из коммерческих организаций на предмет их впечатлений от систем, которыми они пользуются. Ниже приведены их комментарии.

### ADSM/TSM

Пакет ADSM, разработанный компанией IBM, впоследствии был приобретен компанией Tivoli. Сегодня он носит название TSM (Tivoli Storage Manager — диспетчер систем хранения компании Tivoli), хотя права собственности снова отошли к IBM, и представляет собой систему управления данными, поддерживающую также резервное копирование. Дополнительную информацию можно найти на веб-сайте **ibm.com/tivoli**.

Преимущества

- принадлежность компании IBM;
- удачная ценовая политика;
- низкий процент сбоев;

- использование дисковых кеш-буферов, что удобно при работе с низкоскоростными клиентами;
- поддержка клиентов Windows;
- великолепная документация (за отдельную плату).

Недостатки

- плохой графический интерфейс;
- каждые два файла занимают 1 Кбайт в базе данных;
- сложность системы постоянно возрастает.

## Veritas NetBackup

В 2005 году произошло слияние компаний Veritas и Symantec. Объединенная компания продает системы резервного копирования для различных платформ. Информацию о них можно получить на веб-сайте `symantec.com`. NetBackup — продукт, предназначенный для обеспечения резервного копирования и восстановления данных в средних и крупных гетерогенных сетях, но небольшие фирмы могут обойтись системой BackupExec.

Преимущества

- неплохой графический интерфейс;
- взаимодействие с высокоскоростными выделенными каналами связи с системой хранения данных (архитектура "сервер-хранилище данных") и файловыми серверами NetApp;
- инсталляция по методу принудительной рассылки в UNIX и Windows;
- возможность записи лент в формате GNU-версии команды `tar`;
- централизованная база данных, но поддерживается и распределенное резервное копирование.

Недостатки

- ряд программных ошибок;
- неудачная ценовая политика;
- пресловутые слабые места системы безопасности.

## EMC NetWorker

Американская компания-монстр EMC — мировой лидер на рынке продуктов, услуг и решений для хранения информации и управления ею — приобрела Legato NetWorker в 2003 году. В то время Veritas и Legato были ведущими компаниями в области резервного копирования в вычислительных сетях масштаба предприятия.

Преимущества

- удачная ценовая политика;
- сервисное программное обеспечение может работать на всех наших примерах платформ;
- поддержка различных клиент-платформ;
- комплексное восстановление системы с нуля.

Недостаток

- существенное перекрытие по продуктам EMC.

## Прочие программы

Уже упоминавшийся Кертис Престон, автор книги по резервному копированию, вышедшей в издательстве O'Reilly, ведет очень полезную веб-страницу **backupcentral.com**, посвященную данной тематике (программы зеркального дублирования дисков, работы с файловыми системами, удаленного резервного копирования, внешнего хранения данных и др.).

# 10.10. Рекомендуемая литература

Preston W. Curtis. *Backup & Recovery: Inexpensive Backup Solutions for Open Systems.* Sebastopol, CA: O'Reilly Media, 2007.

# 10.11. Упражнения

10.1. Проанализируйте процедуру резервного копирования, применяемую в вашей компании или организации. На каком компьютере (или компьютерах, если их несколько) выполняется резервное копирование? Какие типы накопителей используются? Где хранятся ленты? Можно ли улучшить существующую схему?

10.2. Какие действия необходимо предпринять для восстановления файлов в системе Bacula? Как найти нужную ленту?

10.3. ★ Укажите действия, которые необходимо осуществить для выполнения трех запрошенных операций восстановления при наличии следующего вывода команд **df** и **/etc/dumpdates**. Аргументируйте свое решение. Примите, что запрос на восстановления датируется 18 января.

10.4. Вот результаты выполнения команды **df** на сайте `khaya.cs.colorado.edu`.

```
/dev/hda8      256194      81103     161863     33%    /
/dev/hda1       21929       4918      15879     24%    /boot
/dev/hda6     3571696      24336    3365924      1%    /local
/dev/hda10     131734       5797     119135      5%    /tmp
/dev/hda5     1815580    1113348     610004     65%    /usr
/dev/hda7      256194      17013     225953      7%    /var
```

10.5. А вот содержимое файла **/etc/dumpdates** на **khaya.cs.colorado.edu**.

```
/dev/hda8          2 Sun Jan  17  22:59:23 2010
/dev/hda6          3 Sun Jan  17  22:51:51 2010
/dev/hda7          3 Sun Jan  17  22:50:24 2010
/dev/hda5          9 Sun Jan  17  22:46:25 2010
/dev/hda5          1 Tue Jan  12  22:45:42 2010
/dev/hda7          0 Tue Jan  12  23:14:47 2010
/dev/hda6          1 Tue Jan  12  23:14:32 2010
/dev/hda8          1 Tue Jan  12  23:14:17 2010
/dev/hda6          0 Sun Jan  10  22:47:31 2010
/dev/hda1          1 Fri Jan   8  22:16:05 2010
/dev/hda7          1 Thu Jan   7  22:08:09 2010
/dev/hda1          4 Sun Jan   3  22:51:53 2010
/dev/hda7          2 Thu Dec  24  22:53:52 2009
/dev/hda5          0 Tue Nov   3  22:46:21 2009
/dev/hda1          0 Mon Sep  21  22:46:29 2009
```

```
/dev/hda8          0 Mon Aug   24    23:01:24 2009
/dev/hda1          3 Wed Jul   29    22:52:20 2009
/dev/hda6          2 Wed Jul   29    23:01:32 2009
```

а) "Пожалуйста, восстановите весь мой домашний каталог (**/usr/home/clements**) в том состоянии, в котором он был несколько дней назад. Похоже, что я потерял всю кодовую базу своего проекта."

б) "Я случайно выполнил команду **sudo rm -rf /*** на своем компьютере **khaya**. Не могли бы Вы восстановить все файловые системы из последних резервных копий?"

в) "Все мои MP3-файлы, которые я скачивал из BitTorrent в течение более месяца, пропали. Они хранились в каталоге **/tmp/mp3/**. Не могли бы Вы восстановить их?"

10.6. ★ Разработайте схему резервного копирования для описанных ниже сценариев. Предполагается, что каждый компьютер оснащен жестким диском емкостью 400 Гбайт и домашние каталоги пользователей хранятся локально. Выберите устройство резервного копирования, оптимальное по соотношению стоимости и трудоемкости эксплуатации. Аргументируйте свой выбор.

а) научно-исследовательский центр с 50 компьютерами. На каждом компьютере хранится множество важных и часто изменяемых данных

б) небольшая компания, занимающаяся выпуском программного обеспечения и располагающая десятью компьютерами. Исходные коды хранятся на сервере, емкость жесткого диска которого составляет 4 Тбайт. Разрабатываемые исходные коды постоянно меняются в течение дня, а домашние каталоги пользователей меняются редко. Стоимость системы резервного копирования не играет особой роли, зато крайне важна безопасность данных

в) домашняя сеть с двумя компьютерами. Здесь основной фактор — это цена, а пользователи — не системные администраторы

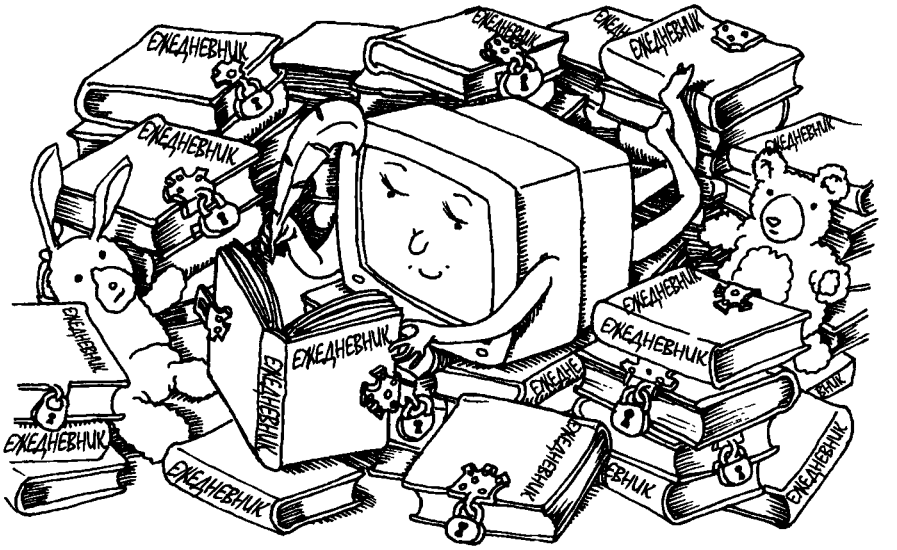
10.7. ★ Разработайте стратегию восстановления архивов для каждого из трех сценариев, перечисленных в упражнении 10.4.

10.8. ★ Напишите операторы конфигурирования Bacula, которые реализуют план архивирования, разработанный для упражнения 10.4.

10.9. ★ Опишите действия, которые нужно было бы предпринять для выполнения команды **dump** для удаленного накопителя на магнитной ленте по безопасному тунелю SSH.

# Глава 11

# Система Syslog и журнальные файлы




Системные демоны, ядро, утилиты и службы — все они генерируют журнальные данные, которые регистрируются и попадают на далеко не безразмерные диски. "Срок полезной службы" большинства данных ограничен, поэтому их приходится группировать, сжимать, архивировать и, наконец, удалять. Доступом и данными аудита необходимо управлять точно в соответствии с регулятивными правилами хранения информации или политикой безопасности, принятой для вашего узла.

Опытные администраторы стараются без промедления просматривать журналы регистрации. Системные журналы зачастую содержат важную информацию, которая может помочь в решении досадных проблем с конфигурацией. Если отказывается запускаться какой-нибудь демон или застарелая ошибка продолжает "ставить палки в колеса" загружающейся системе, первым делом проверьте журналы регистрации.

Ранее в UNIX были предприняты попытки использовать интегрированную систему регистрации событий syslog, но можно сказать, что результаты в лучшем случае имели переменчивый успех. Несмотря на то что демон **syslogd** все еще считается "главным героем" в области регистрации, множество приложений, сетевых демонов, сценариев запуска и другие члены "комитета бдительности" используют собственные журналы регистрации "специального вида". Следствием такого беззакония стало появление журналов, которые существенно отличались от "подобных себе" среди разновидностей UNIX и даже дистрибутивов Linux.

В большинстве случаев регистрируемое событие перехватывается в виде одной строки текста, которая включает время и дату, тип и степень серьезности события, а также любые другие релевантные детали. Различные компоненты сообщения могут разделяться

пробелами, символами табуляции или знаками пунктуации (в зависимости от специфики файла).

Поскольку большинство журналов регистрации представляет собой текстовые файлы, их можно просматривать или анализировать такими стандартными инструментами, как **cat**, **grep**, **tail** и Perl. Многие современные системы также включают средства управления журналами регистрации, которые выполняют ротацию, сжатие и управление журналами регистрации (ежедневно или еженедельно).

Системные журналы, управляемые системой syslog, обычно содержат события от нескольких источников. Например, "жалобы" от ядра и сетевого демона могут соседствовать друг с другом. На узлах, которые содержат централизованный сервер регистрации, события от нескольких узлов можно группировать и обрабатывать вместе.

Следующий фрагмент отображает типичные события от централизованного сервера регистрации.

```
Dec 18 15:12:42 av18.cs.colorado.edu sbatchd[495]: sbatchd/main: ls_info()
   failed: LIM is down;  try later; trying …
Dec 18 15:14:28 proxy-1.cs.colorado.edu pop-proxy[27283]: Connection from
   128.138.198.84
Dec 18 15:14:30 mroe.cs.colorado.edu pingem[271]: maltese-
   office.cs.colorado.edu has not answered 42 times
Dec 18 15:15:05 schwarz.cs.colorado.edu vmunix: Multiple softerrors: Seen 100
   Corrected Softerrors from SIMM J0201
Dec 18 15:15:16 coyote.cs.colorado.edu PAM_unix[17405]: (sshd) session closed
   for user trent
Dec 18 15:15:48 proxy-1.cs.colorado.edu pop-proxy[27285]: Connection from
   12.2.209.183
Dec 18 15:15:50 av18.cs.colorado.edu  last message repeated 100 times
```

🕮 Дополнительная информация о PAM приведена в разделе 22.5.

Этот пример содержит записи от нескольких различных узлов (av18, proxy-1, mroe, schwarz и coyote), а также от таких программ, как **sbatchd**, **pop-proxy**, **pingem**, и библиотеки подключаемых модулей аутентификации (Pluggable Authentication Modules — PAM).

Важность использования хорошо определенной межузловой стратегии регистрации событий значительно возросла наряду с принятием таких формальных ИТ-стандартов, как COBIT (свод правил по управлению и контролю ИТ) и ISO 27002, а также с выработкой отраслевых норм. В наши дни для действий, связанных с регистрацией событий, эти внешние стандарты могут потребовать от вас поддержки централизованного корпоративного хранилища с использованием отметок времени, предоставляемых посредством протокола NTP, и строгого соблюдения расписания. Некоторые стратегии мы обсудим ниже в этой главе.

# 11.1. Обнаружение файлов регистрации

UNIX-системы часто критикуют за несогласованный подход к управлению журнальной регистрацией, и эта критика вполне справедлива. Посмотрите на содержимое каталога файлов регистрации — вы обязательно найдете файлы с такими именами, как **maillog**, **ftp.log** и, возможно, **lpNet**, **lpd-errs** или **console_log**. Помимо произвольных имен, файлы регистрации часто разбросаны по разным каталогам и файловым системам. По умолчанию большинство этих файлов хранится в каталоге **/var/adm** или **/var/log**.

Системы Linux в этом плане — гораздо более интеллектуальные, хотя в каждом дистрибутиве журнальные файлы и названы, и распределяются по каталогам по-разному. В основном, Linux-приложения записывают журнальную информацию в файлы, находящиеся в каталоге **/var/log**.

В табл. 11.1 представлена информация о наиболее часто используемых журнальных файлах демонстрационных дистрибутивов. Указаны, в частности, следующие сведения:

- журнальные файлы, подлежащие архивированию или другой обработке;
- программа, создающая каждый из этих файлов;
- информация о том, где задается имя файла;
- периодичность удаления устаревшей информации, которая считается приемлемой;
- системы (из числа демонстрационных), в которых используется каждый из этих файлов;
- описание содержимого файлов.

Имена файлов в табл. 11.1 даны относительно каталогов **/var/adm**, **/var/log** или **/var/log/syslog**, если не указано иное.

Журнальные файлы обычно принадлежат пользователю **root**, хотя соглашения о владельцах и режимах доступа к этим файлам неодинаковы в разных системах. В некоторых случаях менее привилегированный демон (такой, как **httpd** или **mysqld**) может потребовать доступ для записи, а затем задать ее владельца и режим работы соответственно. Вам, возможно, для просмотра важных журнальных файлов, которые имеют строгие разрешения доступа, придется использовать команду **sudo**. В качестве альтернативы для журнальных файлов, которые не содержат важные системные данные, можно изменить разрешения доступа и позволить "всеобщее" чтение. Мы обычно рекомендуем последний вариант применять для файлов регистрации, которые вам нужно просматривать регулярно (например, журналы регистрации Apache в каталоге **/var/log/httpd**).

Следует отметить, что многие журнальные файлы из числа перечисленных в табл. 11.1 контролируются системой Syslog, но стандартная ее конфигурация сильно зависит от системы. Если бы файл **/etc/syslog.conf** был более согласованным, управление журнальными файлами велось бы примерно одинаково во всех операционных системах.

Журнальные файлы могут очень быстро увеличиваться в размере, особенно это касается файлов для службы электронной почты, веб- и DNS-серверов. Неконтролируемый файл регистрации может заполнить весь диск и тем самым "положить систему". Поэтому мы предпочитаем использовать отдельный раздел для самых "зашумленных" и занятых журнальных файлов. В системах Linux отдают предпочтение каталогам **/var** или **/var/log**. Соглашения в других системах отличаются друг от друга.

## Специальные журнальные файлы

Большинство журнальных файлов — это обычные текстовые файлы, в которые записываются сообщения о "важных" событиях. Но некоторые файлы из числа перечисленных в табл. 11.1 имеют совершенно другое назначение.

Файл **wtmp** (иногда **wtmpx**) содержит записи о том, когда пользователи входили в систему и выходили из нее, а также когда система выключалась или перезагружалась. Это довольно простой файл (новые записи просто добавляются в конец), тем не менее, он хранится в бинарном виде. Расшифровать содержимое файла можно с помощью команды **last**.

🕮 Подробнее о разреженных файлах см. сноску в разделе 10.4.

В файле `lastlog` регистрируется время последнего входа в систему каждого пользователя. Это разреженный бинарный файл, записи которого индексируются по идентификатору пользователя (UID). Размер файла будет меньше, если назначать идентификаторы по порядку, хотя вряд ли об этом стоит сильно беспокоиться. Файл `lastlog` не должен участвовать в цикле ротации, поскольку его размер в большинстве случаев остается неизменным.

**Таблица 11.1. Журнальные файлы**

| Файл | Программа | Место[а] | Частота[а] | Системы[а] | Содержимое/назначение |
|---|---|---|---|---|---|
| `acpid` | `acpid` | Ф | 64k | RZ | События, связанные с питанием |
| `auth.log` | `sudo` и др.[б] | S | M | U | Авторизационные сообщения |
| `apache2/*` | `httpd` (версия 2) | Ф | Д | ZU | Журналы HTTP-сервера Apache (версия 2) |
| `apt*` | APT | Ф | M | U | Программы для установки программных пакетов |
| `boot.log` | Сценарии запуска системы | Ф[в] | M | R | Выходная информация сценариев запуска |
| `boot.msg` | Ядро | В | – | Z | Образ буфера сообщений ядра |
| `cron`, `cron/log` | `cron` | S | Н | RAH | Сведения о выполнении и об ошибках демона `cron` |
| `cups/*` | CUPS | Ф | Н | ZRU | Сообщения, связанные с печатью (CUPS) |
| `daemon.log` | Различные | S | Н | U | Все сообщения средств демона |
| `debug` | Различные | S | Д | U | Отладочные сообщения |
| `dmesg` | Ядро | В | – | RU | Образ буфера сообщений ядра |
| `dpkg.log` | `dpkg` | Ф | M | U | Журнал управления пакетом |
| `faillog`[г] | `login` | Н | Н | RZU | Сообщения о неудачных попытках регистрации в системе |
| `httpd/*` | `httpd` | Ф | Д | R | Журналы HTTP-сервера Apache (в каталоге `/etc`) |
| `kern.log` | Ядро | S | Н | U | Все сообщения средств ядра |
| `lastlog` | `login` | В | – | RZ | Время последней регистрации в системе каждого пользователя (бинарный файл) |
| `mail*` | Связанные с электронной почтой | S | Н | Все | Все сообщения средств электронной почты |
| `messages` | Различные | S | Н | RZUS | Это основной системный журнальный файл |
| `rpmpkgs` | `cron.daily` | В | Д | R | Список установленных RPM-пакетов |
| `samba/*` | `smbd` и др. | Ф | Н | – | Samba (совместное использование файлов в системах Windows/CIFS) |
| `secure` | `sshd` и др. | S | M | R | Конфиденциальные авторизационные сообщения |
| `sulog` | `su` | Ф | – | SAH | Учет удачных и неудачных попыток использования команды `su` |

*Окончание табл. 11.1*

| Файл | Программа | Место[а] | Частота[а] | Системы[а] | Содержимое/назначение |
|---|---|---|---|---|---|
| **syslog*** | Различные | S | H | SUH | Это основной системный журнальный файл |
| **warn** | Различные | S | H | Z | Все сообщения уровня предупреждений/ошибок |
| **wpars/*** | **wpar** | Ф | - | A | События загрузочных разделов |
| **wtmp** | **login** | B | M | Все | Сообщения о регистрации в системе (бинарный файл) |
| **xen/*** | Xen | Ф | 1m | RZU | Информация от монитора виртуальных машин Xen |
| **Xorg.*n*.log** | **Xorg** | Ф | H | RS | Сообщения об ошибках сервера X Windows |
| **yum.log** | **yum** | Ф | M | R | Журнал управления пакетом |

[а] В столбцах "Место", "Частота" и "Системы" используются следующие обозначения: S — Syslog, B — встроенное имя, Ф — конфигурационный файл, Д — ежедневно, H — еженедельно, M — ежемесячно, *NN*[km] = размер в килобайтах или мегабайтах, Z — SUSE, R — Red Hat и CentOS, S — Solaris, H — HP-UX, A — AIX.

[б] Команды **passwd**, **login** и **shutdown** тоже записывают информацию в журнал авторизации, который хранится в каталоге **/var/adm**.

[в] В действительности сообщения направляются в систему Syslog, но название средства и уровень важности задаются в файле **/etc/initlog.conf**.

[г] Двоичный файл, который должен быть прочитан с помощью утилиты **faillog**.

Назначение файла **utmp** — попытаться сохранить запись каждого пользователя, который регистрируется в данный момент. Иногда это выполняется неправильно: например, если оболочка была разрушена несоответствующим сигналом, а родительская оболочка не восстановлена надлежащим образом. Файл **utmp**, как правило, доступен для "всеобщей" записи.

## Особенности систем

Поставщики, казалось бы, прячут журнальные файлы где-то на диске. Многие из них можно найти посредством розыскных мероприятий, проводимых на базе файлов конфигурации демонов и файла конфигурации syslog. В этом разделе мы "прольем свет" на некоторые скрытые журнальные файлы, оставленные поставщиками в "темных углах и закоулках".

Дистрибутивы Linux заслуживают приз за самый простой вариант управления регистрацией событий. Журналы регистрации получают понятные имена и последовательно сохраняются в каталоге **/var/log**. Все наши примеры дистрибутивов также включают превосходную утилиту **logrotate**, предназначенную для выполнения ротации и усечения этих журналов, а также управления ими. Новые программные пакеты могут вставлять файл конфигурации в каталог **/etc/logrotate.d** в соответствии со стратегией управления этими журналами регистрации. Утилита **logrotate** подробно описана ниже в этой главе (см. раздел 11.4).

И наоборот, система Solaris отличается самой неорганизованной коллекцией журналов регистрации. Но использование каталога **/var/log** облегчит вашу работу. Вот несколько полезных ссылок.

- **/var/log/***
- **/var/cron/log**
- **/var/lp/logs/***
- **/var/saf/_log**
- **/var/saf/zsmon/log**
- **/var/svc/log**
- **/var/adm/***

Из демона **cron** вы можете запускать поддерживаемый поставщиками сценарий **/usr/lib/newsyslog** для выполнения ротации основных журналов регистрации **/var/adm/messages** и **/var/log/syslog**.

В системе HP-UX журналы регистрации хранятся в каталоге **/var/adm**. Здесь также находится множество странных и даже загадочных файлов, которые не являются журналами регистрации, поэтому будьте бдительны. Файл **nettl.LOG000** предназначен для управления сетью и ведения статистики (см. man-страницу **nettl**). По умолчанию регистрационные записи, формируемые системой Syslog, попадают в каталог **/var/adm/syslog**.

## 11.2. Syslog: система регистрации событий

Syslog — это полнофункциональная система регистрации событий, написанная Эриком Оллманом (Eric Allman). Она выполняет две важные функции: освобождает программистов от утомительной механической работы по ведению журнальных файлов и передает управление журнальной регистрацией в руки администраторов. До появления системы Syslog каждая программа сама выбирала схему регистрации событий, а у системных администраторов не было возможности контролировать, какая информация и где именно сохраняется.

Система Syslog отличается высокой гибкостью. Она позволяет сортировать сообщения по источникам и уровню важности и направлять их в различные пункты назначения: в журнальные файлы, на терминалы пользователей и даже на другие компьютеры. Одной из самых ценных особенностей этой системы является ее способность централизовать процедуру регистрации событий в сети.

Система Syslog была давно принята "на вооружение" основными версиями UNIX и Linux (за исключение системы AIX). Система AIX включает демон syslog и библиотечные программы, но не обеспечивает никакой стандартной конфигурации системы Syslog и использует собственный демон для формирования отчетов об ошибках (подробнее см. раздел 11.3 ниже в этой главе). Поскольку система Syslog столь широко применяется программными расширениями, мы считаем, что глубокое понимание системы Syslog важно и для администраторов AIX.

## Архитектура системы Syslog

Система Syslog состоит из трех компонентов:

- **syslogd** — демон журнальной регистрации (его конфигурационный файл называется **/etc/syslog.conf**);
- **openlog()**, **syslog()**, **closelog()** — библиотечные функции, которые передают сообщения демону **syslogd**;
- **logger** — команда пользовательского уровня, которая принимает регистрационные сообщения от интерпретатора команд.

🕮 Сигналы описаны в разделе 5.3.

Демон **syslogd** запускается на этапе начальной загрузки системы и работает непрерывно. Его работой нельзя управлять посредством демона **inetd**. Программы, взаимодействующие с системой Syslog, записывают журнальные сообщения (путем вызова библиотечной функции **syslog()**) в специальный файл **/dev/log**, который представляет собой UNIX-сокет. Демон **syslogd** читает сообщения из этого файла, сверяется со своим конфигурационным файлом и пересылает сообщения адресатам.

По сигналу "отбой" (**HUP**, сигнал 1) демон **syslogd** закрывает свои журнальные файлы, перечитывает конфигурационный файл и вновь переходит в режим непрерывной регистрации сообщений. Если файл **syslog.conf** изменился, нужно послать демону **syslogd** сигнал **HUP**, иначе изменения не вступят в силу. Сигнал **TERM** приводит к завершению работы демона.

Демон **syslogd** записывает свой идентификатор процесса (PID) в каталог **/var/run** (в системе AIX для этого используется каталог **/etc**). Это облегчает передачу демону сигналов из сценариев. Например, следующая команда посылает демону сигнал отбоя.

```
solaris$ sudo kill -HUP `/bin/cat /var/run/syslogd.pid`
```

Попытки выполнить сжатие или ротацию журнального файла, который был открыт демоном **syslogd** для записи, приводят к непредсказуемым последствиям. О выполнении ротации журнальных файлов с помощью утилиты **logrotate** можно прочитать в разделе 11.4.

Предпочтительный метод перезапуска утилиты **syslogd** в системе AIX — использование команды **refresh**.

```
aix$ sudo refresh -s syslogd
```

Команда **refresh** контактирует с контроллером системных ресурсов (System Resource Controller), который управляет различными подсистемами, в том числе и подсистемами регистрации. Подробнее см. man-страницу **refresh**.

## Конфигурирование демона syslogd

Поведение демона **syslogd** зависит от содержимого файла **/etc/syslog.conf**. Это текстовый файл относительно простого формата. Пустые строки, а также строки, начинающиеся с символа решетки (#), игнорируются. Базовый синтаксис записей файла таков.

*селектор* **<Tab>** *действие*

Например, строка

```
mail.info       /var/log/maillog
```

обеспечивает сохранение сообщений, поступающих от почтовой системы, в файле **/var/log/maillog**. Поля *селектор* и *действие* должны разделяться одним или двумя сим-

волами табуляции. Пробелы в большинстве версий не работают и скрывают ошибки, которые потом довольно трудно отследить. Такие ошибки легко появляются в результате операций вырезки и вставки, выполняемых с помощью мыши.

Селектор указывает программу ("средство"), которая посылает журнальное сообщение, и уровень важности этого сообщения. Формат селектора выглядит следующим образом.

```
средство.уровень
```

Названия средств и уровней важности следует выбирать из стандартного списка значений; программы не могут самостоятельно составлять такие списки. Есть средства, определенные для ядра, для основных групп утилит и для локальных программ. Все остальное проходит под общим названием *user* (т.е. пользователь).

Селекторы могут содержать ключевые слова * и **none**, которые обозначают "все" и "ничего" соответственно. В селекторе разрешается через запятые перечислять группу средств. Допускается также разделение самих селекторов с помощью точки с запятой.

В общем случае селекторы объединяются методом логического сложения, т.е. указанное действие будет выполняться для сообщения, соответствующего любому селектору. Селектор уровня **none** означает исключение перечисленных в нем средств, независимо от того, что указано в остальных селекторах этой же строки.

Приведем несколько примеров селекторов.

```
средство.уровень                          действие
средство1,средство2.уровень               действие
средство1.уровень1;средство2.уровень2     действие
*.уровень                                 действие
*.уровень;плохое_средство.none            действие
```

В табл. 11.2 перечислены допустимые названия средств. В настоящее время определено 21 средство.

**Таблица 11.2. Названия средств Syslog**

| Средство | Программы или сообщения, которые его используют |
|---|---|
| * | Все средства, кроме **mark** |
| **auth** | Команды, связанные с безопасностью и авторизацией |
| **authpriv** | Конфиденциальные авторизационные сообщения |
| **cron** | Демон **cron** |
| **daemon** | Системные демоны |
| **ftp** | Демон **ftpd** |
| **kern** | Ядро |
| **local0-7** | Восемь разновидностей локальных сообщений |
| **lpr** | Система спулинга построчной печати |
| **mail** | Система **sendmail** и другие почтовые программы |
| **mark** | Метки времени, генерируемые через одинаковые промежутки |
| **news** | Система телеконференций Usenet (устаревшее средство) |
| **syslog** | Внутренние сообщения демона **syslogd** |
| **user** | Пользовательские процессы (это установка по умолчанию, если не указано иное) |
| **uucp** | Устаревшее средство, которое в последних версиях системы игнорируется |

Не воспринимайте слишком серьезно различие между syslog-сообщениями типа **auth** и **authpriv**. В действительности все авторизационные сообщения являются конфиденциальными, и ни одно из них не должно быть открытым для всеобщего доступа.

Демон **syslogd** сам генерирует сообщения с метками времени, которые регистрируются, если в файле **syslog.conf** для них указан селектор со средством **mark**. Метки времени помогают точно определить, когда произошел сбой, к примеру "между 3:00 и 3:20 утра", а не просто "сегодня ночью". Это необходимо при устранении проблем, возникающих регулярно.

В табл. 11.3 перечислены уровни важности, существующие в системе Syslog (в порядке убывания важности).

**Таблица 11.3. Уровни важности сообщений Syslog**

| Уровень | Приблизительное значение |
|---|---|
| **emerg** | Экстренные сообщения |
| **alert** | Срочные сообщения |
| **crit** | Критические состояния |
| **err** | Другие ошибочные состояния |
| **warning** | Предупреждения |
| **notice** | Необычные события, которые заслуживают внимания |
| **info** | Информационные сообщения |
| **debug** | Отладочные сообщения |

Уровень сообщения определяет его важность. Границы между различными уровнями несколько размыты. Существует четкое различие между событиями, заслуживающими внимания, и предупреждениями (а также между предупреждениями и сообщениями об ошибках), но трудно однозначно разграничить значения уровней **alert** и **crit**.

В файле **syslog.conf** уровни обозначают минимальную важность, которую сообщение должно иметь для того, чтобы быть зарегистрированным. Например, сообщение почтовой системы, имеющее уровень важности **warning**, будет соответствовать селектору **mail.warning**, а также селекторам **mail.info**, **mail.notice**, **mail.debug**, ***.warning**, ***.notice**, ***.info** и ***.debug**. Если в файле **syslog.conf** указано, что сообщения **mail.info** должны регистрироваться в файле, то сообщения **mail.warning** тоже будут направляться в этот файл.

В Linux-версиях системы Syslog перед названиями уровней могут ставиться символы = и !, означающие "только этот уровень" и "за исключением этого и более высоких уровней" соответственно (табл. 11.4).

**Таблица 11.4. Примеры использования квалификаторов уровня в файле syslog.conf**

| Селектор | Назначение |
|---|---|
| **mail.info** | Выбор почтовых сообщений уровня **info** и выше |
| **mail.=info** | Выбор почтовых сообщений только уровня **info** |
| **mail.info;mail.!err** | Выбор почтовых сообщений уровней **info**, **notice** и **warning** |
| **mail.debug;mail.!=warning** | Выбор почтовых сообщений любого уровня, кроме **warning** |

Поле *действие* определяет способ обработки сообщения. Возможные варианты перечислены в табл. 11.5.

**Таблица 11.5. Действия системы Syslog**

| Действие | Назначение |
|---|---|
| *имя_файла* | Дописать сообщение в указанный файл на локальном компьютере |
| *@имя_узла* | Переслать сообщение демону **syslogd**, выполняющемуся на компьютере с указанным именем |
| *@IP_адрес* | Переслать сообщение демону **syslogd** на компьютере с указанным IP-адресом |
| \| *файл_FIFO* | Записать сообщение в указанный именованный канал[a] |
| *пользователь1, пользователь2,...* | Вывести сообщение на экраны терминалов указанных пользователей, если они зарегистрированы в системе |
| * | Вывести сообщение на экраны всех зарегистрированных пользователей |

[a] Для получения дополнительной информации введите команду **info mkfifo** (только для Linux-версий **syslogd**).

Если в качестве действия задано имя файла (или имя файла FIFO), то это должно быть полное имя. При отсутствии указанного файла в системе Linux демон **syslogd** создает его в процессе записи первого сообщения. В других системах заданный файл должен уже существовать — демон **syslogd** не будет его создавать.

В дистрибутивах Linux перед именем файла можно ставить дефис, чтобы после записи каждого сообщения не осуществлялся системный вызов **sync**. Синхронизация помогает сохранить как можно больше журнальной информации в случае сбоя системы, но она достаточно накладна с точки зрения производительности системы. Естественно, мы рекомендуем включать дефисы (тем самым запрещая синхронизацию). Удаляйте дефисы только временно, пока занимаетесь изучением проблемы, которая является причиной аварийного состояния ядра.

Если указывается не IP-адрес, а имя узла, оно должно быть известно одной из служб преобразования имен, например DNS или NIS.

В системе Solaris реализация syslog выполняет файл **syslog.conf** через макропроцессор предварительной обработки m4. Обратитесь к man-страницам и, не жалея, используйте кавычки, чтобы записи конфигурации выражали ваши намерения. Например, вы должны заключать в кавычки все ключевые слова m4 и все выражения, которые содержат запятую. Вот как выглядит типичная запись в стиле m4.

```
auth.notice     ifdef(`LOGHOST', `/var/log/authlog', `@loghost')
```

Обратите внимание на то, что используемые кавычки представляют собой одиночные "обратные апострофы". Приведенная выше строка направляет сообщения в файл **/var/log/authlog**, если значение **LOGHOST** не определено. В противном случае сообщения направляются в компьютер **loghost**. В m4 операторы **ifdef** очень мощные; они позволяют системным администраторам создавать один файл **syslog.conf**, который может быть использован на всех компьютерах.

Хотя в селекторе можно указывать сразу несколько средств и уровней, выполнение нескольких действий не предусмотрено. Для того чтобы послать сообщение в два разных места (например, в локальный файл и на центральный регистрационный узел), нужно включить в конфигурационный файл две строки с одинаковыми селекторами.

Поскольку сообщения системы Syslog могут использоваться для организации атак типа "отказ от обслуживания", демон **syslogd** в большинстве дистрибутивов Linux откажется принимать сообщения от других компьютеров, если только он не был запущен с флагом **-r**. По умолчанию демон также отказывается выступать в роли внешнего диспетчера сообщений: сообщения, поступающие от сетевого узла, не могут быть переслены на другой узел. Флаг **-h** отменяет такое поведение. Если нужно, чтобы эти опции были всегда включены, модифицируйте соответствующим образом сценарий запуска syslog. В RHEL конфигурацию syslog необходимо отредактировать в **/etc/sysconfig/syslog**.

## Примеры конфигурационных файлов

Поскольку понять содержимое файла **syslog.conf** несложно, мы не станем подробно анализировать конфигурационные файлы наших демонстрационных дистрибутивов. Гораздо полезнее будет рассмотреть некоторые распространенные схемы журнальной регистрации.

Ниже приведены примеры трех файлов **syslog.conf**, которые соответствуют автономному компьютеру, используемому в небольшой сети, клиентскому компьютеру в крупной сети и центральному регистрационному узлу этой же сети. Последний называется **netloghost**[1].

### *Автономный компьютер*

Ниже показана базовая конфигурация для автономного компьютера.

```
# Файл syslog.conf для небольшой сети или автономной системы.
# Экстренные сообщения направляются всем зарегистрированным пользователям.
*.emerg[2]                                                    *
# важные сообщения
*.warning;daemon,auth.info;user.none    /var/log/messages
# ошибки принтера
lpr.debug                               /var/log/lpd-errs
```

В первой не являющейся комментарием строке организуется вывод экстренных сообщений на экраны терминалов всех пользователей, зарегистрированных в данный момент в системе. В качестве примера можно привести сообщения, выдаваемые командой **shutdown** перед выключением системы.

Вторая значащая строка обеспечивает запись важных сообщений в файл **/var/log/messages**. Уровень **info** ниже уровня **warning**, поэтому выражение **daemon,auth.info** включает дополнительную регистрацию сообщений, поступающих от команд **passwd** и **su**, а также от всех демонов. Третья строка предназначена для записи сообщений об ошибках принтера в файл **/var/log/lpd-errs**.

### *Сетевой регистрационный клиент*

Как показано в следующем примере, клиентский компьютер обычно пересылает важные сообщения на центральный регистрационный узел.

[1] Точнее, ***netloghost*** — один из псевдонимов узла. Наличие псевдонима позволяет заменять регистрационный узел с минимальными затратами на переконфигурирование системы. Псевдоним можно добавить в файл ***/etc/hosts*** или задать с помощью записи ***CNAME*** в базе данных DNS. Подробнее о записях ***CNAME*** рассказывается в разделе 17.8.

[2] Если пользователи оконной системы X не выполняют программу ***xconsole***, они не получат эти сообщения.

```
# Файл syslog.conf для клиентских компьютеров
# Экстренные сообщения направляются всем зарегистрированным пользователям.
*.emerg;user.none                              *
# Важные сообщения пересылаются на центральный компьютер.
*.warning;lpr,local1.none          @netloghost
daemon,auth.info                   @netloghost
# Сообщения от локальных программ отправляются на центральный компьютер.
local2.info;local7.debug           @netloghost
# Сообщения об ошибках принтера хранятся локально.
lpr.debug                          /var/log/lpd-errs
# Сообщения утилиты sudo регистрируются от имени средства local2.
# Сохраняем их копии.
local2.info                        /var/log/sudo.log
# Сообщения ядра также хранятся локально.
kern.info                          /var/log/kern.log
```

В такой конфигурации локально хранится не очень много информации. Если компьютер **netloghost** выключен или недоступен, регистрационные сообщения будут безвозвратно утеряны. На этот случай можно создавать локальные копии важных сообщений.

В системе, где инсталлирован большой объем локального программного обеспечения, неоправданно много сообщений будет регистрироваться от имени средства **user** на уровне **emerg**. В показанном примере такая ситуация заведомо исключается благодаря выражению **user.none** в первой значащей строке.

🕮 Подробнее об утилите **sudo** рассказывалось в разделе 4.3.

Вторая и третья значащие строки служат для пересылки всех важных сообщений на центральный регистрационный сервер; исключение составляют сообщения подсистемы печати и университетской (локальной) системы доступа по магнитным карточкам. В четвертой строке организуется отправка локальной журнальной информации также на центральный компьютер. Последние три строки обеспечивают сохранение локальных копий сообщений об ошибках принтера, журнальных сообщений утилиты **sudo** и сообщений ядра.

### Центральный регистрационный узел

Ниже показана конфигурация компьютера **netloghost** — центрального защищенного регистрационного узла сети среднего размера, включающей около 7000 компьютеров.

```
# Файл syslog.conf для главного регистрационного узла.
# Экстренные сообщения направляются на консоль и в журнальный файл,
# дополняемые метками времени.
*.emerg                                        /dev/console
*.err;kern,mark.debug;auth.notice              /dev/console
*.err;kern,mark.debug;user.none                /var/log/console.log
auth.notice                                    /var/log/console.log

# Сообщения, не являющиеся экстренными,
# записываются в обычные журнальные файлы.
*.err;user.none;kern.debug                     /var/log/messages
daemon,auth.notice;mail.crit                   /var/log/messages
lpr.debug                                      /var/log/lpd-errs
mail.debug                                     /var/log/mail.log
# В следующих строках обрабатываются сообщения локальных программ
# авторизации, таких как sudo и npasswd.
local2.debug                                   /var/log/sudo.log
```

```
local2.alert                                    /var/log/sudo-errs.log
auth.info                                       /var/log/auth.log

# другие локальные программы
local0.info                                     /var/adm/nbl.log
local4.notice                                   /var/admlog/da.log
local6.debug                                    /var/adm/annex-isn.log
local7.debug                                    /var/admlog/tcp.log

# локальные сообщения (стандартное правило обработки сообщений,
# для которых не задано средство)
user.info                                       /var/admlog/user.log
```

Сообщения, поступающие от локальных программ и демонов **syslogd** по сети, записываются в журнальные файлы. В некоторых случаях выходная информация каждого средства помещается в отдельный файл.

Центральный регистрационный компьютер генерирует метки времени для каждого получаемого сообщения. Это время не всегда совпадает со временем отправки сообщения. Если в систему входят компьютеры, расположенные в разных часовых поясах, или системные часы не синхронизированы, информация о времени окажется недостоверной.

## Отладка системы Syslog

Команда **logger** позволяет выдавать журнальные сообщения в сценариях интерпретатора команд. Кроме того, эту команду можно использовать для проверки изменений в конфигурационном файле демона **syslogd**. Например, если в файл была добавлена строка

```
local5.warning          /tmp/evi.log
```

и нужно убедиться в том, что она действительна, введите такую команду.

```
hp-ux$ logger -p local5.warning "test message"
```

Строка, содержащая фразу "test message" (тестовое сообщение), должна быть записана в файл **/tmp/evi.log**. Если этого не произошло, то, возможно, вы забыли создать файл **evi.log**, предоставить соответствующие права доступа к этому файлу или послать демону **syslogd** сигнал отбоя. А, может быть, вы использовали пробелы вместо табуляции?

## Альтернатива системе Syslog

Хотя система Syslog — проверенное средство регистрации событий в системах UNIX и Linux, в попытках устранения ряда ее недостатков были разработаны несколько альтернативных систем. Одна из них — Syslog-ng ("Syslog, next generation" — Syslog следующего поколения) — теперь по умолчанию используется в системах SUSE. С точки зрения конфигурирования, она достаточно сильно отличается от Syslog, и мы не станем подробно ее описывать в этой книге. Если захотите ее использовать в системе, отличной от SUSE, она доступна по адресу: `balabit.com`.

Система Syslog-ng предоставляет дополнительные возможности конфигурирования, фильтрования, в зависимости от содержимого сообщений, обеспечения целостности сообщений и улучшенную поддержку ограничений брандмауэра при пересылке сообщений по сети.

Система SDSC Secure Syslog (разработанная в центре суперкомпьютеров в Сан-Диего) известна также как высокопроизводительная система Syslog. Реализуя спецификации RFC3195 (Reliable Delivery for syslog), она предоставляет "признаваемую судебными органами" систему аудита событий. Ее разработка выполнялась специально для организаций

с высоким сетевым трафиком, и система содержит множество компонентов оптимизации производительности. Ее исходный код можно загрузить из сайта SourceForge:

`sourceforge.net/projects/sdscsyslog`.

Система Rsyslog — еще одна мощная альтернатива следующего поколения — включается в стандартные поставки ряда популярных дистрибутивов Linux (в том числе Fedora 8). Rsyslog позиционирует себя как расширенный модуль **syslogd** для систем Linux и Unix, обладающий продвинутой многопоточностью и акцентированный на безопасности и надежности. Rsyslog поддерживает удаленную передачу логов на другой сервер посредством TCP (в отличие от протокола UDP, используемого оригинальным демоном **syslog**) и может взаимодействовать с протоколом защищенных сокетов SSL (Secure Sockets Layer), использование которого в некоторых системах может быть выдвинуто в качестве обязательного условия по регулятивным причинам. Через подключаемые расширения Rsyslog даже может записывать системные логи сразу в базу данных (MySQL, Posrgres SQL, Oracle и множество других), а не только в файл. Rsyslog — оптимальное решение, функционально превосходящее свои аналоги и легко осваиваемое новичками. Больше о возможностях Rsyslog можно узнать на сайте **rsyslog.com**.

## Журнальная регистрация на уровне ядра и на этапе начальной загрузки

Ядро и сценарии запуска системы представляют собой немалую проблему с точки зрения журнальной регистрации. В первом случае проблема заключается в том, чтобы средства регистрации событий, происходящих на этапе начальной загрузки и в ядре, не зависели от типа файловой системы и ее организации. Во втором случае трудность состоит в последовательной регистрации событий, происходящих в сценариях, а не в системных демонах или других программах, — их сообщения должны регистрироваться на другом уровне. Необходимо также избежать ненужного дублирования записей в сценариях и переадресации их вывода.

Первая из описанных проблем решается путем создания внутреннего буфера ограниченного размера, куда ядро заносит свои журнальные сообщения. Этот буфер достаточно велик для того, чтобы вместить сообщения обо всех действиях, предпринимаемых ядром на этапе начальной загрузки. Когда система полностью загрузится, пользовательский процесс сможет получить доступ к журнальному буферу ядра и скопировать его содержимое в нужное место. Обычно для этого запускается команда **dmesg**, результаты работы которой даже содержат сообщения, которые были сгенерированы до запуска демона **init**.

Сообщения, генерируемые ядром в процессе работы, обрабатываются демоном **klogd**. С функциональной точки зрения он расширяет возможности команды **dmesg**. Демон может или просто создать образ журнального буфера ядра и завершить работу, или динамически извлекать сообщения из буфера по мере их появления, записывая их в файл или передавая системе Syslog. По умолчанию демон работает во втором режиме. Его сообщения обрабатываются системой Syslog в соответствии с установками для средства "kern" (обычно они записываются в каталог **/var/log/messages** или **/var/log/syslog**).

В наших демонстрационных дистрибутивах сценарии запуска не используют флаг **-c** команды **dmesg** при создании начального образа журнального буфера, вследствие чего содержимое буфера читается, но не сбрасывается. Когда демон **klogd** запускается, он

находит в буфере те же сообщения, которые прочла команда **dmesg**, и передает их системе Syslog. По этой причине некоторые сообщения появляются не только в файле **dmesg** или **boot.msg**, но и в основном syslog-файле системы.

Другой важный момент, связанный с журнальной регистрацией на уровне ядра, касается правильного управления системной консолью. Важно, чтобы по ходу загрузки системы вся выводимая информация попадала на консоль. С другой стороны, когда система загрузилась и начала выполняться, консольные сообщения больше раздражают, нежели помогают, особенно если консоль используется для регистрации в системе.

Команда **dmesg** и демон **klogd** позволяют указывать в командной строке уровень важности консольных сообщений ядра.

```
ubuntu$ sudo dmesg -n 2
```

Сообщения уровня 7 наиболее информативны и включают отладочную информацию. К первому уровню относятся только сообщения о фатальных ошибках системы (чем ниже уровень, тем выше важность сообщений). Все сообщения, выдаваемые ядром, продолжают поступать в журнальный буфер (а также в систему Syslog) независимо от того, направляются они на консоль или нет.

В каталог **/proc/sys** ядро помещает также ряд управляющих файлов, которые позволяют отсекать повторяющиеся журнальные сообщения. Механизм настройки параметров ядра подробнее описан в подразделе 13.3. Специальными управляющими файлами являются **/proc/sys/kernel/printk_ratelimit**, в котором указан минимальный интервал времени в секундах между записываемыми сообщениями ядра после активизирования "заглушки" (по умолчанию это значение равно 5), и **/proc/sys/kernel/printk_ratelimit_burst**, в котором указано количество групповых сообщений, вызывающее активизирование заглушки (значение по умолчанию — 10). Эти параметры носят лишь рекомендательный характер и поэтому не обеспечивают абсолютной гарантии полного перекрытия потока излишних сообщений. Кроме того, они применяются только к сообщениям, созданным в ядре с помощью функции **printk_ratelimit()**.

К сожалению, журнальная регистрация на уровне сценариев запуска системы контролируется не так хорошо, как на уровне ядра.

В Red Hat Enterprise Linux имеется команда **initlog**, которая перехватывает сообщения, выдаваемые командами сценариев, и направляет их в систему Syslog. К сожалению, эту команду приходится явно указывать в сценариях, что повышает их сложность. В конечном итоге журнальные сообщения попадают в файл **/var/log/boot.log**.

В других наших дистрибутивах вообще не делаются попытки перехватывать сообщения сценариев запуска. Часть информации регистрируется отдельными командами и демонами, но основная ее часть остается "незамеченной".

## 11.3. Регистрация сообщений и обработка ошибок в системе AIX


В системе AIX управление журналами регистрации отличается от других систем UNIX. Несмотря на наличие системы Syslog в стандартной поставке, она не сконфигурирована. При формировании отчета о системных ошибках AIX опирается на демон **errdemon**, который предназначен для обработки системных диагностических сообщений (таких, как уведомления о сбоях оборудования

или всей файловой системы), но не для обработки регистрирующих сообщений для отдельных демонов. Таким образом, предусмотрительный системный администратор воспользуется мудростью демона **errdemon** для AIX-диагностики и возможностями правильно сконфигурированной системы Syslog для управления журналами регистрации централизированных приложений.

Демон **errdemon** запускается при начальной загрузке системы в каталоге **/etc/rc.bootc** и читает события, связанные с ошибками, из специального файла **/dev/error**. Как ядро, так и некоторые пользовательские AIX-приложения записывают данные об ошибках в этот файл в соответствии с предопределенными шаблонами из каталога **/dev/adm/ras/errtmplt**. Демон **errdemon** сравнивает новые записи с шаблонами и записывает результат в двоичном формате в файл **/var/adm/ras/errlog**. Система AIX любит двоичные форматы!

Файл **errlog** — циркулярный, поэтому, когда файл достигает своего максимального размера (1 Мбайт по умолчанию), на место первого события записывается новое. Кроме того, демон **errdemon** буферизирует события, которые еще не были записаны в журнал регистрации. Можно просмотреть или изменить настройки, запустив непосредственно демон **/usr/lib/errdemon**. Детали инициирования работы демона можно узнать на соответствующей man-странице.

Поскольку файл **errlog** — не текстовый, для чтения его содержимого вы можете использовать другой инструмент **errpt**. Без аргументов утилита **errpt** напечатает в сокращенной форме список всех событий, записанных в журнале регистрации. Для получения более подробного варианта того же списка добавьте аргумент **-a**. Вот как выглядит пример записи из нашей системы AIX.

```
aix$  errpt -a
---------------------------------------------------------------------------
LABEL:                  DMPCHK_NOSPACE
IDENTIFIER:             F89FB899

Date/Time:              Sat Mar 21 15:00:01 MST 2009
Sequence Number:        224
Machine Id:             0001A4C4D700
Node Id:                ibm
Class:                  O
Type:                   PEND
WPAR:                   Global
Resource Name:          dumpcheck

Description
The copy directory is too small.

Probable Causes
There is not enough free space in the file system containing the copy
directory to accommodate the dump.

        Recommended Actions
        Increase the size of that file system.

Detail Data
File system name
/var/adm/ras
```

```
Current free space in kb
   108476
Current estimated dump size in kb
   197836
-------------------------------------------------------------------------------
```

Это конкретное событие означает, что системный дамп не поместится в заданной файловой системе приемника. Большинство меток разделов не требует дополнительных разъяснений, но за дополнительной информацией по **errpt** вам стоит все же обратиться к соответствующей man-странице.

Несмотря на то что демон **errdemon** представляет собой полезный источник данных о регистрации событий в автономной системе AIX, его использование может противоречить получившей более широкое определение стратегии регистрирования событий на уровне предприятия. Вам, возможно, придется написать сценарий для перехвата событий демона **errdemon** в формате системы Syslog или перенаправлять их для централизованного архивирования. Кроме того, из оперативной IBM-документации вы можете узнать, как отправлять отчеты об ошибках в систему Syslog через менеджер ODM (Object Data Manager).

Возможно, вам придется удалять записи из журналов регистрации ошибок, и для этой цели специалисты из IBM предусмотрели команду **errclear**. Эта команда удаляет все сообщения по истечении количества дней, заданного в качестве аргумента. Например, при выполнении команды **errclear 7** будут удалены сообщения об ошибках, с момента регистрации которых прошло более одной недели.

Чтобы стереть все сообщения об ошибках, выполните команду **errclear 0**, а чтобы стереть заданное сообщение — команду **errclear -j** *идентификатор* **0**.

## Конфигурация системы Syslog в среде AIX

По умолчанию AIX-файл **syslog.conf** содержит длинный список комментариев и не имеет никаких анализируемых конфигурационных данных. Системный администратор вправе сам принять решение о конфигурировании системы Syslog в соответствии с настройками в других системах.

Всегда не такая, как все, система AIX предоставляет свое средство циркуляции регистрационной информации в рамках демона **syslogd**. Журналы регистрации могут подвергаться циркуляции через регулярные интервалы времени или по достижении ими заданного размера. Их можно сжимать и архивировать в определенных каталогах. И хотя мы ценим эти функциональные возможности, они не могут управлять файлами, которые находятся вне контроля системы Syslog (например, журналы регистрации, генерируемые приложениями, которые "ничего не знают" о системе Syslog). Чтобы реализовать всестороннее управление журналами регистрации, вам, возможно, придется сочетать средства ротации демона **syslogd** с одним из инструментов, описываемых ниже в этой главе.

Для воспроизведения конфигурации системы Syslog в стиле Linux присоедините следующие строки к концу файла **/etc/syslog.conf** и выполните команду **refresh -s syslogd**. Не забудьте использовать символы табуляции вместо пробелов и создать каждый файл заранее.

```
mail.debug      /var/log/mail
user.debug      /var/log/user
kern.debug      /var/log/kern
syslog.debug    /var/log/messages
daemon.debug    /var/log/daemon
```

```
auth.debug      /var/log/secure
local2.debug    /var/log/sudo
```

Вы задаете ротацию журналов регистрации в файле **syslog.conf** путем присоединения опции **rotate** к концу строки конфигурации. Журналы регистрации могут подвергаться ротации по достижении заданного размера файла или по истечении заданного интервала времени (временного инкремента). Если вы установите ограничение как по размеру, так и по времени, демон **syslogd** выполнит ротацию файла при реализации любого критерия. Более того, файлы могут быть сжаты или заархивированы с переносом в новый каталог. Возможные опции перечислены в табл. 11.6.

**Таблица 11.6. Опции ротации AIX-журналов регистрации, задаваемые в файле syslog.conf**

| Опция | Значение |
|---|---|
| **rotate** | Означает, что заданный файл должен быть подвергнут ротации |
| **size** *N*[**km**][a] | Выполняет ротацию, когда файл достигает заданного размера[б] |
| **files** *N* | Сохраняет заданное количество версий в ротации |
| **time** *N*[**hdwmy**][a] | Выполняет ротацию после истечения заданного интервала времени[б] |
| **compress** | Сжимает файл после ротации с помощью команды **compress** |
| **archive** *позиция* | Перемещает файл после ротации в заданную позицию |

[a] **k** — килобайты, **m** — мегабайты.

[б] Между элементом *N* и единицами измерения не должно быть пробелов. Например, **3m** — правильная запись, а **3 m** — нет.

[в] **h** — часы, **d** — дни, **w** — недели, **m** — месяцы, **y** — годы.

Например, вот несколько строк файла конфигурации **syslog.conf** из предыдущего примера, который был расширен опциями ротации.

```
# Rotate at 500MB, keep 4 files
mail.debug   /var/log/mail      rotate size 500m files 4

# Rotate after 1 week, keep 10 files, compress the file
user.debug   /var/log/user      rotate files 10 time 1w compress

# Rotate after 100KB or 2 months, whichever occurs first, keeping 4 files
kern.debug   /var/log/kern      rotate size 100k files 4 time 2m

# Keep 1 year of weekly logs, compress the file, move the file to /logs
syslog.debug /var/log/messages rotate files 52 time 1w compress archive /logs
```

## 11.4. Утилита LOGROTATE: управление журнальными файлами

Прекрасная утилита **logrotate** реализует различные схемы управления журнальными файлами и является частью всех рассматриваемых нами дистрибутивов Linux. Она также выполняется в системах Solaris, HP-UX и AIX, но для этого вам придется ее инсталлировать. Мы отдаем предпочтение утилите **logrotate**, а не пакету **logadm**, который входит в поставку системы Solaris.

Конфигурационный файл утилиты **logrotate** состоит из набора спецификаций для групп журнальных файлов, которыми следует управлять. Параметры, определенные вне

какой-либо спецификации (например, **errors**, **rotate** и **weekly** в показанном ниже примере), являются глобальными. Их можно переопределять для конкретного журнального файла, а также указывать несколько раз в различных частях файла, меняя стандартные установки.

Ниже представлен несколько искусственный пример такого файла.

```
# Глобальные параметры
errors errors@book.admin.com
rotate 5
weekly

# Определения и параметры ротации журнальных файлов
/var/log/messages {
   postrotate
      /bin/kill -HUP `cat /var/run/syslogd.pid`
   endscript
}

/var/log/samba/*.log {
   notifempty
   copytruncate
   sharedscripts
   postrotate
      /bin/kill -HUP `cat /var/lock/samba/*.pid`
   endscript
}
```

В этой конфигурации раз в неделю осуществляется ротация файла **/var/log/messages**. Сохраняются пять последовательных версий файла, а демон **syslogd** уведомляется о каждом шаге ротации. Журнальные файлы системы Samba (их может быть несколько) также подвергаются ротации каждую неделю, но вместо того чтобы последовательно перемещать файлы, а затем создавать новый начальный файл, сценарий последовательно копирует журнальные файлы, усекая первый. Сигнал **HUP** посылается демонам Samba только после того, как произойдет ротация всех журнальных файлов.

В табл. 11.7 перечислены наиболее полезные параметры файла **logrotate.conf**.

**Таблица 11.7. Параметры файла logrotate.conf**

| Опция | Назначение |
|---|---|
| **compress** | Сжимать все версии журнального файла, кроме текущей |
| **daily**, **weekly**, **monthly** | Осуществлять ротацию журнальных файлов ежедневно, еженедельно или ежемесячно |
| **delaycompress** | Сжимать все версии журнального файла, кроме текущей и предыдущей |
| **endscript** | Этот параметр помечает конец блоков **prerotate** и **postrotate** |
| **errors** *почтовый_адрес* | Направлять сообщения об ошибках по указанному адресу |
| **missingok** | Не выдавать предупреждений об отсутствии журнального файла |
| **notifempty** | Не осуществлять ротацию журнального файла, если он пуст |
| **olddir** *каталог* | Помещать старые версии журнального файла в указанный каталог |
| **postrotate** | Этот параметр помечает начало блока команд, выполняемых после ротации журнального файла |
| **prerotate** | Этот параметр помечает начало блока команд, выполняемых перед ротацией журнального файла |

*Окончание табл. 11.7*

| Опция | Назначение |
|---|---|
| **rotate** *n* | Включать в цикл ротации указанное число версий журнального файла |
| **sharedscripts** | Запускать сценарии один раз для всей группы журнальных файлов |
| **size=***порог* | Выполнять ротацию, если размер журнального файла превышает пороговую величину (например, 100 Кбайт, 4 Мбайт) |

Утилита **logrotate** обычно запускается демоном **cron** раз в день. Ее стандартный конфигурационный файл называется **/etc/logrotate.conf**, но в командной строке можно указывать сразу несколько файлов (или каталогов, содержащих конфигурационные файлы). Эта особенность широко используется в дистрибутивах Linux, в которых каталог **/etc/logrotate.d** определен как стандартное место хранения конфигурационных файлов утилиты **logrotate**. Программные пакеты, поддерживающие эту утилиту (а таковых немало), при инсталляции выбирают схему журнальной регистрации, что существенно упрощает их администрирование.

Кроме утилиты **logrotate**, дистрибутив Ubuntu включает также более простую программу **savelog**, которая управляет ротацией отдельных файлов. Она проще утилиты **logrotate** и не использует конфигурационный файл (да и не нуждается в нем). В некоторых пакетах предпочтение отдается собственным конфигурациям, создаваемым с помощью утилиты **savelog**, а не определенным утилитой **logrotate**.

## 11.5. Поиск полезной информации в журнальных файлах

Система Syslog прекрасно подходит для сортировки и маршрутизации журнальных сообщений, но в результате все равно получается большой набор журнальных файлов. Они могут содержать много важной информации, но иногда ее не так просто найти. Необходимо дополнительное программное обеспечение для анализа журнальных файлов и поиска в них нужных сообщений.

В этом сегменте рынка предлагается много бесплатных продуктов, и большинство из них работает примерно одинаково: программа сканирует недавние записи журнального файла, сравнивает их с регулярными выражениями из базы данных и обрабатывает важные сообщения. Различия между программами проявляются в степени гибкости и размере готовых баз данных с шаблонами.

Двумя наиболее распространенными программами обработки журнальных файлов являются утилиты **swatch** и **logcheck**. Их можно загрузить из сайта `sourceforge.net` (программа **logcheck** распространяется в составе пакета **sentrytools**: `sourceforge.net/projects/sentrytools`).

Утилита **swatch** представляет собой Perl-сценарий, работающий в соответствии с директивами конфигурационного файла. Синтаксис этого файла достаточно гибок, так как позволяет выполнять все поддерживаемые в языке Perl операции сравнения с шаблонами. Утилита **swatch** может обработать конфигурационный файл за один вызов, но обычно она выполняется в фоновом режиме, отслеживая новые сообщения по мере их поступления, подобно команде **tail -f**. Недостатком является то, что конфигурационный файл, по сути, нужно создавать с нуля. Утилита не "знает" об особенностях работы системы и о том, какие журнальные сообщения в ней генерируются.

Утилита **logcheck** — более простой сценарий интерпретатора **sh**. В дистрибутив входит также программа на языке C, которая помогает утилите запоминать текущую позицию в журнальном файле. Благодаря этому у сообщения меньше шансов пройти незамеченным на этапе загрузки или останова системы. Утилита может запускаться периодически с помощью демона **cron**, а не выполняться постоянно.

С утилитой **logcheck** поставляются базы шаблонов для нескольких версий UNIX и Linux. Даже если в самой утилите нет особой надобности, стоит просмотреть эти базы данных, так как в них могут содержаться очень интересные и полезные шаблоны.

Недостаток этих утилит заключается в том, что за раз они обрабатывают только один журнальный файл. Если система Syslog направляет сообщения одновременно в несколько файлов, приходится дублировать некоторые из сообщений в каком-нибудь центральном файле, который часто обнуляется, а затем передавать его на последующую обработку одному из сценариев. Это проще, чем организовывать сложную сеть сценариев, управляющих несколькими журнальными файлами.

Централизованная система ведения и анализа журналов регистрации Splunk (**splunk.com**) объединяет регистрационные и статусные сообщения от многих разных источников в одну базу данных сообщений, доступную для поиска. Базовая версия этой системы распространяется бесплатно.

Утилита SEC (Simple Event Correlator — простой коррелятор событий) представляет собой иной тип средства управления журнальными файлами. Она является сценарием Perl, который считывает строки из файлов, именованных каналов или из стандартного ввода и преобразует их в различные классы "входных событий", сравнивая их с регулярными выражениями. Затем в соответствии с правилами, определенными в конфигурации, программа преобразует входные события в такие выходные события, как выполнение конкретного сценария или передача сообщения указанному каналу или файлу.

Дистрибутив SEC с подробной man-страницей и множеством примеров доступен по адресу: **kodu.neti.ee/~risto/sec**. Дополнительные примеры доступны на веб-сайте. Пакет SEC не является "полностью готовой к использованию" утилитой, как рассмотренные выше программы, но он может послужить хорошей отправной точкой для создания специализированного средства анализа журнальных данных.

Независимо от того, какая программа используется для сканирования журнальных файлов, есть ряд общих закономерностей.

- Большинство сообщений, связанных с безопасностью, должно просматриваться немедленно. Важно отслеживать неудачные попытки входа в систему, равно как и неудачные вызовы команд **su** и **sudo**, чтобы предотвратить возможный взлом системы, пока еще не поздно. Если кто-то просто неправильно ввел свой пароль (как чаще всего бывает), то оперативное предложение помощи произведет хорошее впечатление и повысит репутацию системного администратора.
- Сообщения о нехватке места на диске должны помечаться и немедленно обрабатываться. Когда диск переполнен, работать с ним невозможно.
- Многократно повторяемые события заслуживают внимания, хотя бы в профилактических целях.

## 11.6. Методы обработки журнальных файлов

За прошедшие годы регистрационные события вышли из области системного администрирования и превратились в сложнорешаемые проблемы в сфере управления

событиями на уровне предприятий. Обработка событий, связанных с безопасностью, стандарты информационных технологий и законодательные акты могут потребовать целостного и систематического подхода к управлению данными регистрации.

Данные журналов регистрации от одной системы создают относительно небольшую нагрузку на устройства хранения, но ведение централизованного журнала регистрации событий от сотен серверов и десятков приложений — совсем другая история. В основном, благодаря критически важной природе веб-служб, журналы регистрации, создаваемые приложениями и демонами, по значимости вышли на уровень журналов регистрации, генерируемых операционными системами.

Разрабатывая стратегию обработки журнальных файлов, ответьте на следующие вопросы.

- Сколько систем и приложений вы предполагаете использовать?
- Инфраструктура хранения информации какого типа вам доступна?
- Как долго вы собираетесь хранить журналы регистрации?
- Какого типа события являются для вас значимыми?

Ответы на эти вопросы зависят от требований, предъявляемых к бизнесу, и от применяемых вами стандартов или нормативных документов. Например, один стандарт от Совета по стандартам безопасности индустрии платежных карт (Payment Card Industry Security Standards Council — PCI SSC) требует, чтобы журналы регистрации хранились на носителе с простым доступом к данным (например, это может быть локально монтируемый жесткий диск) в течение трех месяцев и архивировались для долговременного хранения в течение, по крайней мере, одного года. Кроме того, этот же стандарт включает конкретные требования к типам используемых данных.

Как бы вы ни ответили на предложенные выше вопросы, обязательно соберите входную информацию из отделов по соблюдению правил и информационной безопасности, если таковые существуют в вашей организации.

Системы UNIX и UNIX-приложения предусматривают использование легко конфигурируемых параметров регистрации и аудита. В зависимости от объемов использования, возможно, вам придется сократить словесное наполнение журналов регистрации. И наоборот, какое-нибудь очень важное приложение может потребовать дополнительных данных событийного характера. Для большинства приложений обычно рассматривают, как минимум, сбор следующей информации:

- имя пользователя или его идентификатор (ID);
- результат обработки события (успешная или нет);
- адрес источника для сетевых событий;
- дата и время (из такого официального источника, как NTP);
- оповещение о добавлении, изменении или удалении важных данных;
- подробные данные о событии.

Большинство современных систем имеет тенденцию к использованию централизированного подхода к накоплению и анализу журнальных файлов. Такая централизация имеет ряд преимуществ: упрощенные требования к хранению информации, более простые схемы автоматизированного анализа и предупреждений, а также усовершенствованная система безопасности. Кроме того, копирование событий в центральную систему способствует улучшению целостности журналов регистрации, поскольку она менее уязвима для взломщиков.

🕮 Дополнительная информация о RAID приведена в разделе 8.7.

Сервер регистрации должен работать в соответствии с тщательно разработанной стратегией хранения информации. Например, журналы регистрации могут храниться в локальном RAID-массиве в течение 30 дней, в локально монтируемой сети устройств хранения данных (Storage Area Network — SAN) — еще в течение года и, наконец, после архивирования — на лентах для включения в ротацию резервных копий на уровне предприятия — еще в течение трех лет. Требования к хранению данных могут со временем меняться, и реализация может легко и успешно адаптироваться к изменению этих условий.

Доступ к централизированным серверам регистрации должен быть ограничен системными администраторами высокого уровня, программами и персоналом, задействованным в решении вопросов безопасности и адресной совместимости. В рамках организаций эти системы обычно не играют главную роль при принятии ежедневных решений за рамками удовлетворения требований к возможности проведения аудита, поэтому администраторы приложений, конечные пользователи и служба технической поддержки не имеют доступа к ним. Доступ к регистрационным файлам на центральных серверах должен регистрироваться отдельно.

Организация централизированной регистрации требует определенных усилий, и в небольших системах она может не продемонстрировать преимущества от использования сети. Для обеспечения централизации мы предлагаем в качестве разумной пороговой величины использовать двадцать серверов. При меньшем количестве необходимо гарантировать, что журналы регистрации подвергаются надлежащей ротации и архивированию с частотой, достаточной для того, чтобы избежать заполнения диска. Включите регистрационные файлы в решение по мониторингу, чтобы вы были извещены в случае, если некоторый журнал регистрации перестанет увеличиваться в размере.

## 11.7. Упражнения

11.1. Зачем хранить старые журнальные файлы?

11.2. В чем разница между файлами **`lastlog`** и **`wtmp`**? Какой должна быть схема ротации для каждого из них?

11.3. Проанализируйте следующую запись файла **`syslog.conf`**:

```
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
```

Есть ли в ней смысл?

11.4. Поищите в журнальных файлах записи от службы SSH. Какие события регистрируются при успешной попытке входа в систему? А при неудачной попытке? Какие действия следовало бы предпринять для увеличения словесного наполнения процесса регистрации демона SSH?

11.5. Многие стандарты и нормативные документы в индустрии информационных технологий налагают определенные требования, связанные с регистрацией или аудитом. Выберите один из таких стандартов и рассмотрите варианты настройки конфигурации системы Syslog для достижения соответствия выдвигаемым требованиям.

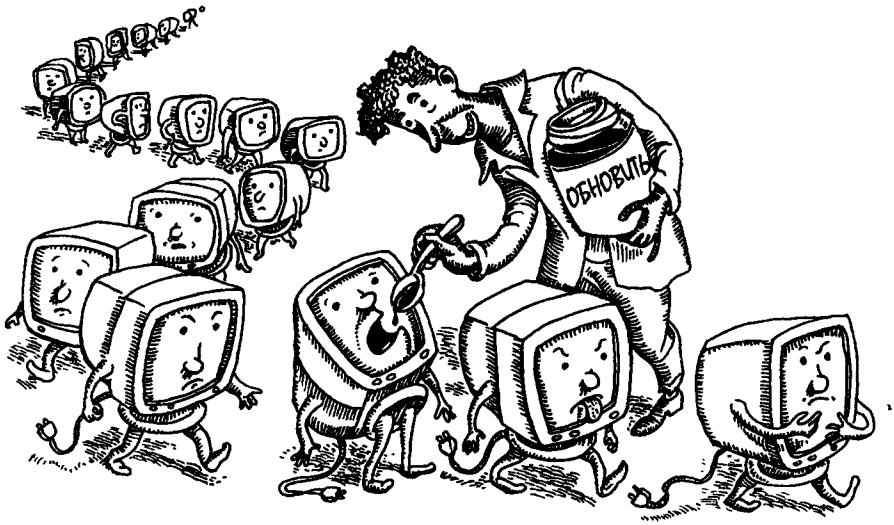
11.6. ★ Где расположен журнал начальной загрузки на вашем Linux-компьютере? В чем трудности журнальной регистрации на этапе начальной загрузки? Как эти трудности решаются демоном **`klogd`**?

11.7. ★ Проанализируйте схему журнальной регистрации, используемую в вашей организации, включая схему ротации журнальных файлов. Сколько дискового пространства отведено для журнальных файлов? Насколько долго хранятся журнальные файлы? Можно ли заранее предсказать ситуации, когда используемых схем окажется недостаточно? Какое решение можно порекомендовать? (Необходим доступ с правами суперпользователя.)

11.8. ★ Некоторые журнальные сообщения являются чрезвычайно важными и должны немедленно просматриваться системным администратором. Что можно предпринять для того, чтобы это произошло как можно скорее?

# Глава 12

# Управление программным обеспечением и конфигурацией




Инсталляция программных средств, их конфигурирование и управление ими — основные обязанности системных администраторов. Они отвечают на запросы пользователей инсталлировать и сконфигурировать программы, усовершенствовать средства защиты их данных и устранить прорехи в системе безопасности, а также управляют переходом к новым версиям программ, которые, с одной стороны, предлагают новые возможности, а, с другой стороны, чреваты проблемами несовместимости. Администраторам обычно приходится выполнять перечисленные ниже задачи:

- осуществлять автоматизированную инсталляцию операционной системы на группу компьютеров;
- выполнять настройку операционных систем в локальных средах;
- следить за выходом "заплат" и своевременно обновлять с их помощью систему и приложения;
- управлять добавочными программными пакетами.

Процесс конфигурирования готового дистрибутива или программного пакета, направленный на удовлетворение всех ваших потребностей (и не допускающий нарушения локальных условий защиты, размещения файлов и топологии сети), часто называют "локализацией". В этой главе рассматриваются методики, которые позволяют упростить инсталляцию программного обеспечения, в том числе в крупных системах. Мы также рассмотрим процедуру инсталляции для каждого из наших примеров операционных систем, включая некоторые возможности автоматизации, использующие распространенные инструменты (предназначенные для конкретной платформы).

# 12.1. Инсталляция систем Linux и OpenSolaris

Процедура инсталляции всех современных дистрибутивов Linux довольно проста. Для OpenSolaris действует множество таких же соглашений, так что процесс инсталляции этой системы практически не отличается от установки Linux, особенно на персональные компьютеры.

Обычно для инсталляции необходимо загрузиться с DVD-диска, ответить на ряд вопросов, по выбору сконфигурировав разделы диска, и указать программе-инсталлятору, какие программные пакеты нужно установить. Такие системы, как Ubuntu и OpenSolaris, включают опцию "live", которая позволяет запускать операционную систему, не инсталлируя ее в действительности на локальный диск. Это очень полезная особенность упомянутых систем, но в наши дни она стала практически стандартной функцией большинства дистрибутивов.

Инсталляция базовой операционной системы с локального носителя представляет собой довольно тривиальную операцию благодаря GUI-приложениям, которые "проведут" вас через все этапы этого процесса. В табл. 12.1 перечислены указатели на подробные инструкции по инсталляции для каждого из наших примеров дистрибутивов.

**Таблица 12.1. Документация по инсталляции для систем Linux и OpenSolaris**

| Дистрибутив | Источник документации |
|---|---|
| Red Hat Enterprise Linux | `redhat.com/docs/manuals/enterprise` |
| SUSE | `en.opensuse.org/Installation` |
| Ubuntu | `help.ubuntu.com/community/Installation` |
| OpenSolaris | `dlc.sun.com/osol/docs/content/dev/getstart` |

## Загрузка по сети на персональном компьютере

Любой, кому придется инсталлировать систему сразу на несколько компьютеров, столкнется с недостатками инсталляции в интерактивном режиме. Это большие затраты времени, "предрасположенность" к ошибкам и скучная необходимость повторения стандартного процесса инсталляции на сотнях систем. Свести к минимуму "ошибки человеческого фактора" можно благодаря контрольному списку локализации, хотя даже его использование не сможет полностью защитить от всевозможных ошибок.

Для того чтобы ослабить воздействие перечисленных выше факторов, большинство систем включает возможности инсталляции по сети, которые упрощают проведение крупномасштабных установочных мероприятий. В самых распространенных методах для загрузки системы без использования физического носителя используются сетевые протоколы DHCP и TFTP, после чего из сетевого сервера извлекаются файлы инсталляции через протоколы HTTP, NFS или FTP. Сетевые варианты инсталляции подходят для узлов с количеством систем более десяти.

Предзагрузочная среда выполнения (Preboot eXecution Environment — PXE), предназначенная для загрузки компьютеров с помощью сетевой карты без использования жестких дисков, стала стандартом (разработанным фирмой Intel), который позволяет загружать системы через сетевой интерфейс. Стандарт PXE действует подобно миниатюрной операционной системе, размещенной в схеме ПЗУ на вашей сетевой плате. Он предлагает системе BIOS использовать его сетевые особенности посредством стандартизированного API. При таком взаимодействии один загрузчик может загружать по сети one-

рационную систему на любом персональном компьютере, понимающем стандарт PXE, не устанавливая при этом какие-либо специальные драйвера для каждой сетевой платы.

🕮 Более подробно о DHCP можно почитать в разделе 14.7.

Внешняя (сетевая) часть протокола PXE не содержит никаких сложностей и подобна процедурам загрузки по сети, используемым в других архитектурах. Компьютер осуществляет широковещательную рассылку специального DHCP-запроса с установленным PXE-флагом, а DHCP-сервер или прокси-сервер возвращает DHCP-пакет, содержащий значения PXE-параметров (имя загрузочного сервера плюс имя загрузочного файла). Клиент получает от сервера свой загрузочный файл по протоколу TFTP (возможно использование многоадресной версии этого протокола), после чего запускает его.

## Конфигурирование протокола PXE в Linux

Существует несколько загрузочных систем на основе PXE, однако самой безотказной в настоящее время является система PXELINUX, разработанная Питером Анвином (H. Peter Anvin). Она является частью комплекта загрузчиков SYSLINUX, разработанного им же, который может пригодиться на все случаи жизни. Домашняя страница этой системы расположена по адресу **syslinux.zytor.com**.

PXELINUX содержит загрузочный файл, который инсталлируется в каталог **tftpboot** сервера и загружается на загружаемый ПК во время работы PXE. ПК выполняет загрузочный файл и загружает его конфигурацию с сервера; конфигурация определяет, какое ядро необходимо использовать. Эта цепочка событий может быть выполнена без вашего вмешательства; как вариант, можно создать специальное меню загрузки.

PXELINUX использует для своих загрузок PXE API и поэтому не зависит от аппаратной части в течение всего процесса загрузки. Более того, она может загружать не только Linux, но и другие операционные системы, а также может загружать устаревшие образы дискет, если вы используете ядро MEMDISK, которое тоже входит в состав пакета SYSLINUX.

На стороне сервера лучше всего использовать DHCP-сервер организации ISC (Internet Systems Consortium's). См. также материал на сайтах **netboot.me** и **boot.kernel.org**.

## Дистанционная загрузка на специализированных компьютерах

PXE — это протокол компании Intel, и его применение ограничено платформами IA-32 и IA-64. На других платформах применяются иные методы загрузки по сети, которые почти всегда более элегантные, чем протокол PXE. Интересно, что сейчас, когда Linux часто инсталлируется на компьютеры с архитектурой, отличной от Intel, многие из "специализированных" UNIX-систем теперь имеют вариант, при котором по сети загружается Linux, а не "родная" операционная система.

Компьютеры SPARC и большинство систем PowerPC используют программу Open Firmware, которую легко загрузить по сети (достаточно ввести команду **boot net**).

Системы IBM и HP также оснащены средствами загрузки по сети, но конкретные процедуры в большой степени зависят от программных пакетов Network Installation Manager (менеджер сетевой инсталляции) и Ignite-UX соответственно. Мы рассмотрим эти инструменты ниже, но только в контексте инсталляции операционных систем. За подробностями сетевой инсталляции обратитесь к документации компаний IBM и HP.

# Использование Kickstart — автоматизированного инсталлятора Red Hat Enterprise Linux

Kickstart — это утилита, предназначенная для автоматической инсталляции программного обеспечения Red Hat. Она представляет собой интерфейс к стандартному инсталлятору Red Hat — программе Anaconda — и зависит как от версии дистрибутива, так и от имеющихся пакетов RPM. Утилита Kickstart достаточно гибкая и автоматически распознает аппаратное обеспечение системы.

## *Создание файла конфигурации для утилиты Kickstart*

Работа утилиты Kickstart контролируется конфигурационным файлом, который обычно называется **ks.cfg**. Формат этого файла довольно прост. Для тех, кто не любит работать с текстовыми файлами, существует графическая утилита **system-config-kickstart**, упрощающая настройку конфигурационного файла.

Файл **ks.cfg** можно также очень легко генерировать программным путем. Предположим, что на серверах и клиентах необходимо инсталлировать разные наборы пакетов; предположим также, что у вас есть два офиса и вам нужно, чтобы система была настроена в них по-разному. Тогда следует написать небольшой Perl-сценарий, который с помощью соответствующих параметров будет генерировать конфигурационные файлы для серверов и клиентов каждого офиса. В случае изменения набора пакетов достаточно будет отредактировать Perl-сценарий, а не многочисленные конфигурационные файлы. Бывают также ситуации, когда для каждого компьютера нужен свой конфигурационный файл. Тогда тем более необходимо автоматизировать создание таких файлов.

Конфигурационный файл утилиты Kickstart состоит из трех упорядоченных частей. Первая из них — это раздел команд, где задаются такие установки, как язык, раскладка клавиатуры и часовой пояс. Здесь же с помощью параметра **url** задается источник, из которого загружается дистрибутив (в показанном ниже примере это компьютер **installserver**).

```
text
lang en_US                      # Язык, используемый в процессе инсталляции...
langsupport en_US               # ... и на этапе выполнения.
keyboard us                     # Американская раскладка клавиатуры.
timezone --utc America/EST      # Аргумент --utc означает, что часы
                                # синхронизируются по Гринвичу
mouse
rootpw --iscrypted $1$NaCl$X5jRlREy9DqNTCXjHp075/
reboot                          # Перезагрузка после инсталляции.
                                # Это почти всегда необходимо.
bootloader --location=mbr       # Стандартный загрузчик инсталлируется
                                # в главную загрузочную запись.
install                         # Новая система инсталлируется,
                                # а не обновляется.
url --url http://installserver/redhat
clearpart --all -initlabel      # Удаляются все существующие разделы
part / --fstype ext3 --size 4096
part swap --size 1024
part /var --fstype ext3 -size 1 --grow
network --bootproto dhcp
auth --useshadow --enablemd5
firewall --disabled
xconfig --defaultdesktop=GNOME --startxonboot
        --resolution 1280x1024 --depth 24
```

По умолчанию утилита Kickstart работает в графическом режиме, что препятствует ее автоматическому запуску. Ключевое слово **text** в начале файла исправляет данную ситуацию.

Параметр **rootpw** задает пароль суперпользователя на новом компьютере. По умолчанию пароль указывается в текстовом виде, что создает проблему с точки зрения безопасности. Всегда пользуйтесь опцией **--iscrypted**, чтобы задать зашифрованный пароль. Для зашифрованного пароля прекрасно работают записи паролей из файла **/etc/shadow** либо вы можете попробовать утилиту **/sbin/grub-md5-crypt** в уже построенной системе.

Директивы **clearpart** и **part** задают список разделов с указанием их размеров. С помощью опции **--grow** можно отвести всю оставшуюся область жесткого диска под один из разделов. Так можно облегчить подгонку систем с разными размерами жестких дисков. Такие усовершенствованные опции разбиения на разделы, как использование менеджера логических томов (Logical Volume Manager — LVM), поддерживаются утилитой Kickstart, но не инструментом **system-config-kickstart**. Обратитесь к встроенной в систему Red Hat оперативно-доступной документации за полным списком параметров форматирования дисков.

Во втором разделе файла приведен список инсталлируемых пакетов, начинающийся с директивы **%packages**. В списке могут быть указаны отдельные пакеты, коллекции (например, **@ GNOME**) либо запись **@ Everything**, обозначающая весь имеющийся набор пакетов. В первом случае задается лишь имя пакета, без номера версии и расширения **.rpm**. Приведем пример.

```
%packages
@ Networked Workstation
@ X Window System
@ GNOME
mylocalpackage
```

В третьем разделе конфигурационного файла указывается произвольный набор команд интерпретатора, которые подлежат выполнению утилитой Kickstart. Существует два возможных набора команд. Один из них начинается с директивы **%pre** и содержит команды, выполняемые перед началом инсталляции. Команды, выполняемые по завершении инсталляции, помечаются директивой **%post**. В обоих случаях возможности системы по распознаванию имен компьютеров ограничены, так что лучше пользоваться IP-адресами. Кроме того, команды второго набора выполняются в среде с измененным корневым каталогом, поэтому они не имеют доступа к инсталляционному носителю.

### *Создание сервера Kickstart*

Утилита Kickstart ожидает, что ее инсталляционные файлы расположены так же, как и на инсталляционном компакт-диске. На сервере ее пакеты должны находиться в каталоге **RedHat/RPMS**. В этот каталог разрешается добавлять собственные пакеты. Но есть ряд нюансов, о которых следует помнить.

Главная особенность заключается в том, что, получив команду инсталлировать все пакеты (запись **@ Everything** в разделе пакетов файла **ks.cfg**), утилита сначала устанавливает базовые пакеты, а затем — пользовательские, причем в алфавитном порядке. Если какой-то пользовательский пакет зависит от других пакетов, не входящих в базовый набор, назовите его наподобие **zzmypackage.rpm**, чтобы он инсталлировался последним.

Если не требуется инсталлировать все пакеты, укажите нужные в разделе **%packages** файла **ks.cfg** либо добавьте их в одну или несколько коллекций. Коллекции обознача-

ются записями вида @ GNOME и представляют собой предопределенные наборы пакетов, компоненты которых перечислены в файле **RedHat/base/comps** на сервере. К сожалению, формат этого файла плохо документирован. Коллекции задаются в строках, которые начинаются с цифры 0 или 1; единица указывает на то, что коллекция выбрана по умолчанию. Менять стандартные коллекции нежелательно. Мы рекомендуем оставить их в том виде, в котором они определены в Red Hat, и указывать дополнительные пакеты непосредственно в файле **ks.cfg**.

### *Задание пользовательского конфигурационного файла*

После создания конфигурационного файла необходимо заставить утилиту Kickstart использовать его. Это можно сделать несколькими способами. Официальный способ — загрузиться с DVD-диска и запросить инсталляцию Kickstart, введя **linux ks** в строке приглашения **prompt:**. Если не указать дополнительных аргументов, система определит свой сетевой адрес по протоколу DHCP. Затем она узнает имя загрузочного DHCP-сервера и загрузочного файла, попытается смонтировать серверный каталог через NFS и установит загрузочный файл в качестве конфигурационного. При отсутствии сведений о загрузочном файле система будет искать файл **/kickstart/***IP_адрес_узла***-kickstart**.

Другой способ задания конфигурационного файла заключается в указании пути к нему в виде аргумента опции **ks**. Здесь есть несколько возможных вариантов. Команда

```
boot: linux ks=http:сервер:/path
```

заставляет утилиту Kickstart загрузить конфигурационный файл по протоколу HTTP, а не через NFS. Аргумент **ks=floppy** заставляет утилиту искать файл **ks.cfg** на локальном гибком диске.

Для того чтобы вообще не использовать загрузочный носитель, вам нужно научиться работать с протоколом PXE. Подробно о нем речь шла в начале этой главы.

## Использование AutoYaST: автоматизированный инсталлятор SUSE

YaST2 — это универсальная утилита инсталляции и конфигурирования SUSE. Она имеет привлекательный графический интерфейс и очень удобна в использовании при инсталляции одной системы. Ее автоматизированный эквивалент, AutoYaST, можно назвать самой мощной инсталляционной программой из всех дистрибутивов, описанных в этой книге. Подробную документацию можно загрузить с адреса **suse.com/~ug/autoyast_doc**.

SUSE разбивает процесс автоматической инсталляции на три фазы: подготовка, инсталляция, конфигурирование. Первоначальная подготовка выполняется с помощью модуля AutoYaST (YaST2).

```
suse$ /sbin/yast2 autoyast
```

Этот модуль помогает указать детали требуемой установки. Результатом выполнения этого модуля является управляющий XML-файл, который сообщает инсталлятору о способе конфигурирования системы SUSE; структура файла описана в интерактивном документе, упомянутом выше.

Процесс конфигурирования можно ускорить. Модуль AutoYaST может читать конфигурационные файлы Kickstart, помогая модернизировать "устаревшие" системы. Если вам понадобится продублировать конфигурацию на компьютере, на котором вы работаете в настоящий момент, то на этот случай тоже есть вариант автоматизации.

Для того чтобы произвести инсталляцию, вам понадобится следующее:

- сервер DHCP с той же маской подсети, что и компьютер, на котором вы планируете выполнить инсталляцию;
- инсталляционный сервер SUSE или хранилище пакетов;
- сервер, на котором будет размещена информация о конфигурации, необходимая для инсталляции.

Последний из этих серверов может предоставить конфигурационные файлы по одному из протоколов, предлагаемых вам на выбор, — HTTP, NFS или TFTP.

При обычной установке нужно создать управляющий файл для каждого компьютера, на котором вы планируете произвести инсталляцию. AutoYaST использует IP-адрес клиента, чтобы определить, какой управляющий файл следует использовать. Этот подход не будет эффективным, если нужно произвести инсталляцию на ряде компьютеров, немного отличающихся друг от друга.

С помощью системы правил можно создавать более сложные инсталляции. На основании таких характеристик системы, как размер диска, идентификатор компьютера или доступность PCMCIA, разные управляющие файлы подгоняются под искомую систему. Содержимое всех выбранных управляющих файлов объединяется, причем в случае возникновения конфликтов содержимое последнего управляющего файла перекрывает содержимое предыдущего файла. (Управляющий файл не должен определять все аспекты конфигурации системы, поэтому такое объединение имеет смысл.)

Управляющие файлы могут также определять "классы" компьютеров на основе их имен или диапазонов IP-адресов. Каждый класс может иметь связанный с ним вспомогательный управляющий файл. Компьютеры могут не относиться к классам, могут относиться к одному классу или к множеству классов, а их конфигурации будут включать содержимое всех управляющих файлов соответствующего класса.

Благодаря объединению содержимого множества управляющих файлов, структура AutoYaST позволяет определять сложные установки с минимальной избыточностью. Человеку трудно воспринимать то, что написано в управляющем XML-файле, зато процесс может с легкостью не только читать эти файлы, но и редактировать их с помощью обычных доступных средств обработки XML-данных.

## Автоматизированная инсталляция систем Ubuntu

Инсталлятор Debian (он называется *debian-installer*) рекомендуется использовать в качестве варианта "предварительной" автоматизированной инсталляции системы Ubuntu. Как и в случае Kickstart, файл предварительной конфигурации отвечает на вопросы, задаваемые инсталлятором. При выполнении "предварительной" инсталляции нельзя использовать существующие разделы диска: в этом случае либо задействуется существующее свободное пространство, либо выполняется перераспределение всего диска.

Все интерактивные части инсталлятора Debian используют утилиту **debconf**, чтобы решить, какие вопросы нужно ставить и какие ответы использовать по умолчанию. Предоставив утилите **debconf** базу данных с заранее сформулированными вопросами, вы можете полностью автоматизировать работу инсталлятора. Вы можете или сгенерировать базу данных вручную (она представляет собой обычный текстовый файл), или выполнить интерактивную инсталляцию на тестовой системе, а затем передать свои ответы утилите **debconf** с помощью следующих команд.

```
ubuntu$ sudo debconf-get-selections --installer > preseed.cfg
ubuntu$ sudo debconf-get-selections >> preseed.cfg
```

Создайте конфигурационный файл, который будет доступен по сети, и передайте его в ядро во время инсталляции с помощью следующего аргумента ядра.

```
preseed/url=http://host/path/to/preseed
```

Синаксис "предынсталляционного" файла, обычно именуемого **preseed.cfg**, довольно прост и во многом сходен с Red Hat-файлом **ks.cfg**. Следующий пример представлен в сокращенном виде.

```
d-i debian-installer/locale string en_US
d-i console-setup/ask_detect boolean false
d-i console-setup/layoutcode string us
d-i netcfg/choose_interface select auto
d-i netcfg/get_hostname string unassigned-hostname
d-i netcfg/get_domain string unassigned-domain
d-i netcfg/wireless_wep string
...
d-i partman-auto/disk string /dev/sda
d-i partman-auto/method string lvm
d-i partman-auto/choose_recipe select atomic
...
d-i passwd/user-fullname string Daffy Duck
d-i passwd/username string dduck
d-i passwd/user-password-crypted password $1$/mkq9/$G//i6tN.x6670.95lVSM/
d-i user-setup/encrypt-home boolean false
tasksel tasksel/first multiselect ubuntu-desktop
d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean true
d-i finish-install/reboot_in_progress note
xserver-xorg xserver-xorg/autodetect_monitor boolean true
...
```

Ряд опций в этом листинге просто запрещают диалоги, которые обычно требуют взаимодействия с пользователем. Например, опция **ask_detect** запрещает выбор раскладки клавиатуры. Аналогично опция **wireless_wep** предотвращает вопросы о ключах WEP (Wireless Encryption Protocol — протокол шифрования в беспроводной связи).

Такая конфигурация старается идентифицировать сетевой интерфейс, который в действительности подключен к сети (**choose_interface select auto**) и получает сетевую информацию через протокол динамического конфигурирования узла DHCP. Предполагается, что значения системного имени главного компьютера и домена предоставляются протоколом DHCP и не переопределяются.

Наличие **partman**-строк свидетельствует о том, что для сегментирования дисковой памяти используется пакет **partman-auto**. Если система имеет несколько дисков, то для инсталляции вы должны указать нужный диск. В противном случае (т.е. для единственного диска) используется значение **/dev/sda**.

Предлагается несколько "рецептов" сегментирования дисковой памяти:

- вариант **atomic** помещает все системные файлы в один раздел;
- вариант **home** создает отдельный раздел для каталога **/home**;
- вариант **multi** создает отдельные разделы для каталогов **/home**, **/usr**, **/var** и **/tmp**.

Вы можете создавать пользователей посредством ряда директив **passwd**. Как в случае Kickstart-конфигурации, мы настоятельно рекомендуем использование хешированн-

ных MD5-значений паролей. Файлы предварительной инсталляции часто хранятся на HTTP-серверах и могут быть обнаружены любопытными пользователями. (Безусловно, MD5-пароль постоянно является объектом для грубых силовых атак.) Опция выбора задачи (`tasksel`) позволяет указать тип подлежащей инсталляции Ubuntu-системы из предложенных вариантов: `standard`, `ubuntu-desktop`, `dns-server`, `lamp-server`, `kubuntu-desktop`, `edubuntu-desktop` и `xubuntu-desktop`.

Представленный выше пример файла предварительной инсталляции взят из документации по инсталляции системы Ubuntu, доступной по адресу: `help.ubuntu.com`. Руководство содержит полную документацию по синтаксису и применению файла предварительной инсталляции.

Несмотря на то что происхождение системы Ubuntu не связано с "родословной" Red Hat, на ее собственный базовый инсталлятор "привита" совместимость с управляющими Kickstart-файлами. Кроме того, Ubuntu включает утилиту `system-config-kickstart` для создания этих файлов. В инсталляторе Kickstart для системы Ubuntu опускается ряд важных функций, которые поддерживаются Red Hat-инсталлятором Anaconda (например, LVM и конфигурация брандмауэра). Если у вас нет веской причины для выбора Kickstart, мы все же рекомендуем использовать инсталлятор Debian.

## 12.2. Инсталляция Solaris

Подобно многим поставщикам аппаратных средств, компания Sun вводит новые серверы с предварительно инсталлированной системой Solaris. Администратору нужно лишь ответить на несколько коротких вопросов и перезагрузить сервер, и операционная система будет готова для локализации. С годами мы оценили эту функцию предварительной инсталляции, поскольку инсталлятор Solaris мы считали отвратительным. Но команда OpenSolaris, можно сказать, "одумалась", и ее новый инсталлятор (первоначально названный "Caiman") заслуживает уважения.

В настоящее время для инсталляции Solaris предлагается "живой" компакт-диск, который предоставляет возможность "опробовать перед покупкой", подобную реализованной в системе Ubuntu. Процесс инсталляции системы чрезвычайно прост и перед установкой на локальный диск предусматривает ответы всего на несколько вопросов.

Как и в мире Linux, администраторам Solaris нужен способ реализовать множество инсталляций систем по сети. Подобно своим Linux-ориентированным "собратьям", системы Solaris, построенные на базе процессоров Intel, для помощи в сетевой загрузке могут использовать серверы PXE. Системы с процессорами SPARC используют ППЗУ OpenBoot PROM, также известное под именем OBP. К OBP обычно можно получить доступ с помощью комбинации клавиш <STOP+A> (на клавиатуре Sun). Посредством соответствующих команд OBP позволяет идентифицировать оборудование и выполнить его диагностику, выявить сбойные ситуации, а также передать процесс загрузки более интеллектуальному начальному загрузчику, подобно тому как действует BIOS в системах Intel. При этом OBP отличается большим количеством функций (по сравнению с большинством PC BIOS), включающих встроенную поддержку загрузки системы по сети.

Средство сетевой загрузки через протокол DHCP или RARP получает IP-адрес, а затем загружает ядро через протокол TFTP. Загруженное для автоматизированной инсталляции, ядро подключается к серверу HTTP или монтирует общую сетевую файловую систему (NFS) для загрузки соответствующего образа системы и запускает процесс инсталляции.

Solaris предлагает два метода автоматической сетевой инсталляции:

- JumpStart — традиционная служба инсталлятора, разработанная компанией Sun;
- Automated Installer — служба замены, используемая системой OpenSolaris.

JumpStart — старое средство инсталляции, которое сначала появилось в Solaris 2.6 и может быть использовано во всех версиях, вплоть до Solaris 10. Подобно большинству методов автоматической инсталляции, служба JumpStart использует файлы предопределенных ответов и выбор клиента на основе правил, что автоматически обеспечивает выбор инсталляции.

Самый большой недостаток JumpStart — слабая масштабируемость. Каждого клиента нужно добавлять в сервер вручную для инсталляции посредством MAC-адреса. В файлах конфигурации задаются типы инсталляции, значения конфигурации и другие параметры для каждого клиента в отдельности. Это усиливает возможности администраторов (в частности, увеличивает гибкость в управлении), но такие файлы конфигурации становятся очень громоздкими в среде с сотнями или тысячами систем.

Служба Automated Installer (AI) — это новинка. Основные цели ее разработки — улучшить масштабируемость и возможности конфигурации. AI уходит корнями в JumpStart, но дистанцируется частично благодаря использованию новой терминологии. Пока писались эти строки, служба AI оставалась все еще в процессе разработки, но ее можно считать практически готовой для использования в коммерческих целях. Больше всего ограничений AI имеет для последних версий системы OpenSolaris и в настоящее время не работает на всех традиционных версиях Solaris.

## Сетевые инсталляции с помощью JumpStart

Исходной целью службы JumpStart было просто разрешить инсталляцию Solaris по сети, но у нее есть все возможности и для автоматической инсталляции. Через некоторое время в компании Sun поняли, что автоматизированные инсталляции требуют более тонкого управления, поэтому и были добавлены расширенные средства, которые сейчас продублированы в Custom JumpStart. Автоматизированная сетевая инсталляция Custom JumpStart включает несколько компонентов.

- Сервер инсталляции, который принимает инсталляционный носитель. Один сервер инсталляции может обслуживать носители для нескольких типов инсталляции, например различных версий системы Solaris, или поддерживать несколько платформ.
- Сервер начальной загрузки, который помогает клиентам выполнять начальную загрузку и направляет их на серверы инсталляции. Сервер начальной загрузки необходим лишь в случае, если система клиента и сервер инсталляции находятся в различных подсетях.
- Ряд файлов, которые идентифицируют клиентов, отвечают на вопросы по конфигурации и выбирают пакеты.
- NFS- или HTTP-сервер, который использует пакеты, файлы инсталляции и информацию о конфигурации.

Компоненты на стороне сервера могут размещаться на одном и том же компьютере. Серверы не зависят от версии инсталлируемой системы и используемой платформы. Например, сервер загрузки и инсталляции для SPARC-ориентированной системы Solaris 9 может предложить услуги по инсталляции для клиентов x86 Solaris 10.

Поскольку параметры сетевой загрузки могут быть включены в DHCP-ответы, вы можете использовать DHCP-сервер в качестве альтернативы для специализированного

сервера начальной загрузки JumpStart. Возможно, протокол DHCP является более приемлемым вариантом для систем x86, которые используют PXE-загрузку, и для клиентских систем, расположенных в разных подсетях по отношению к серверу инсталляции. Здесь мы рассмотрим только случай расположения клиента и сервера в одной подсети (за подробностями обращайтесь по адресу: **docs.sun.com/doc/817-5504**).

Установка сервера инсталляции довольно проста. Инструменты установки можно найти на компакт- или DVD-диске Solaris. Вставьте Solaris-носитель в дисковод сервера инсталляции и выполните следующие команды, чтобы сконфигурировать простой сервер инсталляции.

```
solaris$ sudo mkdir -p /jumpstart/s10sparc
solaris$ cd /cdrom/cdrom0/s0/Solaris_10/Tools
solaris$ sudo ./setup_install_server /jumpstart/s10sparc
```

Здесь мы переносим SPARC-файлы инсталляции в каталог **/jumpstart/s10sparc** сервера инсталляции. Сценарий **setup_install_server** копирует эти файлы и добавляет соответствующие "ловушки" для сетевых инсталляций. Если в вашем распоряжении есть только компакт-диски, то для копирования их содержимого на сервер используйте команду **add_to_install_server**.

Задачи автоматизированной инсталляции конфигурируют несколько файлов:

- файл **rules** идентифицирует клиентов и назначает профили инсталляции;
- отдельные файлы профилей задают схему разделов диска, инсталлируемые пакеты и прочие системные данные;
- файл **sysidcfg** предоставляет предварительно сконфигурированные ответы на вопросы инсталляции;
- сценарии оболочки могут выполняться до и после процесса инсталляции (по необходимости).

Когда клиент запрашивает сетевую инсталляцию, JumpStart использует файл **rules**, чтобы идентифицировать клиента в соответствии с такими атрибутами, как имя узла клиента, подсеть или модель. Если атрибуты совпадают, JumpStart считывает содержимое соответствующего профиля, отвечает на вопросы инсталляции с помощью файла **sysidcfg** и выполняет пользовательские сценарии до и после инсталляции.

Первый шаг в создании JumpStart-конфигурации состоит в создании каталога для хранения всех файлов конфигурации.

```
solaris$ sudo mkdir -m 755 /jumpstart/config
```

Этот каталог должен быть открыт для совместного пользования через протокол NFS или HTTP, чтобы клиенты могли получить к нему доступ. Например, для протокола NFS добавьте строку

```
share -F nfs /jumpstart
```

в файл **/etc/dfs/dfstab** и выполните команду **shareall**, чтобы инициализировать службу NFS.

Синаксис файла **rules** простой, но мощный. Системы могут быть идентифицированы по сети, имени узла, модели, имени домена или по многим другим атрибутам[1]. В следующем файле **rules** указывается один профиль для систем в сети **192.168.10.0**, а другой — для систем SPARC, которые оснащены 2–4 Гбайт памяти.

[1] Для того чтобы получить доступ к Sun-руководству, которое содержит подробные сведения о файле `rules` и файлах профилей, используйте Google-ресурс, задав в строке поиска текст "Custom JumpStart and Advanced Installations".

```
network 192.168.10.0 - profile_a -
arch sparc && memsize 2048-4096 begin profile_b end
```

В этом примере сети нет пользовательских сценариев, но используется профиль инсталляции **profile_a**. В другом примере применяются сценарии **begin** и **end**, а также профильный файл **profile_b**.

Файлы профилей также просты. Файловые системы и типы инсталляции задаются с помощью ключевых слов (их очень много). Вот как может выглядеть пример профиля.

```
install_type initial_install
system_type standalone
partitioning default
filesys any 512 swap          # Задаем размер /swap
cluster SUNWCpall
```

Процесс инсталляции типа **initial_install** начинается с чистого списка, в противоположность выполнению модернизации. Этот профиль использует стандартную схему сегментирования дисковой памяти. Строка **cluster SUNWCpall** идентифицирует "группу инсталляции" пакетов; в данном случае это доступные Solaris-пакеты.

Файл **sysidcfg**, который предварительно конфигурирует другие аспекты инсталляции, состоит из строк следующего формата.

*Ключевое_слово=значение*

Ключевые слова не различаются по прописным и строчным буквам и, за исключением ключевого слова **network_interface**, могут употребляться лишь один раз. Если ключевое слово используется несколько раз, учитывается только его первый экземпляр.

Некоторые ключевые слова зависят от других и могут заключаться в фигурные скобки. Такие зависимые ключевые слова не могут быть использованы, если не задано соответствующее родительское (независимое) ключевое слово. Независимые ключевые слова перечислены в табл. 12.2. О зависимых ключевых словах можно узнать, обратившись к man-странице, посвященной файлу **sysidcfg**.

**Таблица 12.2. Независимые ключевые слова для JumpStart-файла sysidcfg**

| Ключевое слово | Что оно определяет |
|---|---|
| **keyboard** | Раскладка клавиатуры и язык |
| **name_service** | Конфигурация службы имен для NIS, DNS или LDAP |
| **network_interface** | Информация о подключении к сети: имя узла, IP-адрес и т.д. |
| **nfs4_domain** | Домен, предназначенный для NFS версии 4 |
| **root_password** | Зашифрованный пароль пользователя **root** |
| **security_policy** | Сетевой протокол аутентификации Kerberos |
| **service_profile** | Доступные сетевые службы |
| **system_locale** | Язык системы |
| **terminal** | Тип терминала |
| **timeserver** | Сетевой сервер даты и времени |
| **timezone** | Часовой пояс системы |

В качестве альтернативы для файла **sysidcfg** можно использовать ограниченный набор параметров предварительной конфигурации, которые задаются через протокол DHCP или с помощью такой сетевой службы, как DNS. Однако мы рекомендуем использовать файл **sysidcfg**, поскольку альтернативный вариант позволяет задать ограниченный набор параметров.

В следующем примере файла **sysidcfg** конфигурируется система **sake**, которая имеет один сетевой интерфейс.

```
keyboard=US-English
system_locale=en_US
timezone=US/Mountain
terminal=sun-cmd
timeserver=time.nist.gov
name_service=DNS {domain_name=solaris.booklab.atrust.com
  name_server=192.168.2.10
  search=atrust.com,booklab.atrust.com}
nfs4_domain=dynamic
root_password=m4QPOWNY
network_interface=e1000g0 {hostname=sake
  default_route=192.168.10.254
  ip_address=192.168.10.15
  netmask=255.255.255.0}
```

Если вы передаете один и тот же файл **sysidcfg** нескольким клиентам, IP-адрес, безусловно, позволит различать системы. Для того чтобы принудительно сконфигурировать сетевой интерфейс при первой загрузке системы, вы можете не включать эти данные в файл. Или же, чтобы получить сетевой адрес от протокола динамической конфигурации узла DHCP (вместо статического его назначения), используйте следующую строку.

```
network_interface=e1000g0 {dhcp}
```

После формирования файлов **rules**, **sysidcfg** и ваших профилей скопируйте их в каталог **/jumpstart/config** и выполните Sun-утилиту **check**, которая проверит достоверность конфигурации. Использование сценария **check**, который должен быть запущен из каталога **config**, обязательно: он создает файл **rules.ok**, удостоверяющий для JumpStart, что представленные файлы синтаксически приемлемы. Не опускайте этот этап — в противном случае JumpStart не будет работать.

```
solaris$ sudo cp /jumpstart/s10sparc/Solaris_10/Misc/jumpstart_sample/check
      /jumpstart/config
solaris $  sudo ./check
Validating rules...
Validating profile profile_A...
The custom JumpStart configuration is ok.
```

В конце процесса конфигурации ваш каталог **/jumpstart/config** должен выглядеть примерно так.

```
solaris$ ls -l
-rwxr-xr-x 1       root root   52152 Aug 23 19:42 check
-rw-r--r-- 1       root root     413 Aug 23 19:29 profile_a
-rw-r--r-- 1       root root      48 Aug 23 19:13 rules
-rw-r--r-- 1       root root      62 Aug 23 19:43 rules.ok
-rw-r--r-- 1       root root     314 Aug 23 17:35 sysidcfg
```

Каждого устанавливаемого клиента необходимо добавить на сервер инсталляции через JumpStart; это двухступенчатый процесс. Во-первых, добавьте MAC-адрес клиента в файл сервера **/etc/ethers**. Во-вторых, выполните утилиту **add_install_client**, чтобы добавить клиента в базу данных конфигурации, как показано в следующих строках.

```
solaris$  cd /jumpstart/s10sparc/Solaris_10/Tools
solaris$  sudo ./add_install_client -c server:/jumpstart sake sun4v
```

В данном случае клиент с именем **sake** будет использовать для инсталляции JumpStart NFS на хост-сервере. Реальный процесс сетевой инсталляции для клиента вы начнете после OBP-приглашения.

```
ok boot net - install
```

В этом непростом процессе предусмотрена возможность проведения пользовательских и гибких инсталляций при участии определенных интеллектуальных ресурсов.

## Сетевые инсталляции с помощью автоматизированного инсталлятора

Разработчики системы OpenSolaris оценили сложность службы JumpStart и решили создать новое средство установки OpenSolaris. Этот инструмент, Automated Installer (AI), воспроизводит стиль JumpStart, но позволяет справиться с определенными сложностями благодаря удобной утилите **installadm**. В самой простой форме инсталляцию сервера можно выполнить с помощью одной команды. Все файлы, которые вам нужно запустить посредством утилиты AI, собраны в пакете **SUNWinstalladm-tools**.

Любой AI-сервер предлагает одну или несколько "служб инсталляции", каждая из которых представляет вариант инсталляции операционной системы, определяемый клиентами во время начальной загрузки посредством многоадресной службы доменных имен DNS. Различные службы могут удовлетворять разным инсталляционным потребностям (например, одна служба предназначена для веб-серверов, специально адаптированных к конкретной среде, а другая — для серверов баз данных).

Установив местоположение инсталлятора, клиент отыскивает конфигурацию, или манифест, который отвечает его системному описанию. Клиент выполняет инсталляцию с помощью данных из файлов манифеста. Никакой конфигурации клиента не требуется, хотя при необходимости возможны пользовательские инсталляции клиентов.

AI-инсталляция сервера связывает все необходимые части воедино в удобный пакет, включающий службы DHCP и TFTP. Перед добавлением их в сеть обязательно согласуйте свои действия с вашим администратором сети.

Неявно утилита AI создает три XML-отформатированных файла манифеста.

- AI-файл манифеста содержит данные о сегментировании дисковой памяти и организации пакетов (подобен профильному файлу JumpStart).
- SC-файл манифеста содержит информацию о конфигурации системы (часовой пояс и учетные записи) — во многом так же, как JumpStart-файл **sysidcfg**.
- Файл критериев сопоставляет два других файла манифеста с устройствами клиентов, подобно тому как это делает файл **rules** в службе JumpStart.

Если вы считаете, что использование языка XML напрягает вас интеллектуально, можете вручную отредактировать манифесты и создать пользовательские конфигурации. Обычно достаточно лишь запустить утилиту **installadm** для добавления, удаления, разрешения, запрещения и перечисления новых инсталляционных служб, а также для создания пользовательских конфигураций клиентов.

Например, следующая команда **installadm** создает новую инсталляционную службу, которую вы можете использовать для инсталляции клиента. В данном примере в качестве источника инсталляции используется ISO-образ системы OpenSolaris выпуска 0906. Опция **-c 10** вынуждает сервер DHCP обеспечить до 10 динамических адресов, начиная с 192.168.1.200. Образ инсталляции копируется в каталог **/export/install**.

```
solaris$ sudo installadm create-service -s ~/osol-0906-x86.iso
   -i 192.168.1.200 -c 10 /export/install
Setting up the target image at /export/install ...
Warning: Using default manifest </usr/share/auto_install/default.xml>
Registering the service _install_service_46501._OSInstall._tcp.local
Creating DHCP Server
Created DHCP configuration file.
Created dhcptab.
Added "Locale" macro to dhcptab.
Added server macro to dhcptab - opensolaris.
DHCP server started.
dhtadm: Unable to signal the daemon to reload the dhcptab
Added network macro to dhcptab - 192.168.1.0.
Created network table.
adding tftp to  /etc/inetd.conf
Converting /etc/inetd.conf
copying boot file to /tftpboot/pxegrub.I86PC.OpenSolaris-1
Service discovery fallback mechanism set up
```

Для того чтобы инсталлировать клиента, выполните сетевую загрузку обычным способом. Сервер использует предопределенные правила для отыскания образа, загрузки его на сторону клиента и выполнения инсталляции.

Автоматизированный инсталлятор (Automated Installer) изменяется довольно быстро. После инсталляции пакета обратитесь за информацией о текущем его состоянии по адресу: `/usr/share/doc/auto_install/index.html`.

# 12.3. Инсталляция HP-UX

Как сервер-ориентированная операционная система, предназначенная, в основном, исключительно для крупных приложений, которые требуют большого объема ответственной работы, система HP-UX не заботится о предоставлении перспективного процесса инсталляции следующего поколения. Ее программное обеспечение инсталляции с текстовым интерфейсом можно назвать практическим руководством, которое просто проведет вас через основные опции конфигурации: сегментирование дисковой памяти, установка сетевых параметров, инсталляция программ.

Для узлов системы, которые требуют сетевых и автоматизированных инсталляций, доступна HP-опция Ignite-UX. Инсталлятор Ignite-UX может установить по сети несколько систем HP-UX одновременно. Клиенты PA-RISC выполняют загрузку с помощью протокола начальной самозагрузки BOOTP (Bootstrap Protocol), а системы Itanium используют протокол DHCP. Вы можете сконфигурировать несколько программных репозиториев. Например, пакеты инсталляции могут быть предоставлены из одного хранилища, "заплаты" — из другого, а пакеты приложений — из третьего. В качестве бонуса Ignite-UX также включает службу, которая восстанавливает конфигурацию компьютера из недавнего образа.

Для установки Ignite-UX необходимо выполнить следующие действия.

- Инсталлируйте программное обеспечение Ignite-UX и пакеты HP-UX на сервере.
- Сконфигурируйте Ignite-UX с предложением соответствующих вариантов инсталляции.
- Разрешите использовать такие зависимости служб Ignite-UX, как NFS и BOOTP.
- Добавьте в сервер MAC-клиент и IP-адреса.

После конфигурации сервера вы можете добавить в системы клиентов опцию начальной загрузки (через HP-менеджер загрузки EFI Boot Manager), чтобы заставить их инсталлировать систему HP-UX из сервера Ignite-UX. В качестве альтернативного варианта для систем, которые уже работают под управлением HP-UX, вы можете использовать команду **bootsys**, чтобы "перенести" инсталляцию с сервера на сторону клиента.

В наших примерах систем Ignite-UX устанавливается заранее, но если в вашей системе его нет, попробуйте выполнить команду **swinstall -s /dvdrom Ignite-UX**. Здесь **/dvdrom** означает точку монтирования для DVD-диска, который содержит носитель операционной системы, или "операционную среду" в терминологии HP. Результаты инсталляции выражаются в виде ряда инсталлированных пакетов, и некоторые из них перечислены ниже.

```
hp-ux$ sudo swlist Ignite-UX
...
# Ignite-UX                C.7.5.142    HP-UX System Installation Services
Ignite-UX.BOOT-COMMON-IA   C.7.5.142    Boot Components for IPF clients
Ignite-UX.BOOT-COMMON-PA   C.7.5.142    Boot Components for PA-RISC clients
Ignite-UX.BOOT-KRN-11-11   C.7.5.142    Boot Kernel for B.11.11 clients
Ignite-UX.BOOT-KRN-11-23   C.7.5.142    Boot Kernel for B.11.23 clients
Ignite-UX.BOOT-KRN-11-31   C.7.5.142    Boot Kernel for B.11.31 clients
Ignite-UX.BOOT-SERVICES    C.7.5.142    Boot Services for Installations
...
```

Файлы конфигурации и двоичные файлы Ignite-UX разбросаны случайным образом по всей файловой системе. Самые важные компоненты перечислены в табл. 12.3.

**Таблица 12.3. Важные двоичные файлы, каталоги и файлы конфигурации, используемые Ignite-UX**

| Имя файла | Назначение |
|---|---|
| **/etc/bootptab** | Действует как незашифрованная "база данных" для **bootpd** |
| **/etc/opt/ignite/instl_boottab** | Регистрирует IP-адреса для загрузки клиентов PA-RISC |
| **/opt/ignite/bin/bootsys** | Обновляет клиентов, которые уже работают под управлением HP-UX |
| **/opt/ignite/bin/make_config** | Создает файл с данными о конфигурации с помощью инсталляционного хранилища |
| **/opt/ignite/bin/make_depots** | Создает инсталляционные хранилища на основе некоторых исходных носителей source media |
| **/opt/ignite/bin/manage_index** | Добавляет хранилище в индекс Ignite-UX |
| **/opt/ignite/lbin/setup_server** | Совместно использует каталог **/var/opt/ignite/clients** через протокол NFS |
| **/var/opt/ignite/clients** | Сохраняет файлы конфигурации клиентов (dir) |
| **/var/opt/ignite/data** | Традиционно использовался для хранилищ инсталляции (dir) |
| **/var/opt/ignite/INDEX** | Индексирует все хранилища инсталляции, известные Ignite-UX |

Команда **make_depots** извлекает пакеты инсталляции и устанавливает их в качестве хранилища операционной среды для инсталляции на стороне клиентов. После создания хранилища выполните команду **make_config**, чтобы прочитать содержимое хранилища и создать файл конфигурации, который его описывает. Конфигурация становится известной для Ignite-UX посредством выполнения команды **manage_index**, которая добавляет конфигурации в файл **INDEX**. Ниже приведен пример последовательности команд для версии 11i v3.

```
hp-ux$ cd /opt/ignite/bin
hp-ux$ sudo ./make_depots -s /dev/dsk/c2t2d0
  -d /var/opt/ignite/depots/Rel_B.11.31/oe_media
hp-ux$ sudo ./make_config -s /var/opt/ignite/depots/Rel_B.11.31/core_media
  -c /var/opt/ignite/data/Rel_B.11.31/oe_media_cfg
hp-ux$ sudo ./manage_index -n "HP-UX B.11.31 Default" -c "11i v3"
hp-ux$ sudo ./manage_index -a
  -f /var/opt/ignite/data/Rel_B.11.31/oe_media_cfg -c "11i v3"
```

Прежде чем сервер смогут использовать клиенты, вы должны разрешить протокол BOOTP и совместно используемый каталог **clients**. Отдельные клиенты должны добавляться в файл **instl_boottab** или **bootptab**, в зависимости от типа компьютеров, PA-RISC или Itanium. Для совместного использования каталога **config** посредством файловой системы NFS достаточно выполнить сценарий **/opt/ignite/lbin/setup_server**. Эта команда неявно создает совместно используемый NFS-ресурс в каталоге **/etc/dfs/sharetab**.

Вы можете включить протокол BOOTP, "раскомментировав" строку **bootps** в файле **/etc/inetd.conf**. А затем предложите сценарию **inetd** перечитать конфигурацию, запустив его на выполнение с ключом **-c**: **/usr/sbin/inetd -c**.

Прежде чем служба Ignite-UX будет предлагать клиенту службы инсталляции, обычно распознают клиента по MAC-адресу. Но чтобы облегчить вашу ношу как администратора, мы рекомендуем использовать HP-понятие "анонимных клиентов", которые не связаны ни с какими конкретными MAC-адресами.

Системы PA-RISC и Itanium опираются на различные механизмы начальной загрузки, и эти две службы конфигурируются несколько по-разному. Для того чтобы сконфигурировать службы начальной загрузки Ignite-UX для систем PA-RISC, отредактируйте файл **/etc/opt/ignite/instl_boottab**. Строки в этом файле имеют следующий формат.

*IP_адрес*:*MAC_адрес*:*дата_и_время_последнего_использования* **[ :reserve ]**

Поле *IP_адрес* назначает IP-адрес для нового клиента для использования во время доступа к службам инсталляции. Необязательное поле *MAC_адрес* идентифицирует компьютер клиента; если вы его опустите, указанный IP-адрес может использовать любой клиент (однако обратите внимание на взаимосвязь с ключевым словом **reserve**). Третье поле используется и поддерживается службой Ignite-UX; поэтому оставьте его пустым в случае добавления новых элементов.

Если в последнем столбце представлено ключевое слово **reserve**, IP-адрес резервируется для использования клиентом, чей MAC-адрес указан во втором поле. Если слово **reserve** не задано, второе поле просто отображает последний MAC-адрес, который использовал заданный IP-адрес.

Файлы **/etc/bootptab** и **/etc/dhcptab**, используемые для начальной загрузки систем Itanium, имеют совсем другой формат. Эти файлы хорошо прокомментированы и проиллюстрированы многочисленными примерами, которые мы не считаем нужным здесь повторять. (Обратите внимание на то, что в системах HP-UX единственный демон, **bootpd**, обслуживает запросы, поступаемые как от BOOTP, так и от DHCP.) Метод загрузки через протокол DHCP предпочтительнее для систем Itanium, поскольку он может предоставлять услуги анонимным клиентам.

Просмотрите комментарии в файлах **/etc/bootptab** и **/etc/dhcpv6tab**. Если вы уже сконфигурировали сервер Ignite-UX так, как описано выше, клиенты смогут выполнять загрузку с него по сети. На стороне клиента прервите обычный процесс загрузки, войдите в менеджер загрузки EFI Boot Manager и добавьте сетевое устройство. Клиент запросит IP-адрес, а сервер Ignite-UX ответит и начнет инсталляцию.

Этот метод работает лучше всего для систем, которые совместно используют подсеть с сервером Ignite-UX. Для конфигураций, в которых клиент не включен в подсеть этого сервера, система HP перечисляет ряд опций в руководстве Ignite-UX Administration Guide.

## Автоматизация инсталляций Ignite-UX

Конфигурация сетевой загрузки Ignite-UX — это необходимое условие для автоматизированной инсталляции, однако конфигурирование Ignite-UX без задания подробных данных автоматической загрузки приводит к интерактивной инсталляции. Кроме того, Ignite-UX может следующее:

- использовать конфигурацию, сохраненную от предыдущей инсталляции, для автоматизации будущих инсталляций;
- опираться на файлы конфигурации, которые могут быть установлены для каждого клиента, для нужных версий или просто по желанию администратора;
- задавать стандартные значения для некоторых параметров конфигурации (например, DNS-серверы) и использовать другие значения для выбора во время интерактивной инсталляции.

Файлы автоматизированной инсталляции размещаются в каталоге **/opt/ignite/data**, а несколько примеров и образцы конфигураций включены в пакет инсталляции Ignite-UX. Поэтому вам лучше всего начать с просмотра подкаталогов **release** и **example**.

# 12.4. Инсталляция системы AIX с помощью сетевого менеджера инсталляции

Сетевой менеджер инсталляции (Network Installation Manager — NIM) — это AIX-ответ инсталляторам Kickstart, JumpStart и Ignite-UX. Версии NIM, начиная с AIX 5.3, могут также инсталлировать системы Linux. "Главный" NIM-сервер инсталлирует клиентов от одного или нескольких образов инсталляции, где под клиентом может подразумеваться автономный компьютер, "пустая" (т.е. без диска и данных) рабочая станция или рабочий раздел диска.[2] Инсталляции опираются на использование протоколов TFTP, NFS и DHCP или BOOTP, почти как в других системах.[3] Сетевой менеджер инсталляции NIM включен в стандартный носитель инсталляции AIX.

Все NIM-среды имеют, по крайней мере, один "сервер ресурсов", который предлагает клиентам некоторый набор программ. Этот сервер и "главный" NIM-сервер могут использовать одну систему или разные. Для улучшения качества процесса инсталляции среды, которые имеют сложные сетевые топологии или географически разделены, должны использовать локализованные серверы ресурсов.

Существует три способа конфигурации NIM:

- путем использования веб-ориентированного системного менеджера;

[2] Бездисковый клиент не имеет жесткого диска для локальных файловых систем и для хранения данных использует сетевые службы. Клиент без данных имеет только локальное пространство для свопинга и, возможно, файловые системы `/tmp` и `/home`.

[3] Поскольку все поставщики используют приблизительно одни и те же протоколы и архитектуры, то было бы хорошо, если бы они скооперировались и создали одну стандартную систему инсталляции, не правда ли? :-)

- с помощью "быстрых" путей (с минимальным временем распространения) **smit nim** или **smit eznim**;
- из командной строки с помощью утилиты **nim**.

Мы считаем, что SMIT (System Manager Interface Tool — интерфейсные средства администратора системы) — самый быстрый и самый удобный интерфейс для конфигурирования NIM-среды. Версия "EZ" охватывает большинство часто выполняемых таких NIM-задач, как быстрая настройка главного сервера, обновление, резервирование или реинсталляция существующих клиентов и конфигурирование новых клиентов. Полнофункциональная версия **smit nim** несколько усложнена из-за использования EZ NIM, но зато содержит некоторые дополнительные возможности конфигурации, например, позволяя включать пользовательские программные пакеты и более гибкие средства управления инсталлированными клиентами.

Если вы настаиваете на выполнении операций из командной строки, тогда лучше всего начать с утилиты **nim_master_setup**. (В действительности SMIT-варианты EZ-NIM для конфигурирования главного сервера просто вызывают утилиту **nim_master_setup** с заданными параметрами.) Эта утилита инициализирует файловые системы для программных NIM-ресурсов, создает необходимые файлы конфигурации и копирует образец файла конфигурации клиента, который вы можете отредактировать для своих локальных клиентов.

Самый распространенный вариант использования утилиты — **nim_master_setup -a device=/dev/cd0**, где **/dev/cd0** — устройство, которое содержит носитель инсталляции для соответствующей версии системы AIX. В отличие от большинства других систем инсталляции, описанных в этой главе, главный NIM-сервер может инсталлировать выпуски AIX только того же (или более раннего) уровня модификации: например, сервер версии AIX 5.2 не может инсталлировать системы версии AIX 6.1.

Самые полезные NIM-ориентированные утилиты командной строки перечислены в табл. 12.4.

**Таблица 12.4. NIM-ориентированные утилиты командной строки**

| Утилита | Назначение |
|---|---|
| **nim_master_setup** | Инсталлирует и конфигурирует главный NIM-сервер |
| **nim_update_all** | Обновляет ресурсы инсталляции и клиентов |
| **nim_clients_setup** | Определяет новых клиентов и инициализирует инсталляцию операционной системы |
| **nim_master_recover** | Восстанавливает главную NIM-базу данных на новом сервере |
| **nim** | Множество функций: конфигурирует ресурсы, определяет клиентов и т.д. |
| **nimclient** | Извлекает ресурсы (например, обновления) из сервера (запускает на стороне клиентов) |

# 12.5. Управление пакетами

Все варианты UNIX и Linux включают какую-нибудь систему управления пакетами, которая помогает управлять конфигурацией. Традиционно пакеты использовались для распространения программного обеспечения, однако их можно применять также для распространения конфигурационных файлов и административных данных. Пакеты имеют ряд преимуществ по сравнению с неструктурированными архивами **.tar.gz**.

Наверное, важнее всего то, что процесс инсталляции пакета протекает как одна транзакция: при возникновении ошибки инсталляцию можно прервать или повторить.

Пакетные инсталляторы обычно знают о конфигурационных файлах и, как правило, не перезаписывают локальные настройки, выполненные системным администратором. Они или создают резервные копии изменяемых файлов, или предлагают образцы конфигурационных файлов под другими именами (например, **pkg.conf.rpmnew**). Если после инсталляции пакета в системе обнаруживаются нарушения, то, теоретически, можно все восстановить в прежнее состояние. Естественно, то, что написано в теории, не всегда проходит на практике, поэтому делать это в производственной системе нужно лишь после проверки установки на тестовой системе.

Системы управления пакетами "понимают" внутрипакетные зависимости, что позволяет разработчикам пакетов быть уверенными в правильной инсталляции не только самих приложений, но и всех библиотек и вспомогательных файлов, от которых они зависят. Следует иметь в виду, что не все системы управления пакетами в полной мере справляются с внутрипакетными зависимостями: одни — лучше, другие — хуже.

В ходе инсталляции пакета могут также выполняться различные сценарии, поэтому иногда они могут сделать гораздо больше, чем просто выгрузить новые файлы.

С помощью пакетов удобно распространять изменения локальных настроек. Можно создать пакет, который в процессе инсталляции соберет сведения о компьютере (или получит их из локальной базы данных) и на их основе модифицирует конфигурационные файлы. В виде пакетов можно рассылать локальное ПО вместе со вспомогательными файлами и даже сторонние приложения, первоначально имевшие другой формат. Пакетам можно присваивать версии, с тем чтобы при инсталляции новой версии пакета автоматически обновлялись зависящие от него компоненты.

Можно также использовать механизм зависимостей для создания групп пакетов. Например, можно создать такой пакет, который сам по себе ничего не инсталлирует, но от него зависит много других "заплат". Появление новой версии пакета приводит к автоматической установке всех этих "заплат" за один раз.

## 12.6. Управление Linux-пакетами

В системах Linux широко распространены два формата пакетов. В Red Hat, SUSE и большинстве других дистрибутивов применяется диспетчер пакетов RPM (Red Hat Package Manager). В Ubuntu используются пакеты отдельного формата **.deb** (названного "в честь" дистрибутива Debian, на основе которого был создан дистрибутив Ubuntu). Оба формата функционально идентичны.

Преобразование из формата в формат можно произвести без каких-либо трудностей с помощью утилиты **alien** (**kitenet.net/programs/alien**). Ей ничего не известно о программах данного пакета, поэтому если содержимое еще не совместимо с вашим дистрибутивом, **alien** не поможет. Вообще, лучше всего использовать "родной" механизм упаковки, используемый в вашем дистрибутиве.

Системы упаковки RPM и **.deb** теперь работают в виде двухуровневых средств управления конфигурацией. На нижнем уровне находятся средства, которые инсталлируют, деинсталлируют и запрашивают пакеты: **rpm** для RPM и **dpkg** для **.deb**.

Над этими командами находятся системы, которые знают, как нужно производить поиск пакетов в Интернете, анализировать зависимости между пакетами и модернизировать все пакеты в системе. Система yum (Yellowdog Updater, Modified) работает с системой RPM. Система Red Hat Network работает для Red Hat Enterprise Linux и использует

RPM. Система Advanced Package Tool (APT) первоначально была создана для работы с пакетами **.deb**, а сейчас она может работать также с пакетами RPM.

Ниже будет дан обзор низкоуровневых команд **rpm** и **dpkg**. В разделе 12.7 поговорим о комплексных системах обновления.

## Команда rpm: управление пакетами RPM

Команда **rpm** инсталлирует, проверяет и запрашивает состояние пакетов. Когда-то она еще и создавала пакеты, однако сейчас эта функция отведена команде **rpmbuild**. Тем не менее опции **rpm** по-прежнему имеют сложные взаимодействия и вместе могут использоваться только в некоторых комбинациях. Воспринимать утилиту **rpm** нужно так, будто это несколько разных команд с одним и тем же именем.

Режим, который вы выбираете для работы **rpm** (например, **-i** или **-q**), определяет, к каким функциям этой утилиты вы хотите обратиться. Команда **rpm --help** перечислит все опции, разбивая их по режимам. Если вам часто приходится иметь дело с пакетами RPM, нужно будет внимательно прочесть главную страницу.

Обычно используются опции **-i** (install), **-U** (upgrade), **-e** (erase) и **-q** (query). Последняя опция является довольно сложной в том плане, что она служит для включения остальных опций; чтобы изложить определенный вопрос, вам нужно предоставить дополнительный флаг командной строки. Например, команда **rpm -qa** отображает список всех пакетов, инсталлированных в системе.

Давайте рассмотрим небольшой пример. Предположим, требуется инсталлировать новую версию пакета OpenSSH, поскольку опубликован бюллетень, в котором сообщается о том, что в предыдущей версии выявлена брешь. После того как пакет загружен на локальный компьютер, для его инсталляции достаточно ввести команду **rpm -U**.

```
redhat$ sudo rpm -U openssh-2.9p2-12.i386.rpm
error: failed dependencies:
openssh = 2.9p2-7 is needed by openssh-askpass-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-askpass-gnome-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-clients-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-server-2.9p2-7
```

Гм... Похоже, не все так просто! Как видите, инсталлированная в настоящий момент версия 2.9p2-7 связана с рядом других пакетов. Команда **rpm** не позволяет обновить пакет OpenSSH, так как это изменение затрагивает другие пакеты. Этот тип конфликта встречается постоянно, поэтому лучше разрабатывать такие системы, как APT и yum. В реальном мире мы вряд ли пытались бы распутывать зависимости вручную, однако в качестве примера сделаем это для утилиты **rpm**.

Можно прибегнуть к принудительному обновлению с помощью опции **--force**, но это вряд ли оправдано. Информация о зависимостях присутствует здесь для того, чтобы сэкономить ваше время и избавить вас от лишних проблем, а не для того, чтобы запутать вас. Для любого системного администратора нет ничего хуже, чем нарушение работы SSH в дистанционной системе.

Вместо этого мы будем использовать обновленные версии пакетов зависимостей. Если бы мы были находчивыми, то перед модернизацией могли бы определить, что остальные пакеты зависят от OpenSSH.

```
redhat$ rpm -q --whatrequires openssh
openssh-askpass-2.9p2-7
openssh-askpass-gnome-2.9p2-7
openssh-clients-2.9p2-7
openssh-server-2.9p2-7
```

Теперь предположим, что обновленные копии необходимых пакетов получены. Можно инсталлировать их последовательно, но команда **rpm** все берет на себя. Достаточно указать список пакетов в командной строке, и команда **rpm** отсортирует их в соответствии с имеющимися зависимостями.

```
redhat$ sudo rpm -U openssh-*
```

Проверим результат.

```
redhat$ rpm -q openssh
openssh-2.9p2-12
```

Все получилось!

Обратите внимание на тот факт, что **rpm** понимает, о каком пакете идет речь, даже если не указывать полное имя или версию пакета.

## Команда dpkg: управление пакетами .deb в системе Ubuntu

В Debian аналогом команды **rpm** является команда **dpkg**. К полезным ее опциям относятся **--install**, **--remove**, а **-l** перечисляет пакеты, инсталлированные в системе. Обратите внимание на то, что команда **dpkg --install**, выполненная в отношении пакета, уже находящегося в системе, перед инсталляцией удаляет предыдущую версию пакета.

С помощью команды **dpkg -l | grep** *пакет* легко определить, инсталлирован ли уже указанный пакет. Например, чтобы отыскать HTTP-сервер, выполните следующую команду.

```
ubuntu$ dpkg -l | grep -i http
ii  lighttpd 1.4.13-9ubuntu4 A fast webserver with minimal memory footpri
```

В результате будет найдена программа **lighttpd** — облегченный веб-сервер (прекрасный открытый исходник). Начальные буквы **ii** означают, что заданная программа уже инсталлирована.

Предположим, группа, занимающаяся вопросами безопасности системы Ubuntu, выпустила "заплату" к редактору nvi. После загрузки "заплаты" нужно выполнить команду **dpkg** для ее установки. Из показанного ниже примера видно, что эта команда гораздо более многословна, чем **rpm**, и сообщает о том, что именно она делает.

```
ubuntu$ sudo dpkg --install ./nvi_1.79-16a.1_i386.deb
(Reading database ... 24368 files and directories currently installed.)
Preparing to replace nvi 1.79-14 (using ./nvi_1.79-16a.1_i386.deb) ...
Unpacking replacement nvi ...
Setting up nvi (1.79-16a.1) ...
Checking available versions of ex, updating links in /etc/alternatives ...
(You may modify the symlinks there yourself if desired - see 'man ln'.)
Leaving ex (/usr/bin/ex) pointing to /usr/bin/nex.
Leaving ex.1.gz (/usr/share/man/man1/ex.1.gz) pointing to
  /usr/share/man/man1/nex.1.gz.
...
```

Теперь можно ввести команду **dpkg -l**, чтобы узнать, все ли прошло нормально. Флаг **-l** допускает наличие шаблона поиска, благодаря чему можно получить информацию только по редактору nvi.

```
ubuntu$ dpkg -l nvi
Desired=Unknown/Install/Remove/Purge
|Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:uppercase=bad)
||/Name Version Description
+++-==========-==========-==============================
i i nvi 1.79-16a.1 4.4BSD re-implementation of vi.
```

Инсталляция завершилась успешно.

## 12.7. Использование высокоуровневых систем управления пакетами в Linux

Системы управления пакетами, такие как APT и yum, а также Red Hat Network, ставят перед собой следующие задачи:

- упростить определение местонахождения и загрузку пакетов;
- автоматизировать процесс обновления или модернизации систем;
- способствовать управлению зависимостей между пакетами.

Понятно, что, помимо команд, на стороне клиентов эти системы должны выполнять множество других функций. Все они требуют, чтобы компании, обеспечивающие поддержку дистрибутивов, организовывали свои предложения в согласованном порядке, чтобы клиенты могли иметь доступ к их ПО осознанно.

Так как ни один поставщик не в состоянии предложить программы для Linux "на любой вкус", любая система допускает существование множества хранилищ программного обеспечения. Хранилища могут быть локальными по отношению к вашей сети, поэтому эти системы предлагают первоклассную базу, чтобы вы могли создать собственную систему внутреннего распределения.

Служба Red Hat Network неразрывно связана с дистрибутивом Red Hat Enterprise Linux. Это коммерческая служба, пользование которой стоит денег, зато она обладает привлекательным интерфейсом и более мощными возможностями в плане автоматизации, чем APT и yum. Служба Red Hat Network — это "отполированная" открытая версия дорогостоящего Red Hat-сервера Satellite Server. Клиент может ссылаться на хранилища yum и APT, и эта возможность позволяет таким дистрибутивам, как CentOS, адаптировать клиентский графический интерфейс под нестандартное использование.

APT документирована гораздо лучше, чем Red Hat Network, и к тому же бесплатна. Она также является более гибкой в плане настройки. APT разрабатывалась для Debian и dpkg, но была расширена и теперь может работать с RPM. Доступными являются все версии, работающие со всеми нашими примерами дистрибутивов. На данный момент система APT более других подходит на роль универсального стандарта для распространения программного обеспечения.

Программа yum — это аналог APT, предназначенный для RPM. yum является также стандартным администратором пакетов для Red Hat Enterprise Linux версии 5, хотя может работать в любой системе, основанной на RPM, при условии что вы сможете указать ей хранилища, имеющие соответствующий формат.

Нам нравится APT, и мы считаем, что это разумный выбор, если нужно настроить собственную автоматизированную сеть распределения пакетов. Более подробно об этом

можно прочитать в разделе "Создание локального зеркала хранилища" далее в этой главе. Но в большинстве случаев безопаснее всего использовать систему управления пакетами, которая входит в состав выбранного вами дистрибутива.

В SUSE реализована собственная система управления пакетами на основе пакетов RPM (известная как ZYpp) с интерфейсом командной строки (именуемым Zypper). В дополнение к обычным функциям (конфигурация репозитория, инсталляция пакета и запросы статуса), Zypper прекрасно зарекомендовал себя в реализации анализа зависимостей. Версия Zypper 1.0 вышла в составе openSUSE 11.1. Подробнее о Zypper см. ниже в этой главе.

## Хранилища пакетов

Дистрибуторы Linux поддерживают хранилища программного обеспечения, которые работают совместно с выбранными ими системами управления пакетами. Конфигурация, выбираемая по умолчанию для системы управления пакетами, обычно указывает на один или несколько хорошо известных веб- или FTP-серверов, находящихся под управлением дистрибьюторов.

Однако из этого факта нельзя сразу понять, что может содержаться в таких хранилищах. Должны ли они включать только наборы пакетов, принятых в качестве официальных, основных, версий? Официальные версии плюс текущие обновления системы защиты? Современные версии всех пакетов, которые существовали в официальных выпусках? Полезные программы сторонних производителей, которые официально не поддерживаются дистрибьютором? Исходный код? Бинарные файлы для многочисленных архитектур аппаратных средств? Если вы запускаете **apt-get upgrade**, **yum upgrade** или **zypper dup**, чтобы обновить систему, что именно под этим подразумевается?

Вообще, системы управления пакетами должны отвечать на все эти вопросы и облегчать организациям выбор специфических профилей, которые они хотят включить в их "мир программного обеспечения". Следующие концепции помогут структурировать этот процесс.

"Выпуск" (release) — это самосогласованное отображение "вселенной" пакета. Прежде чем наступила эпоха Интернета, именованные выпуски операционных систем были более или менее постоянными и были связаны с одним определенным моментом времени; "заплаты" системы защиты создавались отдельно. В наши дни выпуск представляет собой более расплывчатое понятие. Выпуски выходят во время обновления пакетов. Некоторые выпуски, такие как Red Hat Enterprise Linux, предназначены специально для того, чтобы задержать выходы новых версий; по умолчанию в них включаются только обновления защиты. Остальные выпуски, такие как бета-версии, меняются часто и существенно. Однако во всех случаях выпуск является базовой линией, трендом, или той мерой, "до которой я хочу обновить свою систему".

"Компонент" (component) — подборка программ в рамках выпуска. Дистрибутивы имеют различные отличия, однако общим является отличие между основным программным обеспечением, предлагаемым дистрибьютором, и дополнительным программным обеспечением, предлагаемым энтузиастами. Другое отличие, присущее миру Linux, кроется между свободными частями открытого исходного кода выпуска и частями, связанными с некоторым соглашением о коммерческом использовании.

Отдельного внимания со стороны администратора заслуживают минимально активные компоненты, которые включают только исправления в системе защиты. Некоторые

выпуски позволяют комбинировать компонент защиты с постоянным базовым компонентом для создания относительно стабильной версии дистрибутива.

"Архитектура" (architecture) — это специфический класс аппаратных средств (оборудования). Предполагается, что компьютеры, относящиеся к некоторому классу архитектуры, будут иметь одинаковые характеристики, позволяющие запускать на них одинаковые исполняемые файлы. Архитектуры являются специфическими экземплярами выпусков (например, "Ubuntu Karmic Koala for x86_64"). Так как компоненты являются подразделениями выпусков, для каждого из них существует соответствующий экземпляр, характерный для данной архитектуры.

Индивидуальные пакеты являются элементами, составляющими компоненты, а следовательно, и выпуски. Пакеты обычно являются специфическими для архитектуры и выходят в виде версии, не зависящей от главного выпуска и остальных пакетов. Соответствие между пакетами и выпусками является неявным в том плане, как производится настройка сетевого хранилища.

Существование компонентов, которые не поддерживаются дистрибьютором (например, "universe" и "multiverse" для Ubuntu), поднимает вопрос о том, как эти компоненты относятся к основному выпуску ОС. Можно ли о них говорить как о настоящих "компонентах" специфического выпуска или же они представляют собой нечто другое? С точки зрения управления пакетами ответ прост: "universe" — это настоящий компонент. Они связаны со специфическим выпуском и выходят вместе с ним. Разделение управления интересно с точки зрения администрирования, но не влияет на системы управления пакетами.

## Служба Red Hat Network

После того как система Red Hat перестала входить в группу "потребительских товаров" Linux, Red Hat Network стала платформой для управления системой для Red Hat Enterprise Linux. Подписываясь на нее, вы приобретаете право доступа к Red Hat Network. В простейшем виде служба Red Hat Network представляет собой популярный веб-портал и список рассылки. В этом смысле она мало чем отличается от тех служб уведомления о "заплатах", которые уже много лет эксплуатируются различными поставщиками UNIX-систем. Дополнительные возможности открываются лишь при условии, что вы готовы за них заплатить. Информацию и текущие расценки можно узнать на сайте **rhn.redhat.com**.

Служба Red Hat Network предлагает веб-ориентированный интерфейс для загрузки новых пакетов (можно также работать в режиме командной строки). Начиная с версии Red Hat Enterprise 5, в качестве интерфейса командной строки (Command Line Interface — CLI) используется **yum** (до этого времени использовался громоздкий и проблемный инструмент, именуемый **up2date**), который даже позволяет загружать и инсталлировать пакеты без вмешательства человека. После того как вы зарегистрируетесь, система будет получать все необходимые "заплаты" в автоматическом режиме.

Недостатком такого подхода является то, что решения об обновлении системы принимаются за вас. Определите для себя, в какой степени следует доверять разработчикам Red Hat (а также тех программ, которые они предлагают в виде пакетов).

Разумным компромиссом будет выделение одного из компьютеров в качестве сервера автоматических обновлений. Периодически можно создавать образы серверной системы и проверять, в какой степени они подходят для внутреннего распространения.

## APT: усовершенствованное средство управления пакетами

APT (Advanced Packaging Tool) — это одна из зрелых систем управления пакетами. С помощью всего лишь одной команды **apt-get** можно модернизировать всю систему и программное обеспечение; также с ее помощью (как и с помощью Red Hat Network) можно постоянно сохранять ваши "коробочные" версии современными без вмешательства со стороны человека.

Первое правило, которого следует придерживаться при использовании утилиты **apt-get** в Ubuntu (это касается всех средств управления пакетами этой системы), требует игнорировать утилиту **dselect**, являющуюся клиентской надстройкой системы. Речь не идет о том, что утилита **dselect** плохая, просто ее интерфейс неудачен и может напугать новичка. В документации вы можете найти рекомендации в пользу использования **dselect**, тем не менее не принимайте их во внимание и используйте APT.

В случае типичной инсталляции Ubuntu, которая загружается с одного из стандартных "зеркал", узнать список доступных пакетов можно на веб-сайте `packages.ubuntu.com`. Этот веб-сайт имеет удобную поисковую систему. Если же создается внутренний сервер APT (см. раздел "Создание локального зеркала хранилища" ниже в этой главе), то, конечно, администратор сам знает, какие пакеты доступны, и может сформировать их список.

Дистрибутивы обычно имеют пустые пакеты, которые существуют для того, чтобы формировать списки зависимых пакетов. Утилита **apt-get** автоматически загружает и инсталлирует зависимые пакеты, благодаря чему можно легко устанавливать или обновлять блоки пакетов. Например, инсталлируя пакет **gnome-desktop-environment**, можно быть уверенным в том, что установлено все необходимое для пользовательского интерфейса GNOME.

Если файл **/etc/apt/sources.list** (см. его подробное описание ниже в этой главе) настроен, а имя необходимого пакета известно, то осталось лишь выполнить команду **apt-get update**, чтобы обновить информацию о пакетах. После этого от имени привилегированного пользователя нужно ввести команду **apt-get install** *имя_пакета*, которая, собственно, и осуществит инсталляцию пакета. Эта же команда обновит пакет, если он уже инсталлирован.

Предположим, требуется установить новую версию пакета **sudo**, в котором устранена очередная брешь. Сначала не помешает выполнить команду **apt-get update**.

```
ubuntu$ sudo apt-get update
Get:1 http://http.us.debian.org stable/main Packages [824kB]
Get:2 http://non-us.debian.org stable/non-US/main Release [102B]
...
```

Теперь можно приступить к получению пакета. Обратите внимание на то, что мы используем команду **sudo** в процессе инсталляции нового пакета **sudo** — утилита **apt-get** способна обновлять даже те пакеты, которые в настоящий момент используются.

```
ubuntu$ sudo apt-get install sudo
Reading Package Lists... Done
Building Dependency Tree... Done
1 packages upgraded, 0 newly installed, 0 to remove and 191 not upgraded.
Need to get 0B/122kB of archives. After unpacking 131kB will be used.
(Reading database ... 24359 files and directories currently installed.)
Preparing to replace sudo 1.6.1-1 (using .../sudo_1.6.9p10-1ubuntu3.4_i386.deb)
...
```

```
Unpacking replacement sudo ...
Setting up sudo (1.6.2p2-2) ...
Installing new version of config file /etc/pam.d/sudo ...
```

## Конфигурирование apt-get

Сконфигурировать утилиту **apt-get** несложно. Хорошие инструкции можно найти в документации по управлению пакетами в системе Ubuntu по адресу:

**help.ubuntu.com/community/AptGet/Howto**

Самый важный конфигурационный файл утилиты называется **/etc/apt/sources.list**. В нем сообщается, где искать пакеты. В каждой строке файла указывается следующее.

- Тип пакета. В настоящее время это deb или deb-src для пакетов в стиле Debian либо rpm или rpm-src — для RPM.
- URL-адрес файла, компакт-диска, сервера HTTP или FTP, где находятся пакеты.
- "Дистрибутив" (на самом деле — название выпуска), если нужно работать с несколькими версиями пакета. Дистрибьюторы используют его для главных выпусков, однако вы можете использовать его для других целей, если вам нужны внутренние системы распространения.
- Необязательный список компонентов, т.е. категорий пакетов в рамках дистрибутива.
- Стандартная конфигурация вполне приемлема, если только не нужно создавать собственное хранилище пакетов. Исходные пакеты загружаются при использовании строк, начинающихся с **deb-src**.

Работая в системе Ubuntu, вы почти наверняка захотите включить в нее компонент "universe", который предоставляет доступ к открытым программным средствам большого (по объему) мира Linux. Пакеты "multiverse" включают такие неоткрытые исходные тексты, как некоторые утилиты и компоненты VMware.

В процессе редактирования файла **sources.list** вам следует перенастроить отдельные записи для указания адреса "зеркала", задав более близко расположенный сервер. Полный список "зеркал" Ubuntu находится по адресу: **launchpad.net/ubuntu/+archivemirrors**. Это динамический (и длинный) список зеркал, который регулярно изменяется, поэтому обязательно отслеживайте изменения между выпусками. Убедитесь, что в качестве источника указан элемент **security.ubuntu.com**, и тогда вы точно получите доступ к самым последним "заплатам", связанным с мерами по укреплению безопасности.

Пример файла `/etc/apt/sources.list`

В показанном ниже примере в качестве источника пакетов для загрузки "основных" компонентов Ubuntu используется адрес **us.archive.ubuntu.com** (эти компоненты полностью поддерживаются командой разработчиков Ubuntu). Кроме того, этот список (**sources.list**) включает неподдерживаемые (но с открытым исходным кодом) пакеты "universe" и небесплатные неподдерживаемые пакеты в компоненте "multiverse". В каждом компоненте также предусмотрено хранилище для обновлений, т.е. пакетов с исправленными ошибками. Наконец, последние шесть строк предназначены для обновлений, связанных с безопасностью.

```
# Общий формат: тип URL-адрес дистрибутив [компоненты]
deb http://us.archive.ubuntu.com/ubuntu/ karmic main restricted
```

```
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic main restricted
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates main restricted
deb http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
deb http://us.archive.ubuntu.com/ubuntu/ karmic multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic multiverse
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates multiverse
deb http://security.ubuntu.com/ubuntu karmic-security main restricted
deb-src http://security.ubuntu.com/ubuntu karmic-security main restricted
deb http://security.ubuntu.com/ubuntu karmic-security universe
deb-src http://security.ubuntu.com/ubuntu karmic-security universe
deb http://security.ubuntu.com/ubuntu karmic-security multiverse
deb-src http://security.ubuntu.com/ubuntu karmic-security multiverse
```

Поля *дистрибутив* и *компоненты* помогают утилите **apt-get** ориентироваться в иерархии файловой системы хранилища Ubuntu, которая характеризуется стандартизированным размещением. Корневой дистрибутив для каждого выпуска может быть помечен как **intrepid** (смелый), **jaunty** (стильный) или **karmic** (кармический). Доступными компонентами обычно являются **main**, **universe**, **multiverse** и **restricted**. Если вас устраивает наличие неподдерживаемых (и с ограниченной лицензией в случае **multiverse**) программ в вашей среде, добавьте хранилища **universe** и **multiverse**.

## Создание локального зеркала хранилища

Если планируется применять утилиту **apt-get** для большого числа компьютеров, то потребуется локальное кеширование пакетов — загружать копии всех пакетов для каждого компьютера было бы неразумно. Совсем не трудно сконфигурировать зеркало хранилища, которое было бы удобным для локального администрирования. От вас требуется лишь отслеживать обновления с помощью "заплат", связанных с безопасностью.

Лучше всего и удобно для такой работы использовать пакет **apt-mirror**, который доступен на сайте **apt-mirror.sourceforge.net**. Вы можете также инсталлировать пакет из компонента **universe** с помощью команды **sudo apt-get install apt-mirror**.

После инсталляции пакета **apt-mirror** в каталоге **/etc/apt** вы найдете файл **mirror.list**. Это "теневая" версия **sources.list**, но она используется только как источник для зеркального отображения операций. По умолчанию **mirror.list** содержит все хранилища для запуска версии Ubuntu.

Для того чтобы в действительности воспроизвести хранилища в **mirror.list**, достаточно запустить на выполнение **apt-mirror**.

```
ubuntu$ sudo apt-mirror
Downloading 57 index files using 20 threads...
Begin time: Sat Aug 29 18:53:44 2009
[20]... [19]... [18]... [17]... [16]... [15]... [14]...
```

По умолчанию пакет **apt-mirror** помещает свои копии хранилища в каталог **/var/spool/apt-mirror**. Можно закомментировать **set base_path** в **mirror.list**, но не забудьте создать в новом корневом каталоге зеркала подкаталоги **mirror**, **skel** и **var**.

Пакету **apt-mirror** требуется много времени для первого прохода, поскольку ему приходится отображать множество гигабайтов данных (на данный момент приблизительно 40 Гбайт для выпуска Ubuntu). Последующие проходы протекают быстрее и должны быть реализованы автоматически (из демона **cron**). Для того чтобы избавиться от уже устаревших файлов, можете запустить сценарий **clean.sh** из подкаталога **var** своего зеркала.

Для того чтобы приступить к использованию своего зеркала, воспользуйтесь (с помощью своего любимого веб-сервера) базовым каталогом через протокол HTTP. Мы любим использовать символические ссылки на веб-корень. Вот пример.

```
ln -s /var/spool/apt-mirror/us.archive.ubuntu.com/ubuntu /var/www/ubuntu
```

Для того чтобы заставить клиентов использовать свое локальное зеркало, отредактируйте файлы **sources.list** так, как если бы выбрали нелокальное зеркало.

## Автоматизация работы утилиты apt-get

Утилиту **apt-get** можно запускать по графику с помощью демона **cron**. Даже если пакеты не инсталлируются автоматически, полезно регулярно выполнять команду **apt-get update**, чтобы следить за обновлениями сводных файлов.

Команда **apt-get dist-upgrade** загрузит и инсталлирует новые версии любых пакетов, имеющихся на локальном компьютере. Флаг **dist-upgrade** эквивалентен флагу **upgrade**, но включает более точную обработку зависимостей. Флаг **dist-upgrade** может удалить пакеты, которые утилита посчитает несовместимыми с модернизированной системой, так что к этому нужно быть готовым.

Можно даже позволить компьютерам выполнять обновления в автоматическом режиме. Для этого предназначена опция **-yes**, при наличии которой на любой вопрос, задаваемый утилитой **apt-get**, будет выдано оптимистичное "Да!". Имейте в виду, что некоторые обновления (например, пакеты ядра) могут не вступать в силу до тех пор, пока система не будет перезагружена.

Обычно не рекомендуется автоматически загружать обновления непосредственно с "зеркал" дистрибутивов. Другое дело, когда имеются внутренние серверы APT и система управления версиями. Следующий маленький сценарий позволит клиенту поддерживать синхронизацию с сервером APT.

```
#!/bin/sh
apt-get update
apt-get -yes dist-upgrade
```

Этот сценарий может запускаться демоном **cron** (см. главу 9) каждую ночь. Можно также создать ссылки на него в сценариях запуска системы (см. главу 3), чтобы обновления выполнялись на этапе начальной загрузки.

Если процедуры обновления запускаются на большом числе компьютеров, необходимо распределить их по времени, чтобы не перегрузить сеть. Это можно сделать с помощью небольшого Perl-сценария, приведенного в разделе 19.2.

Если вы не вполне доверяете источнику пакетов, можно автоматически загружать их, но не инсталлировать. Это позволяет делать опция **--download-only** утилиты **apt-get**. Просмотрите полученные пакеты и определите сами, какие из них инсталлировать. Загруженные пакеты находятся в каталоге **/var/cache/apt**, который со временем может серьезно разрастись. Удаляйте неиспользуемые файлы командой **apt-get autoclean**.

## Система yum: управление выпусками для RPM

Система yum (Yellowdog Updater, Modified) представляет собой администратор метапакетов, основанный на RPM[4]. Называть yum клоном **apt-get**, пожалуй, неправильно, однако в плане тематики и реализации она очень похожа на **apt-get**, но на практике проще и медленнее. Yum — это официальная система управления пакетами для Red Hat Enterprise Linux; она предварительно устанавливается и на многих других дистрибутивах. При необходимости самую последнюю версию yum можно получить из хранилища пакетов вашего дистрибутива.

Как и в случае с **apt-get**, команда на стороне сервера (**yum-arch**) компилирует базу данных заголовочной информации из большого набора пакетов (нередко из целого выпуска). После этого база данных заголовков совместно используется пакетами посредством протоколов HTTP или FTP. Клиенты используют команду **yum** для выбора и инсталляции пакетов; **yum** выявляет ограничения зависимостей и выполняет дополнительные действия, необходимые для завершения процесса инсталляции требуемых пакетов. Если запрошенный пакет зависит от других пакетов, **yum** загружает и инсталлирует и эти пакеты.

Подобие между **apt-get** и **yum** расширяется на опции командной строки, которые понятны им обоим. Например, **yum install foo** загружает и инсталлирует самую свежую версию пакета **foo** (и его зависимости, если это необходимо). Однако существует, как минимум, одно "предательское" отличие: **apt-get update** обновляет кеш информации о пакетах **apt-get**, а **yum update** — каждый пакет в системе (аналогично команде **apt-get update**). Более того, есть еще команда **yum upgrade**, которая делает то же, что и **yum update**, но устаревшими приемами.

Команда **yum** не рассматривает частичные имена пакетов, если не включить символы универсализации оболочки (такие, как * и ?). Например, **yum update 'perl*'** обновляет все пакеты, имена которых начинаются с "perl". Не забывайте заключать символы универсализации в кавычки, чтобы избежать возникновения ошибок.

В отличие от **apt-get**, **yum** во время запуска по умолчанию сверяет информацию о пакетах, хранящуюся в кеше, с содержимым сетевого хранилища. Для того чтобы отменить этот процесс, используйте опцию **-C**, в результате чего **yum makecache** будет обновлять локальный кеш (на это уйдет некоторое время). К сожалению, опции **-C** недостаточно, чтобы повысить производительность медлительной **yum**.

Конфигурационным файлом **yum** является **/etc/yum.conf**. Он включает общие опции и указатели на хранилища пакетов. Можно активизировать одновременно множество хранилищ; каждое хранилище может быть связано с множеством URL-адресов.

## Система управления пакетами Zypper для SUSE: теперь еще более мощная!

После многих лет довольно вялого управления пакетами системы SUSE стали предлагать оптимальный инструмент в форме Zypper, полнофункционального диспетчера пакетов следующего поколения, созданного на базе RPM. Из всех описанных здесь инструментов Zypper предоставляет наиболее гибкие и мощные возможности для инсталляции, удаления и запроса пакетов. Это также единственная утилита, которая включает управление хранилищами из командной строки.

[4] Yum Fish Bait не следует путать с Live Prey Technology (LPT), yum3x.com.

Освоить этот инструмент будет нетрудно для тех, кто знаком с утилитой **apt-get** или **yum**. Основные команды Zypper представлены в табл. 12.5.

**Таблица 12.5. Команды Zypper**

| Команда | Назначение |
|---|---|
| **zypper addrepo** *uri* | Добавляет хранилище в рабочий комплект |
| **zypper dist-upgrade** | Обновляет систему, обеспечивая соответствие текущему выпуску дистрибутива |
| **zypper info** *пакеты* | Отображает информацию о пакетах |
| **zypper install** *пакеты* | Загружает и инсталлирует пакеты |
| **zypper list-updates** | Перечисляет все обновленные пакеты в хранилище |
| **zypper modifyrepo** *uri* | Модифицирует свойства хранилища |
| **zypper refresh** | Обновляет метаданные хранилища в локальном кеше |
| **zypper remove** *пакеты* | Демонтирует пакеты |
| **zypper repos** | Перечисляет хранилища в текущем рабочем комплекте |
| **zypper search** *строка* | Выполняет поиск пакетов с совпадающими именами |
| **zypper shell** (или **sh**) | Запускает интерактивный сеанс zypper |
| **zypper update** | Инсталлирует обновленные версии текущих пакетов |

В приведенном ниже примере мы использовали команду **zypper sh**, чтобы открыть оболочку Zypper для непосредственного ввода команд.

```
suse$ zypper sh
zypper>  repos
# | Alias            | Name                   | Enabled | Refresh
--+------------------+------------------------+---------+--------
1 | openSUSE 11.1-0  | openSUSE 11.1-0        | Yes     | No
2 | repo-debug       | openSUSE-11.1-Debug    | No      | Yes
3 | repo-non-oss     | openSUSE-11.1-Non-Oss  | Yes     | Yes
4 | repo-oss         | openSUSE-11.1-Oss      | Yes     | Yes
5 | repo-source      | openSUSE-11.1-Source   | No      | Yes
6 | repo-update      | openSUSE-11.1-Update   | Yes     | Yes
```

Вместо выполнения команды **zypper refresh** вручную для поддержания данных пакетов на современном уровне, вы могли бы разрешить автоматическое обновление с помощью команды **zypper -f**.

Файлы конфигурации Zypper, включая конфигурацию программных хранилищ, находятся в каталоге **/etc/zypp**. Большинству администраторов нет необходимости использовать эти файлы, но при возникновении такой необходимости вы должны знать, что эти файлы снабжены подробными комментариями и могут оказаться весьма полезными.

## 12.8. Управление пакетами для систем UNIX

Инсталляция программного обеспечения и управление пакетами — это та область, в которой Linux имеет явное преимущество перед традиционными операционными системами UNIX. Инсталлирование, модернизация и поиск программ в системе Linux стали практически тривиальными задачами для конечного пользователя или администратора.

Поисковые средства стали очень мощными, сообщества пользователей очень большими, а количество активных разработчиков исчисляется в тысячах.

В отличие от Linux, системы UNIX оставляют администраторам гораздо меньше пакетов для выбора и имеют хуже организованный контроль над существующими пакетами. В этом разделе мы рассмотрим программы управления пакетами, которые больше всего используются в каждом из наших примеров систем UNIX.

## Управление пакетами в Solaris

С момента выхода SunOS 2.0 в системах Solaris пакетированием традиционно занималась версия SVR4 (System V Release 4) с некоторыми инкрементными усовершенствованиями, которые сохранили "хромоту" системного уровня на протяжении почти двадцати лет. К сожалению, версия SVR4 остается дефектной в таких областях, как управление зависимостями, практичность, поддержка новых технологий (например, ZFS) и сетевых хранилищ. Для OpenSolaris разработчики решили "списать" старую систему и создать совершенно новую.

В настоящее время система OpenSolaris использует кроссплатформенную систему управления пакетами (Image Packaging System — IPS), которую можно определить как огромный шаг вперед по сравнению SVR4. В качестве ключевой архитектурной детали она включает сетевые хранилища. В дополнение к стандартным функциям управления пакетами, система предлагает инструменты, предназначенные для разработчиков пакетов и упрощающие создание и централизацию пакетов. Система IPS также обеспечивает обратную совместимость с унаследованными SVR4-пакетами.

На данный момент можно утверждать, что IPS-пакеты заметно неоднородны в таких форматах, как **.deb** или **RPM**. IPS-пакет — это не один файл, который вы могли бы легко скопировать в системах. Пакет, скорее, представляет собой коллекцию файлов, зависимостей и других данных, которые необходимо обслуживать из хранилища посредством IPS-демона **pkg.depotd**. Система IPS все еще остается в состоянии разработки, и более готовый к употреблению формат обещают представить буквально "на днях".

Команда **pkg** используется для выполнения большинства операций IPS: инсталляции, удаления, поиска, запроса статуса и т.д. Кроме того, она позволяет управлять хранилищами, хотя в **pkg**-документации и синтаксисе команд их называют "издателями" (publishers)[5]. Все эти команды — **pkg install**, **pkg uninstall**, **pkg search** и **pkg info** — выполняют ожидаемые функции. Команда **pkg image-update** подобна APT-команде **dist-upgrade**: она обновляет все инсталлированные пакеты до уровня самых последних доступных версий.

По умолчанию хранилище выпусков OpenSolaris является стандартным издателем. В настоящее время он содержит приблизительно 1 700 пакетов[6].

```
solaris$ pkg publisher

PUBLISHER TYPE STATUS URI
opensolaris.org (preferred) origin online http://pkg.opensolaris.org/rele…
```

Подробнее о системе IPS можно узнать, заглянув в man-страницы для **-s5 pkg**, а для получения информации о команде **pkg** см. man-страницы для **pkg**.

[5] Команда разработчиков различает понятия "хранилище" (repository) и "издатель" (publisher), но здесь мы будем считать их эквивалентными.

[6] Сравните с более чем 30 000 в Ubuntu Karmic Koala.

# Управление пакетами в HP-UX

Система управления пакетами в HP, формально именуемая как Software Distributor (SD), предложила пользователям HP-UX надежную систему управления пакетами уже начиная с версии 10. Это серьезный инструмент с набором функций, которые наверняка приведут любого системного администратора в состояние эйфории от эмоционального возбуждения.

- Большинство средств предлагает режимы (графический и "из командной строки"), зависящие от способа вызова.
- Программами можно управлять в удаленных системах посредством демона **swagentd**, который запускается во время загрузки и общается с системой через пользовательский протокол данных UDP либо через протокол TCP[7].
- Хранилища программ могут быть расположены на локальном носителе или в сетевых каталогах.
- Браузер заданий (job browser) позволяет администраторам отслеживать состояния инсталляций удаленных систем в режиме реального времени.

Ряд исполняемых файлов, имена которых начинаются с **sw** (это, в основном, **well**), составляет комплект инструментальных средств SD (Software Distributor). Отдельные его утилиты и их функции перечислены в табл. 12.6.

**Таблица 12.6. Список команд, реализованных в Software Distributor**

| Команда | GUI? | Назначение |
|---|---|---|
| **install-sd** | – | Реинсталлирует систему Software Distributor system |
| **sd** | Да[а] | Управляет удаленными заданиями: создание, диспетчеризация, мониторинг |
| **swacl** | – | Конфигурирует опции безопасности SD |
| **swask** | – | Выполняет интерактивные сценарии инсталляции |
| **swconfig** | – | Конфигурирует (или реконфигурирует) инсталлированные программы |
| **swcopy** | Да | Копирует пакеты в хранилище для будущей инсталляции |
| **swinstall** | Да | Инсталлирует пакеты программ из хранилища |
| **swjob** | – | Альтернатива команде **sd** в виде командной строки |
| **swlist** | Да | Перечисляет инсталлированные программы или программы, расположенные в хранилище |
| **swmodify** | – | Модифицирует каталог программ, инсталлированных в системе[б] |
| **swpackage** | – | Создает новые пакеты программ |
| **swreg** | – | Регистрирует хранилище программ |
| **swremove** | Да | Удаляет пакеты из системы или хранилища |
| **swverify** | – | Подтверждает целостность инсталлированного программного обеспечения |
| **swagentd** | – | Действует как командный агент SD (запускается во время загрузки) |

[а] Эта утилита оснащена только графическим интерфейсом (GUI) и не имеет командной строки. Эквивалентом командной строки является **swjob**.

[б] Называется также базой данных инсталлированных продуктов или IPD.

[7] В действительности демон `swagentd` вызывает процесс `swagent`, но он прозрачен, т.е. не заметен для пользователя.

Большинство команд SD поддерживает специальный флаг **-x**, который модифицирует стандартные опции применительно к данному инструменту. Например, среди опций команды **swinstall** есть такие: **allow_incompatible** (разрешает инсталляцию пакета, предназначенного для другой архитектуры) и **autoreboot** (при необходимости перезагружает систему после инсталляции). Полный список опций каждой утилиты представлен на соответствующей man-странице или в файле **/usr/lib/sw/sys.defaults**. Более удивительно то, что в файле **~/.swdefaults** могут быть сконфигурированы стандартные настройки для каждого пользователя.

Чаще всего используется команда **swinstall**, особенно теми администраторами, которые усердно инсталлируют "заплаты безопасности" сразу после их выхода в свет. Команда **swinstall -i** запускает интерактивную инсталляцию. Графический интерфейс (GUI) активизируется в случае, если используется система X; в противном случае вы будете использовать текстовый интерфейс.

К сожалению, система HP-UX не имеет удобного оперативного хранилища программ, из которого вы могли бы легко инсталлировать "заплаты". Однако утилита HP-UX Software Assistant позволяет анализировать систему с использованием ссылки на предоставляемый системой HP каталог "заплат", загружает соответствующие "заплаты" и создает программное хранилище, из которого вы можете брать "заплаты" и ими "латать дыры" в системе с помощью команды **swinstall**.

Рассмотрим несколько примеров. Для того чтобы инсталлировать "заплату безопасности" **PHKL_40197** из NFS-хранилища программ на узле **hpux.booklab**, мы должны выполнить такую команду.

```
hp-ux$ sudo swinstall -s hpux.booklab:/hpux/patches PHKL_40197
```

Команда **swinstall** выполняет конфигурационные сценарии автоматически во время инсталляции. Если вам позже понадобится реконфигурировать какой-нибудь пакет, используйте команду **swconfig**. Команда **swconfig** настраивает программный пакет для локальной системы (например, создавая модификации для файла в каталоге **/etc** или изменяя разрешения (права доступа) на использование каталога). Для того чтобы реконфигурировать пакет **sudo**, мы предлагаем выполнить следующую команду.

```
hp-ux$ sudo swconfig -x reconfigure=true sudo
```

Для того чтобы удалить этот пакет, мы бы выполнили команду **swremove sudo**. Команда **swverify** исследует пакет и выполняет проверку целостности его основных компонентов. Используйте команду **swverify**, чтобы устранить такие проблемы, как наличие испорченных файлов настроек или недостающие каталоги.

Команда **swlist** покажет вам программы, инсталлированные в системе или доступные в хранилище. Интерактивный интерфейс, как правило, удобнее для операций поиска, но посмотрите, насколько просто можно с помощью командной строки получить список Java-пакетов в нашем примере системы HP-UX.

```
hp-ux$  sudo swlist 'Java*'
# Инициализация...
# Контактирование с целевым объектом "hpux11"...
# Целевой объект:  hpux11:/
# Java15JDK 1.5.0.11.00 Java 1.5 JDK for HP-UX
Java15JDK.Jdk15 1.5.0.11.00 Java 1.5 JDK
Java15JDK.Jre15 1.5.0.11.00 Java 1.5 JRE
# Java15JRE 1.5.0.11.00 Java 1.5 JRE for HP-UX
Java15JRE.Jre15 1.5.0.11.00 Java 1.5 JRE
```

Операция удаления — мощная функция системы Software Distributor. Инструменты удаления позволяют системным администраторам записывать программы на удаленные

системы (по желанию это может быть сделано по заранее определенному расписанию). Большинство команд поддерживает удаленные операции, а некоторые из них (например, **swinstall**) поддерживают их в виде графического интерфейса пользователя (GUI). Операции удаления выполняются с помощью вызовов удаленных процедур (Remote Procedure Call — RPC), а безопасность контролируется с помощью списков управления доступом. Подробнее о конфигурировании удаленных операций см. **man 5 sd**.

## Управление программами в AIX


Большинство поставщиков систем UNIX, по крайней мере, попытались сохранить связь с Linux в области управления программными продуктами, но корпорация IBM решила выбрать для системы AIX стиль "каменного века" (т.е. периода господства ламповых ЭВМ). В большинстве случаев рекомендуется использовать "быструю" SMIT-команду (или fastpath-команду[8]) **install_software** (System Manager Interface Tool — интерфейсные средства администратора системы), которая неявно вызывает команду **installp**. Fastpath-команда **smit easy_install** — еще один вариант, который требует меньшего количества нажатий клавиш, чем команда **install_software**.

Команда **installp** обрабатывает пакетный IBM-формат (Backup File Format, или **.bff**), а также более старую версию формата RPM, используемую во многих системах Linux. К сожалению, в команде **installp** отсутствует множество функций, которые "испорченные" администраторы привыкли воспринимать как само собой разумеющееся. Речь идет об инсталляции пакетов из сетевого хранилища и даже запросе пакета. При добавлении пакетов в системы AIX мы всегда используем один из вариантов: **smit install_software** или **smit easy_install**.

AIX-команда **lslpp** выводит список инсталлированных программных пакетов, которые IBM называет "наборами файлов" (filesets). Команда **lslpp -L** перечислит все программные продукты в системе — с таким же успехом вы можете использовать и fastpath-команду **smit list_installed_sw**. Инсталлированные программные продукты имеют коды состояния и типа, которые означают условия и происхождение каждого пакета соответственно. Состояние может быть выражено одним из следующих кодовых вариантов: **applied** (принятый), **broken** (испорченный), **committed** (зафиксированный), **locked** (заблокированный), **obsolete** (устарелый) или **inconsistent** (несовместимый). Тип определяется таким набором кодовых вариантов: **fileset**, **product**, **component**, **feature**, **RPM** или **fix**.

# 12.9. Управление изменениями

Ошибки — неизменный аспект нашей жизни. Следовательно, очень важно отслеживать производимые вами изменения в конфигурации, чтобы в случае возникновения ошибок, вызванных этими изменениями, можно было без труда вернуться к надежной конфигурации. В этом разделе мы рассмотрим некоторые распространенные способы управления изменениями на уровне отдельных файлов. Вам остается лишь выбрать средства, которые будут совпадать с вашими потребностями и степенью сложности вашей организации.

[8] Под fastpath-командой подразумевается команда **smit** с параметром **fast path**, который позволяет сэкономить время выполнения за счет непосредственного перехода к нужному (для вашей задачи) меню или диалоговому окну с обходом меню верхнего уровня. — *Примеч. перев.*

## Создание резервных файлов

Эта команда, возможно, покажется вам знакомой.

```
$ cp bigfile.conf bigfile.bak
```

Вы увидели в этой команде что-то позорное? Ну да, ведь настоящие системные администраторы делают гораздо более сложное?

В действительности можно многое сказать в пользу этого вида импровизированного резервирования. Резервные файлы просто создавать, легко сравнивать (с помощью команды **diff**), и потом они создают контрольный след, по крайней мере, резервные файлы могут быть упорядочены по времени модификации. Они не требуют никакого дополнительного программного обеспечения и не допускают, чтобы кто-либо оставил систему резервного копирования в неоднозначном состоянии.

Но мы предлагаем свои варианты подхода к этой теме. Лучше всего переместить (с помощью команды **mv**) исходный файл куда-нибудь под другим именем, а затем скопировать его обратно, присвоив ему прежнее имя. С помощью опции **-p** команды **cp** можно сохранить настройки атрибутов файла. Эта процедура сохраняет время модификации исходного файла и обрабатывает случаи, когда активный процесс имеет открытую ссылку на исходный файл.

Будет еще лучше, если вы добавите короткий аплет-сценарий (scriptlet), подобный представленному ниже, в свой файл **~/.bash_profile** или **~/.profile**. Он определяет "команду" **backup** (в действительности функцию **bash**), которая "собирает" имя резервного файла, причем без вашего участия.

```
function backup () {
   newname=$1.`date +%Y%m%d.%H%M.bak`;
   mv $1 $newname;
   echo "Backed up $1 to $newname.";
  cp -p $newname $1;
}
```

Вот пример.

```
$ backup hello.py
Backed up hello.py to hello.py.20091030.1612.bak.
```

В имени файла кодируется дата и время резервирования и время последней модификации файла. Обе эти составляющие информации потенциально полезны. Данный сценарий кодирует время в таком виде, чтобы скомпонованные имена файлов позволяли корректно сортировать резервные файлы по дате.

Системы, которые регулярно резервируются на магнитную ленту, могут только выиграть от создания резервных файлов вручную. Восстановление из резервного файла всегда происходит быстрее и легче, чем с ленты; более того, вручную созданные резервные копии сохраняют дополнительный уровень предыстории, т.е. записей действий пользователя или системы.

## Формальные системы управления изменениями

Следующий уровень сложности и устойчивости занимают формальные системы управления изменениями, которые представляют собой пакеты программ, отслеживающих, архивирующих и обеспечивающих доступ к множеству версий файлов. Эти пакеты первоначально использовались при разработке программного обеспечения, однако они будут полезными и для системных администраторов.

Системы управления изменениями позволяют решить несколько проблем. Самое главное, они обеспечивают организованный доступ к отслеживанию истории изменений в файле, благодаря чему любые изменения можно понять исходя из контекста, а также можно восстанавливать ранние версии файлов. Связанные группы файлов могут вместе входить в одну версию, при этом будут учтены все зависимости. Наконец, системы управления изменениями координируют работу множества редакторов, поэтому вряд ли возникнет ситуация, когда чьи-то изменения будут навсегда утеряны[9] и несовместимые изменения, производимые в нескольких редакторах, окажутся одновременно активными.

В течение нескольких последних лет наблюдался бум в области систем управления версиями с открытым исходным кодом, причем доступные варианты имели между собой серьезные отличия. Среди новых систем лидирующими являются Arch, Mercurial и Bazaar-NG. Распределенная система управления версиями файлов Git, разработанная создателем Linux Линусом Торвальдсом (Linus Torvalds), сразу получила поддержку в сообществе продуктов с открытым исходным кодом и сейчас используется для управления разработкой исходного кода ядра Linux. Проверенная временем система Subversion (представитель предыдущего поколения) по-прежнему находит широкое применение и предлагает пользователям знаменитый графический интерфейс Windows.

Можно воспользоваться также и некоторыми коммерческими системами управления изменениями. С какой-нибудь одной из них вы, наверное, уже имели дело, если вам приходилось работать в отделе разработки, и вы пытались адаптировать ее к вашим административным данным. Однако с ними нужно быть очень осторожным; наш опыт подсказывает, что коммерческие системы обычно крайне сложны для использования в системном администрировании.

Самыми популярными системами в настоящее время являются Subversion и Git. Каждая из них хорошо зарекомендовала себя в области системного администрирования, но у системы Git есть преимущество, которое заключается в том, что она создает новые хранилища с помощью быстрой и простой операции. О других преимуществах системы Git см. веб-сайт Скотта Чакона (Scott Chacon) **whygitisbetterthanx.com**.

## Система Subversion

В модели Subversion центральный сервер или каталог действует как официальное хранилище проекта. По умолчанию сервер Subversion является модулем в веб-сервере Apache, что удобно для распределенной разработки программ, но не удобно для администрирования. К счастью, почитатели Subversion предлагают другой тип сервера в виде демона **svnserve**. Этот демон можно запускать из домашнего каталога, экспериментируя с системой Subversion, однако в производственных целях он должен иметь свою учетную запись пользователя и запускаться из **inetd**.

Настроить хранилище несложно. Например, ниже показан пример создания нового хранилища Subversion, называемого **admin**.

```
# cd /home/svn
# mkdir repositories
# cd repositories
# svnadmin create admin
# chmod 700 admin
```

[9] Предположим, например, что системные администраторы Алиса и Боб оба редактируют один и тот же файл, и каждый из них вносит в файл некоторые изменения. Алиса первая сохранила файл. Когда Боб сохранил свою копию файла, он перезаписал версию Алисы. Если Алиса закроет редактор, ее изменения будут полностью утрачены и не смогут быть восстановлены.

Если заглянуть в каталог **admin**, то в нем вы сможете найти хорошо организованную структуру хранилища, включая файл **README**. Конфигурационный файл **svnserve.conf** можно найти в подкаталоге **conf**. Этот файл сообщает демону сервера о том, как нужно предоставлять доступ к новому каталогу. Ниже показан пример конфигурации, подходящей для административных файлов.

```
[general]
anon-access = none
auth-access = write
password-db = passwd
realm = The Sysadmin Repository
```

Так как одной из целей, стоящих перед разработчиками Subversion, было создание условий для совместной работы пользователей из разных организаций, в этой системе есть модель управления доступом, которая отделена от модели в операционной системе. Файл **passwd** (в том же каталоге) содержит список пользователей и их пароли (в открытом тексте!). Использование открытого текста нельзя отнести к достоинствам; с другой стороны, смягчающим обстоятельством является то, что пароли никогда не передаются по сети. Кроме того, пароли никогда не вводятся пользователями по памяти, поэтому можно присваивать многосимвольные пароли, длина которых выбирается случайным образом. Как следствие, такие пароли должны быть безопасными. Взгляните на следующий пример.

```
[users]
tobi = lkadslfkjasdljkhe8938uhau7623rhkdfndf
evi = 09uqalkhlkasdgfprghkjhsdfjj83yyouhfuhe
fritz = kd939hjahkjaj3hkuyasdfaadfk3ijdkjhf
```

Естественно, разрешения на файл **passwd** нужно устанавливать отдельно.

Все что осталось — это запустить сервер в отдельном хранилище.

```
# svnserve --daemon --root /home/svn/repositories
```

Будучи непривилегированным пользователем, вы можете сейчас выписать архив **admin** из любого места в сети.

```
$ svn checkout --username tobi svn://server.atrust.com/admin checkout
Authentication realm: <svn://server.atrust.com:3690> The Sysadmin Repository
Password for 'tobi': <password>
```

Когда вы введете пароль первый раз, система Subversion создает его копию в каталоге **.subversion**, который она создает в вашем домашнем каталоге. Для того чтобы добавить или удалить файлы из локальной копии проекта, используйте команду **svn**.

```
$ cd checkout
$ vi foo.c
$ svn add foo.c
```

Когда все будет готово, нужно зафиксировать изменения в хранилище.

```
$ svn commit -m "Initial checkin; added foo.c"
```

Перечислять измененные файлы, которые вы хотите зафиксировать, не нужно, хотя при желании это можно сделать; команда **svn** разберется с этим сама. Если опустить опцию **-m**, команда **svn** запустит редактор, в котором вы сможете отредактировать сообщение о фиксации.

Для того чтобы получить самые последние обновления из хранилища, запустите **svn update** внутри проекта. Subversion выполняет операцию по объединению любых фай-

лов, которые были изменены как в вашей локальной копии проекта, так и в главном хранилище. Файлы с неразрешимыми конфликтами маркируются как "conflicted", и Subversion не разрешит регистрировать их до тех пор, пока вы не устраните проблемы и не сообщите системе, что конфликты были разрешены.

```
$ svn resolved foo.c
```

Если вы хотите узнать, кто производил изменения и какие строки в файле были изменены, спросите об этом систему Subversion.

```
$ svn blame bar.c
```

Эта команда печатает снабженную ссылками версию файла, которая показывает, когда и кем была изменена строка последний раз. (Оптимисты, а также те, кто терпим к чужим ошибкам, могут использовать синоним **svn pralse**.) Кроме того, можно без труда узнать о различиях относительно некоторой даты или версии. Например, если вы хотите узнать, что было изменено в файле **foo.c** начиная с 4 июля 2010 года, вам поможет следующая команда.

```
$ svn diff -r "{2010-07-04}" foo.c
```

Последнюю версию системы Subversion можно загрузить на сайте **subversion.tigris.org**. Стандартной документацией является книга *Version Control with Subversion*, опубликованная в издательстве O'Reilly. Полный текст этой книги доступен в интерактивном режиме на сайте **svnbook.red-bean.com**.

Subversion имеет также знаменитый интерфейс Windows, именуемый TortoiseSVN (зайдите на сайт **tortoisesvn.tigris.org**). Мы использовали его для управления исходными файлами для этой книги.

## Система Git

Характерной особенностью системы Git является то, что она не имеет центрального хранилища. Вместо регистрации конкретной версии файлов некоторого проекта вы просто копируете хранилище (включая всю его историю) и уносите его с собой, подобно тому как рак-отшельник перемещается со своим панцирем. Ваши обращения к хранилищу представляют собой локальные операции, поэтому они выполняются быстро, и вам не приходится беспокоиться о соединении с центральным сервером.

Git использует интеллектуальную систему сжатия данных, чтобы уменьшить стоимость хранения всей истории, и в большинстве случаев эта система достаточно эффективна. Зачастую рабочие требования на область памяти оказываются даже ниже, чем при использовании Subversion.

Git прекрасно подходит для разработчиков, поскольку они могут накапливать свой исходный код в ноутбуке и работать без подключения к сети, используя при этом все преимущества системы управления изменениями. Когда же подходит время совмещать работу многих разработчиков, их изменения могут быть объединены из разных копий хранилища любым способом, подходящим для вашей организации. Всегда можно вернуть две копии хранилища к их общему предшествующему состоянию, вне зависимости от того, сколько изменений и итераций было выполнено после разделения.

Смешанная стратегия копирования и ветвления, реализованная в Git, не очень-то соответствует контексту системного администрирования, но ее использование локального хранилища можно считать большим шагом вперед по отношению к системным администраторам (точнее сказать, большим шагом назад, но в хорошем смысле). Системы управления изменениями предыдущего поколения (например, RCS и CVS) хотя и ис-

пользовали локальные хранилища, но не могли координировать совместную деятельность, обрабатывать объединение изменений и независимую разработку. Сейчас мы прошли полный цикл и очутились в точке, в которой помещение файлов под "надзор" управления версиями является быстрой, простой, локальной операцией. Все современные Git-функции обеспечения совместной работы доступны для использования в соответствующих ситуациях.

Прежде чем начать работу с системой Git, укажите свое имя и адрес электронной почты.

```
$ git config --global user.name "John Q. Ulsah"
$ git config --global user.email "ulsah@book.admin.com"
```

Для того чтобы отслеживать обычные изменения, вносимые в файлы конфигурации, как правило, используются хранилища, которые никогда не дублируются, и, следовательно, никогда не возникает необходимость в их согласовании или объединении. Это соглашение делает использование системы Git очень простым, но поскольку действия будут совершаться от имени пользователя **root**, важно позаботиться, чтобы все потенциальные субъекты деяния указали свои имена и адреса электронной почты, как было показано выше. (Git использует ваши персональные данные для записей в журнале, даже когда вы запускаете эту систему через команду **sudo**.)

Для того чтобы создать хранилище, которое охватывает каталог **/etc**, вам следует выполнить эти команды.

```
$ cd /etc
$ sudo git init
Initialized empty Git repository in /etc/.git/
$ sudo git add .
$ sudo git commit -m "Initial commit"
Created initial commit ed25c29: Initial commit
2538 files changed, 259122 insertions(+), 0 deletions(-)
  create mode 100644 .java/.systemPrefs/.system.lock
  create mode 100644 .java/.systemPrefs/.systemRootModFile
...
```

В приведенной выше последовательности команда **git init** создает инфраструктуру хранилища в каталоге **/etc/.git**. Команда **git add .** помещает каталог **/etc** и все его содержимое в Git-список "стадийности", который представляет собой список файлов, подлежащих использованию в следующей операции **git commit**. Флаг **-m** команды **git commit** включает зарегистрированное сообщение в командной строке. Если вы оставите его, команда **git** откроет редактор, с помощью которого вы сможете построить это зарегистрированное сообщение.

Давайте сделаем изменение и внесем его в хранилище.

```
$ sudo vi mdadm/mdadm.conf
$ sudo git commit mdadm/mdadm.conf -m "Added spare to svr4west array"
Created commit 901bd39: Added spare to svr4west array
  1 files changed, 1 insertions(+), 0 deletions(-)
```

Присваивание имен модифицированным файлам в строке команды **git commit** игнорирует обычное использование в Git области "стадийности" и создает модификацию, которая включает только изменения, внесенные в названые файлы. Существующая область "стадийности" остается неизменной, и Git игнорирует все остальные файлы, которые, возможно, были модифицированы.

Если некоторое изменение коснулось нескольких файлов, у вас есть два варианта действий. Если вы точно знаете, какие файлы были изменены, можете всегда перечислить их в командной строке, как показано выше. Если вам лень это делать, можете выполнить команду **git commit -a**, чтобы заставить Git добавить все модифицированные файлы в области "стадийности" до выполнения операции. Однако последний вариант имеет два подводных камня.

Во-первых, могут существовать модифицированные файлы, которые не связаны с вашими изменениями. Например, файл **/etc/mtab** в системах Linux поддерживается системой, поэтому он может изменяться даже при отсутствии конфигурационных изменений. Позволяя этому файлу участвовать в **commit**-наборах, вы "нарываетесь" на проблемы в будущем, поскольку он (файл) не является реальной составляющей ваших текущих изменений. Если вам позже придется возвращать свои изменения к предшествующему состоянию, будет ошибкой "откатывать" назад и файл **mtab**.

Во-вторых, команда **git commit -a** проверяет на наличие изменений только те файлы, которые находятся "во власти" системы управления изменениями в данный момент. Новые файлы такой "заботой" охвачены не будут.

Для того чтобы избежать этих "неувязок", выполните команду **git status** и оцените ситуацию вручную. Эта команда информирует вас о новых, модифицированных и "поэтапных" файлах одновременно. Например, предположим, что мы отредактировали файл **/etc/mdadm/mdadm.conf** (как в предыдущем примере), а также инсталлировали новый системный демон **foobard**, конфигурация которого содержится в файле **/etc/foobard/foobard.conf**. Тогда Git представит следующую информацию[10].

```
$ sudo git status
# On branch master
# Changed but not updated:
# (use "git add <file>..." to update what will be committed)
#
#       modified: mdadm/mdadm.conf
#       modified: mtab
#       modified: passwd
#Untracked files:
#  (use "git add <file>..." to include in what will be committed)
#
#       foobard/
no changes added to commit (use "git add" and/or "git commit -a")
```

Имя файла **foobard.conf** здесь не указано, поскольку Git не "видит" содержимое каталога **foobard**, который содержит его. Мы видим, что оба файла **mtab** и **passwd** имеют неожиданные изменения. Изменения в файле **mtab** определенно фиктивны, но изменения в файл **passwd** могли быть внесены. Возможно, сценарий инсталляции для **foobard** создал специальную учетную запись либо кто-нибудь отредактировал файл **passwd** и забыл внести изменения в хранилище.

Для того чтобы прояснить ситуацию, можно выполнить команду **git diff passwd** и узнать о реальных изменениях, внесенных в файл **passwd**. Предположим, что в данном случае изменения в файле **passwd** не были связаны с нашими недавними действиями. Следовательно, нам имеет смысл зарегистрировать эти изменения отдельно от только что сделанных.

[10] Даже если эта команда не вносит никаких изменений, тем не менее она должна быть выполнена от имени пользователя root, поскольку она создает заблокированные файлы в каталоге .git.

```
$ sudo git commit passwd -m "Checking in existing passwd changes"
Created commit 6f7853c: Checking in existing passwd changes
 1 files changed, 1 insertions(+), 0 deletions(-)
```

Мы можем заставить Git игнорировать файл **mtab** раз и навсегда, но для этого необходимо выполнить два действия. Во-первых, "удалим" файл **mtab** из текущего образа хранилища.

```
$ sudo git rm --cached mtab
rm 'mtab'
```

Опция **--cached** предотвращает реальное удаление файла **mtab**, поэтому не выбрасывайте ее! По существу, мы переводим операцию виртуального удаления файла в Git-область "стадийности". Git будет вести себя так, как если бы мы удалили файл с помощью команды **rm**.

Во-вторых, удаление файла **mtab** из Git-вселенной предполагает его добавление в Git-список файлов, которые будут игнорироваться в будущем. Это реализуется посредством создания или редактирования файла **.gitignore**.

```
$ sudo sh -c "echo mtab >> .gitignore"[11]
```

Наконец, мы должны выполнить все оставшиеся изменения.

```
$ sudo git add .
$ sudo git commit -m "Installed foobard; added RAID spare"
Created commit 32978e6: Installed foobard; added RAID spare
 4 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
 create mode 100644 foobard/foobard.conf
 delete mode 100644 mtab
```

Обратите внимание на то, что файл **.gitignore** сам становится частью управляемого набора файлов. Было бы неплохо файлы, которые уже находятся под управлением этой системы, повторно добавить в такой набор, поэтому команда **git add .** — это простой способ выразить следующее желание: "Хочу, чтобы новый образ хранилища выглядел так же, как текущий каталог". В данной ситуации вы могли бы просто выполнить команду **git commit -a**, поскольку она не затронет ни файл **foobard.conf**, ни файл **.gitignore** (эти файлы "слишком свежие" для управляющего Git-механизма и должны быть добавлены в круг управления в явном виде).

Стараясь заставить вас подумать, что система Git управляет не только содержимым файлов, но и правами доступа к ним, при добавлении новых файлов в хранилище она отображает режимные коды файлов. Но это не так: Git не отслеживает ни режимы, ни владельцев, ни время модификации. Если вы используете Git для возврата изменений в системных файлах, дважды убедитесь в том, что их атрибуты остались правильными. Отсюда делаем вывод, что вы не можете рассчитывать на использование Git, если предполагаете "с нуля" восстанавливать сложные файловые иерархии в ситуациях, когда для вас важна информация о принадлежности ресурсов и правах доступа на них.

Использовать Git для базового управления изменениями ненамного труднее, чем создавать резервные копии вручную. Однако трудно требовать от всей вашей команды администраторов быстрого обслуживания системы и при этом пребывать в уверенности, что она (система) используется надлежащим образом.

[11] Вызов команды `sh` в явном виде заставляет выполнить оператор перенаправления в контексте корневой оболочки (root shell). Если бы мы просто ввели команду `echo mtab >> .gitignore`, интерпретатор попытался бы открыть файл `.gitignore` до выполнения команды `sudo`.

# 12.10. Локализация и конфигурирование программного обеспечения

Адаптация компьютеров к вашей локальной среде является одним из основных заданий системного администрирования: вы должны сообщить системе обо всех принтерах, доступных в сети, запустить специальный демон лицензирования, добавить задачу `cron`, очищающую каталог `/scratch` один раз в неделю, интегрировать поддержку для специального сканера и т.п. Для выполнения этих задач нужен структурированный и воспроизводимый способ.

Вам нужно помнить о следующем.

- У пользователей нет привилегий системного администратора. Необходимость в них в ходе выполнения обычных операций вызывает подозрение и наверняка свидетельствует о том, что кто-то имеет злой умысел.
- У пользователей нет намерений причинить вред системе. Вы должны организовать защиту системы таким образом, чтобы она была защищена от неумышленных ошибок и не передавала "первому встречному" привилегии системного администратора.
- Прежде чем наказывать пользователей, изредка делающих что-то не так, нужно провести с ними беседу. На неумелые действия администратора пользователи зачастую реагируют неадекватно. Следовательно, если ваши действия мешают пользователям работать, то это свидетельствует о проблемах в архитектуре.
- Старайтесь ориентироваться на заказчика. Беседуйте с пользователями и постарайтесь узнать, какие задачи для них являются сложными. Попытайтесь сделать эти задачи гораздо проще.
- Ваши предпочтения — только ваши, и ничьи более. Пользователи должны иметь собственные предпочтения. По мере возможности такие варианты нужно предлагать всегда и везде.
- Если ваши решения, принимаемые в административных целях, влияют на способность пользователей обучаться системе, постарайтесь их изменить. Расскажите о них пользователям.
- Постоянно обновляйте вашу локальную документацию и делайте ее легкодоступной. Более подробно об этом можно прочитать в разделе 32.5.

## Организация локализации

Когда в организации тысяча компьютеров с различными конфигурациями, администратор вынужден тратить большую часть времени на выяснение причин того, почему такая-то проблема возникла только на этом компьютере. Думаете, решением проблемы будет унификация всех конфигураций? Не спешите с выводами! Ограничения, существующие в реальном мире, а также различные нужды ваших пользователей делают это невозможным.

Между администрированием "многочисленных" и "бесчисленных" конфигураций есть разница. Важно разделить вашу установку на управляемые части. Вы увидите, что одни части локализации применяются ко всем управляемым компьютерам, другие — только к некоторым из них, а третьи — только к отдельным компьютерам.

Помимо выполнения инсталляций с "чистого листа", вам придется также постоянно загружать и инсталлировать обновления. Следует иметь в виду, что на разных компьютерах установлены различные сроки действия и предъявляются разные требования к стабильности и периоду работоспособности.

Предусмотрительный системный администратор никогда не станет развертывать новые выпуски программного обеспечения на всех компьютерах сразу. Наоборот, нужно делать это с учетом нужд каждой группы и потратить некоторое время на выявление проблем на самых ранних этапах, чтобы не допустить серьезного повреждения системы. Важные серверы нельзя обновлять до тех пор, пока не будет уверенности в надежности производимых изменений, и "избегайте пятниц", если вы не готовы проводить долгие выходные перед монитором[12].

Однако в процессе проектирования системы локализации следует убедиться в том, что все исходные данные хранятся в системе управления изменениями. Так вы сможете всегда знать, какие изменения были тщательно проверены и готовы к развертыванию. Кроме того, это позволит вам идентифицировать пользователя, являющегося автором проблематичных изменений. Чем больше человек вовлечено в этот процесс, тем более важным является последнее утверждение.

Свои преимущества есть и у разделения базового выпуска ОС и выпуска локализации. В зависимости от того, какую степень стабильности необходимо достичь в вашей среде, вы можете использовать второстепенные локальные выпуски только для того, чтобы устранить ошибки. Наш опыт подсказывает, что добавление новых функций небольшими порциями приводит к более стабильному выполнению операций, чем при постановке изменений в очередь на следующие крупные выпуски, что сразу же снизит качество обслуживания.

Часто имеет смысл определять максимальное количество "выпусков", с которыми вы хотите работать в любой момент времени. Некоторые администраторы придерживаются мнения, что если программное обеспечение работает надежно, то ничего изменять в нем не нужно. Они полагают, что не имеющая под собой почвы модернизация систем стоит времени и денег и что "передовой выпуск" слишком часто означает "выпуск с множеством ошибок". Те же, кто вооружается этими принципами на практике, должны быть готовы к тому, что активные выпуски придется коллекционировать во вместительном каталоге.

В отличие от них, активные пользователи ссылаются на присущую выпускам сложность и трудность в осмыслении (не говоря уже об управлении) случайной коллекции выпусков, вышедших много лет тому назад. Их козырем являются "заплаты" системы защиты, которые должны обычно применяться универсальным образом и в строгом порядке. Инсталлировать "заплаты" в устаревших версиях операционной системы часто бывает просто невозможно, поэтому администраторы сталкиваются с выбором: не инсталлировать обновления на некоторые компьютеры или полностью переходить на новые внутренние выпуски. Это не сулит ничего хорошего.

Ни одна из этих перспектив не является доказуемо верной, хотя мы стараемся придерживаться мнения, что лучше использовать ограниченное количество выпусков. Лучше выполнять модернизацию по вашему собственному плану, чем руководствоваться чьим-то планом.

[12] Вероятным исключением из этого правила являются "заплаты" системы защиты. Нужно стараться закрывать все бреши в защите сразу же после их выявления. С другой стороны, "заплаты" системы защиты могут сами вызвать ошибки в работе системы.

## Тестирование

Важно протестировать изменения, пока их еще можно отменить. По меньшей мере, это означает тестирование собственных настроек. Но в той же степени это касается и системного программного обеспечения. Один известный поставщик UNIX однажды выпустил "заплату", которая, будучи примененной определенным образом, выполняла команду `rm -rf /`. Представьте последствия установки такой "заплаты" во всей организации без предварительного тестирования.

Тестирование особенно полезно, когда применяются средства, способные автоматически устанавливать "заплаты" (как большинство систем управления пакетами, рассмотренных в этой главе). Не стоит подключать компьютеры, ответственные за выполнение ключевых задач, к службе обновления, спонсируемой поставщиком. Постарайтесь подключить к этой службе простой компьютер и уже с него распространяйте изменения на остальные компьютеры в вашей организации. Не допускайте производство обновлений во время проверки, иначе они могут раньше времени проникнуть в вашу производственную систему.

🕮 О выявлении проблем речь пойдет в разделе 32.2.

Если вы предвидите, что заранее запланированное обновление или изменение может вызвать у пользователей некоторые проблемы, уведомьте их о ваших планах и позвольте им связаться с вами, если у них возникнут какие-либо трудности. Позаботьтесь о том, чтобы пользователи имели возможность оперативно сообщать о замеченных ошибках.

В транснациональной компании можно подключить к тестированию иностранные офисы. Участие пользователей из других стран особенно важно в случае многоязыковых систем. Если никто в американском офисе не говорит по-японски, необходимо связаться с офисом в Токио и попросить их проверить все, что касается поддержки раскладки кандзи (kanji). Вообще, на удивление, большое число системных параметров зависит от региональных особенностей. К примеру, при тестировании систем печати имеет смысл проверять, что произойдет, если задать иной размер бумаги. Ведь в разных странах может использоваться офисная бумага разного формата.

## Локальная компиляция

Давным-давно, когда существовало множество аппаратных платформ UNIX, программы обычно распространялись в виде архивов исходных текстов. Как правило, это были файлы с расширением `.tar.Z`, которые нужно было разархивировать с помощью команды `uncompress`, а потом скомпилировать. Затем программу требовалось инсталлировать в соответствующий каталог, например `/usr/local`. Сегодня, в эпоху систем управления пакетами, все меньше программ инсталлируется подобным образом. Задачи администраторов тоже сильно упростились, так как в пакетах указано, куда их инсталлировать.

Несмотря на простоту управления пакетами, многие предпочитают самостоятельно компилировать программы[13]. Это позволяет лучше контролировать параметры компиляции. Фанатики могут даже изучить исходный код, чтобы быть уверенными в его безопасности. Кому-то это кажется важным, но мы считаем, что если у вас нет времени и достаточного опыта для анализа каждой из 20 000 строк приложения, то такая мера предосторожности приносит минимальную пользу.

[13] Для этой цели можно воспользоваться дистрибутивом Gentoo Linux, который можно перекомпилировать с "чистого листа".

Далеко не все программы выпускаются в виде пакетов для всех дистрибутивов Linux и UNIX, поэтому вам наверняка придется столкнуться с программами, которые необходимо скомпилировать и инсталлировать вручную, особенно если речь идет о 32-разрядных компьютерах на платформе Intel. В организации, разрабатывающей собственное программное обеспечение, необходимо также решить, куда его инсталлировать.

Исторически сложилось так, что локальные программы чаще всего помещают в каталог **/usr/local**, и это соглашение до сих пор широко распространено. Применяемый в UNIX/Linux стандарт FHS (Filesystem Hierarchy Standard — стандарт иерархии файловой системы) определяет, что после начальной инсталляции операционной системы каталог **/usr/local** должен присутствовать и быть пустым.

Каталог **/usr/local** хоть и традиционный, но управлять им не всегда легко. Его способ организации (исполняемые файлы помещаются в подкаталог **/usr/local/bin**, man-страницы — в подкаталог **/usr/local/man** и так далее) таков, что возникает ряд проблем: трудно инсталлировать несколько версий одной и той же программы, подкаталоги сильно разрастаются, сложно поддерживать несколько платформ и т.д.

## Распространение локализаций

Система локализации в вашей организации должна справляться как с исходной инсталляцией, так и с последующими обновлениями. Особую сложность могут представлять обновления. Основным в этом вопросе является достижение эффективности, поскольку нет смысла повторять всю локализацию с самого начала, чтобы обновить разрешения для одного файла. Несмотря на то что этот процесс производится автоматически, модель с повторной сборкой с "чистого листа" делает обновление дорогим и длительным во времени процессом.

Простой и масштабируемый способ организации локализации заключается в поддержке файлов в структуре дерева, которая напоминает файловую систему производственного компьютера. Специальный инсталляционный сценарий может скопировать дерево на искомый компьютер и выполнить все необходимые дополнительные действия по редактированию.

Такой тип установки имеет несколько преимуществ. Вы можете поддерживать столько деревьев локализации, сколько необходимо для реализации вашей локальной административной схемы. Некоторые деревья будут альтернативными, причем каждый компьютер получит только один доступный вариант. Остальные деревья будут использоваться в качестве оверлеев — их можно будет копировать поверх уже существующих деревьев. Деревья локализации могут перезаписывать файлы, если это будет необходимо, или могут полностью заменить их структуру. Каждое дерево, которое потенциально инсталлируется независимым образом, должно быть представлено в отдельном проекте управления изменениями.

Подход с использованием деревьев-оверлеев позволяет внести гибкость в реализацию. Если вы используете систему упаковки для распространения ваших локальных настроек, то оверлеи можно просто реализовать в независимых пакетах. Можно добавить в пакет соответствующие сценарии настройки и задать их таким образом, чтобы они запускались как часть процесса инсталляции.

Еще одна хорошая идея с реализацией заключается в использовании утилиты **rsync**, которая согласовывает искомые компьютеры с деревьями-оверлеями. Утилита **rsync** копирует только те файлы, которые являются устаревшими, поэтому она может быть эффективной для распространения инкрементных изменений. Такого поведения трудно добиться с помощью одной лишь системы упаковки. Утилита **rsync** описана в разделе 19.2.

# 12.11. Использование средств управления конфигурацией

Системы локализации обычно являются "доморощенными". Частично это объясняется тем, что все организации отличаются друг от друга и каждая имеет свои особенности. Однако большой вклад вносит также синдром "придуманного не здесь" (not invented here). Возможно, отсутствие господствующего инструмента с открытым исходным кодом для управления конфигурацией заставляет нас смотреть на эту проблему как на такую, которая выходит за рамки стандартизированных средств.

Тем не менее такие средства существуют и стоят того, чтобы вы обратили на них внимание. В следующих разделах будут рассмотрены распространенные системы в порядке их популярности и схожести.

## Утилита cfengine: компьютерная иммунная система

Одной из наиболее известных утилит локализации является утилита **cfengine**, разработанная Марком Бургессом (Mark Burgess). Она была представлена как "компьютерная иммунная система", выполняющая свои операции на основе модели, в соответствии с которой выбирается способ конфигурирования системы. В случае расхождений между моделью и реальностью, **cfengine** выполняет определенные действия, направленные на то, чтобы привести систему в соответствие. Благодаря этой модели, утилита **cfengine** действительно является полезной для постоянной поддержки конфигурации.

Утилита **cfengine** умеет создавать резервные копии файлов, которые она модифицирует, и может хранить специальный журнал своих изменений. Ее можно запускать в режиме "бездействие", в котором она описывает изменения, которые она может произвести, в действительности не реализуя их.

С помощью специального языка утилиты **cfengine** можно описать способ конфигурирования компьютеров. Это можно сделать, например, так: "Файл **xyz** должен существовать в каталоге **/etc**, иметь права доступа 644 и принадлежать суперпользователю". Можно написать правила в зависимости от содержимого отдельных файлов. Например, можно определить, что файл **/etc/hosts** должен содержать строку "router 192.168.0.1". Утилита **cfengine** добавит эту строку, если она будет отсутствовать.

Язык конфигурации утилиты **cfengine** позволяет формулировать индивидуальные правила в зависимости от таких факторов, как имя компьютера, тип операционной системы или маска подсети. Эта особенность облегчает написание одного конфигурационного файла, который удовлетворит нужды всех компьютеров, находящихся в зоне вашей административной ответственности.

Ниже показан простой пример из мира UNIX. В нем проверяется, является ли **/bin** символической ссылкой на **/usr/bin** в компьютерах Sun, производится дополнительная проверка ссылок в старых "коробочных" версиях OSF и удаление из каталога **/var/scratch** любых файлов, "возраст" которых превышает семь дней.

```
control:
actionsequence = ( links tidy )
links:
   sun4::
   /bin -> /usr/bin
   # другие ссылки
osf::
```

```
  # некоторые специальные osf-ссылки
tidy:
  /var/scratch pattern=* age=7 recurse=inf
```

На домашней странице утилиты cfengine (**cfengine.org**) можно найти ее описание.

## LCFG: крупномасштабная система конфигурирования

LCFG первоначально была разработана Полом Андерсоном (Paul Anderson) в университете Эдинбурга (Edinburgh University) в 1993 году. Последний вариант этой системы известен под именем LCFG(ng); она стала популярна среди многочисленных пользователей за пределами этого университета. LCFG первоначально предназначалась для управления большими установками Solaris или Linux. Веб-сайт этой системы находится по адресу **lcfg.org**.

Как и **cfengine**, LCFG имеет специализированный язык конфигураций. Конфигурации всех управляемых компьютеров хранятся на центральном сервере в виде подборки главных конфигурационных файлов. На основе этих файлов LCFG генерирует специализированные XML-файлы, описывающие конфигурацию каждого управляемого компьютера. Демон центрального сервера наблюдает за изменениями, происходящими в главных конфигурационных файлах, и генерирует необходимые XML-файлы.

XML-файлы публикуются на внутреннем веб-сервере, из которого клиенты могут извлекать свои собственные конфигурации. Клиенты используют разнообразные сценарии компонентов для своего конфигурирования в соответствии с XML-образцами.

## Template Tree 2: помощник cfengine

Система Template Tree 2 была создана в Швейцарском федеральном технологическом институте (Swiss Federal Institute of Technology — ETH) Тобиасом Оэтикером (Tobias Oetiker). Она построена на основе компонентов и управляется центральной конфигурацией. Ее изюминка заключается в возможности использовать двухуровневый подход в определении конфигурации организации. Кроме того, она может работать с перемещенными корневыми каталогами на бездисковых компьютерах.

На нижнем уровне система состоит из множества "пакетов с функциональными особенностями". Такой пакет представляет собой коллекцию файлов, сопровождаемых **META**-файлом, который описывает, как эти файлы должны быть инсталлированы в искомой системе. В качестве функциональной особенности может быть что угодно, начиная от сетевой конфигурации и заканчивая последней версией OpenSSH. Функциональные особенности могут предлагать конфигурируемые параметры, которые можно задавать в главном конфигурационном файле.

На верхнем уровне конфигурации представлен главный конфигурационный файл организации, в котором функции собраны вместе и который связывает их с компьютерами или группами компьютеров. На этом уровне вы должны определить значения для свободных конфигурационных параметров, предлагаемых каждой функцией. Например, одним из параметров для почтового сервера может быть имя почтового домена.

Template Tree 2 комбинирует информацию из главного конфигурационного файла и **META**-файлов отдельных функций для генерирования конфигурационного файла **cfengine** для всей организации. Так как каждая функция должна содержать документацию о ее назначении и использовании, Template Tree 2 может также генерировать составную документацию.

## DMTF/CIM: общая информационная модель

Группа Distributed Management Task Force (DMTF — группа распределенного управления), насчитывающая, по их собственному признанию, более 3000 активных членов, с 1992 года работает над общей информационной моделью (Common Information Model — CIM), пытаясь разработать стандарты для объектно-ориентированной, межплатформенной системы управления.

Как утверждает группа DMTF, CIM — это "схема управления..., предназначенная для установления общей концептуальной структуры на уровне фундаментальной топологии по отношению как к классификации и ассоциации, так и к базовому набору классов, ориентированных на установление общей структуры для описания управляемой среды". Или что-то в этом роде.

Членами DMTF являются все основные производители, начиная от Microsoft и заканчивая Sun. К сожалению, предложенные ими стандарты показывают, что эти фирмы в совершенстве овладели искусством сбивать с толку и вводить модные словечки. Фирмам, принимающим участие в этом проекте и имеющим солидную репутацию, не к лицу демонстрировать свою привычку стандартизировать все подряд, без разбору. Стандарты сосредоточены вокруг XML и объектной ориентации. Однако нам пока что приходится видеть основанный на них удобный продукт.

Во всем этом безобразии есть один светлый момент: благодаря усилиям DMTF поставщики вынуждены предоставлять для своих систем, основанных на открытом стандарте, интерфейсы конфигурации, доступные программным образом. Для сред UNIX и Linux в этом нет ничего нового, однако и DMTF не является детищем UNIX. Членами DMTF являются Cisco, Microsoft, Symantec и многие другие компании, имеющие небогатый опыт в обеспечении удобных способов написания сценариев для своих систем. Если бы эти продукты получили API-конфигурации, то это было бы здорово, даже если бы реализация была несовершенной.

# 12.12. Организация совместного использования программ через NFS

Где следует инсталлировать дополнительное программное обеспечение: на отдельных клиентах или на центральном файловом сервере, на котором его можно совместно использовать через NFS? Если речь идет о Linux, то ответ будет такой: "на клиентах". Однако NFS позволяет выполнять обновления гораздо быстрее (обновить десяток серверов NFS будет гораздо быстрее и надежнее, чем 1000 клиентов) и экономить место на дисках клиентов (это не важно для тех, кто живет в мире терабайтовых дисков).

На самом деле вопрос сводится к сравнению управляемости и надежности. Доступ на основе сетевой файловой системы с каждым днем становится более централизованным и простым в управлении; с его помощью варианты устранения ошибок и новые пакеты становятся сразу же доступными на всех клиентах. Однако работа по сети может вестись медленнее, чем при доступе к локальному диску. Кроме того, в модели с сетевым сервером появляется зависимость от сети и файлового сервера, не только потому, что она добавляет потенциальные места сбоев, но и потому также, что она требует, чтобы клиенты и серверы согласовывали свои действия относительно таких вопросов, как доступ к общим библиотекам и версии этих библиотек. Недостатком является то, что эти библиотеки программ NFS представляют собой продвинутую технологию администриро-

вания и должны быть опробованы только в тех средах, в которых координация действий производится из центра.

Вообще, в сетях, состоящих из однородных систем, большая часть преимуществ достигается за счет совместно используемых хранилищ программного обеспечения. Если ваша организация стандартизировала операционную систему и эта система имеет удобные функциональные средства для управления пакетами, то вам, наверное, лучше всего будет отказаться от использования родной системы.

## Пространства имен пакетов

В традиционной системе UNIX содержимое новых пакетов распределяется по множеству каталогов. Библиотеки поступают в каталог **/usr/lib**, бинарные файлы — в каталог **/usr/bin**, документация — в каталог **/usr/share/docs** и т.д. Linux наследует эти свойства в той или иной мере, хотя стандарт Filesystem Hierarchy Standard (Стандарт иерархии файловой системы) помогает сделать размещение предсказуемым. (Подробные сведения о FHS можно узнать на сайте **pathname.com/fhs**.)

Преимущество этого соглашения состоит в том, что файлы оказываются в хорошо знакомых местах. Например, если переменная окружения **PATH** указывает на каталог **/usr/bin** и другие стандартные бинарные библиотеки, то новые программы будут доступны сразу же после инсталляции.

Недостатком является то, что источники файлов должны быть явным образом отслежены (средствами системы управления пакетами) и что разрозненные файлы трудно использовать совместно по сети. К счастью, системные администраторы могут выйти из затруднения благодаря пространствам имен пакетов.

Суть этой схемы заключается в том, чтобы инсталлировать каждый пакет в его собственный отдельный корневой каталог. Например, вы можете инсталлировать **gimp** в каталог **/tools/graphics/gimp**, а бинарные файлы — в каталог **/tools/graphics/gimp/bin/gimp**. Вы можете создать заново единый бинарный каталог для вашей коллекции утилит, поместив символические ссылки в каталог, такой как **/tools/bin**, например.

```
/tools/bin/gimp -> /tools/graphics/gimp/bin/gimp
```

Пользователи могут добавить каталог **/tools/bin** в свои переменные **PATH**, чтобы иметь возможность пользоваться всеми общими утилитами.

Определять структуру каталога **/tools** можно разными способами. Иерархический подход (например, **/tools/graphics**, **/tools/editors** и так далее) облегчает обзор и повышает производительность. В соглашение о назначении имен можно включить версию программы, архитектуру оборудования, операционную систему или инициалы ответственных пользователей, чтобы одним набором утилит могли пользоваться многие типы клиентов. Например, пользователи Solaris могут включить в свои переменные **PATH** ссылку **/tools/sun4/bin**, а пользователи Ubuntu — ссылку **/tools/ubuntu/bin**.

Когда вы инсталлируете новую версию основной утилиты, то старые версии желательно хранить как можно дольше, особенно если пользователи подолгу работают в проектах, в которых используется данная утилита. В идеале, новые версии утилит должны быть совместимыми со старыми файлами данных и программным обеспечением, однако на практике иногда встречается обратное. Неплохо заставить пользователей разобраться с конфигурацией и получить доступ к старой версии пакета, но неприятно просто прервать их работу и вынудить иметь дело с последствиями.

## Управление зависимостями

Некоторые пакеты зависят от библиотек или других программных пакетов. Когда вы инсталлируете программное обеспечение локальным образом посредством систем управления пакетами, то для решения этих проблем вы получите всестороннюю помощь. Однако когда вы будете создавать собственное сетевое хранилище программ для своей организации, вам придется решать эти проблемы самостоятельно.

Если вы управляете библиотеками таким же образом, как и приложениями, можете компилировать утилиты для использования библиотек из общего каталога **/tools**. Так вы сможете хранить активными одновременно множество версий одной библиотеки. Поскольку зависимые приложения связаны со специфическими версиями библиотеки, установка будет работать стабильно даже после выпуска новых версий библиотеки. Недостатком является то, что этот тип установки может быть довольно сложным, чтобы его использовать и обслуживать в течение продолжительного периода времени.

Старайтесь не создавать ссылки на локальный каталог **/tools/lib**, содержащий ссылки с обобщенными именами на общие библиотеки. Если вы измените ссылки, у вас могут возникнуть неожиданные проблемы, трудно поддающиеся диагностике. Системы с совместно используемыми библиотеками призваны избавить администраторов от некоторых возможных трудностей, однако в сложных установках лучше не рисковать.

Действия, необходимые для того, чтобы компоновщик использовал конкретную версию совместно используемой библиотеки, могут отличаться в разных системах. В Linux вы можете задать переменную окружения **LD_LIBRARY_PATH** или использовать опцию **-R** компоновщика.

## Сценарии упаковщика

К сожалению, совместимость на уровне библиотек — это полдела. То обстоятельство, что утилиты вызывают одна другую напрямую, порождает другую возможность возникновения конфликтов. Например, предположим, что утилита **foo** часто использует утилиту **bar**. После обновления версии **bar**, используемой по умолчанию, может оказаться, что утилита **foo** неожиданно перестала работать. В этом случае вы можете прийти к выводу, что **foo** зависит от некоторого поведенческого аспекта **bar**, который уже не поддерживается (или, по крайней мере, не поддерживается по умолчанию).

Если ваше хранилище программного обеспечения поддерживает множество версий (например, **/tools/util/bar-1.0** и **/tools/util/bar-2.0**), вы можете устранить эту проблему, переместив исходную версию утилиты **foo** в **foo.real** и заменив ее небольшим сценарием упаковщика.

```
#!/bin/sh
# make sure the program finds any files co-packaged with it
# first even if it does not use an explicit path.
PATH=/tools/util/bar-1.0/bin:$PATH
export PATH
exec /tools/util/foo-1.0/bin/foo.real "$@"
```

Теперь утилита **foo** будет запускаться с помощью специальной переменной окружения **PATH**, и она будет вызывать старую версию утилиты **bar**, а не новую.

Упаковщики представляют собой превосходные инструменты, которые могут не только справляться с зависимостями между пакетами, но и решать такие задачи, как защита, зависимость от архитектуры или операционной системы и отслеживание использования. В некоторых организациях упаковывают все совместно используемые библиотеки.

## 12.13. Рекомендуемая литература


- *Intel Corporation and SystemSoft. Preboot Execution Environment (PXE) Specification, v. 2.1*, 1999. `pix.net/software/pxeboot/archive/pxespec.pdf`
- *PXELinux Questions.* `syslinux.zytor.com/wiki/index.php/PXELINUX`
- Rodin, Josip. *Debian New Maintainers' Guide.* `debian.org/doc/maint-guide`. Этот документ содержит хорошую информацию о пакетах формата `.deb`.
- Silva, Gustavo Noronha. *APT HOWTO.* `debian.org/doc/manuals/apt-howto`
- Hohndel, Dirk and Fabian Hershell. *Automated Installation of Linux Systems Using YaST.* `usenix.org/events/lisa99/full_papers/hohndel/hohndel_html`
- Stuckelberg, Marc Vuileumier and David Clerc. *Linux Remote-Boot mini-HOWTO: Configuring Remote-Boot Workstations with Linux, DOS, Windows 95/98 and Windows NT*, 1999. `tldp.org/HOWTO/Remote-Boot.html`
- *The Red Hat Enterprise Linux System Administration Guide.* `redhat.com/docs`
- Wachsmann, Alf. *How to Install Red Hat Linux via PXE and Kickstart.* `stanford.edu/~alfw/PXE-Kickstart/PXE-Kickstart.html`
- Burgess, Mark. *"Cfengine: A Site Configuration Engine." USENIX Computing Systems, vol. 8, No 3*, 1995. `cfengine.org`
- *HP Ignite-UX Administration Guide.* `docs.hp.com/en/5992-5309/5992-5309.pdf`
- *NIM from A to Z.* `www.redbooks.ibm.com/redbooks/pdfs/sg247296.pdf`. Это полное руководство по использованию сетевого менеджера инсталляции (Network Installation Manager).
- *Solaris Advanced Installation Guide.* `docs.sun.com/app/docs/doc/802-5740`


## 12.14. Упражнения

12.1. Опишите различия между программами Kickstart, JumpStart и Ignite-UX. Назовите преимущества и недостатки каждой из них.

12.2. Инсталлируйте Subversion из сайта `subversion.tigris.org`. Установите `snvserve` и создайте хранилище. Как можно открыть хранилище, чтобы оно использовалось из любого места в сети при поддержке надлежащего уровня безопасности?

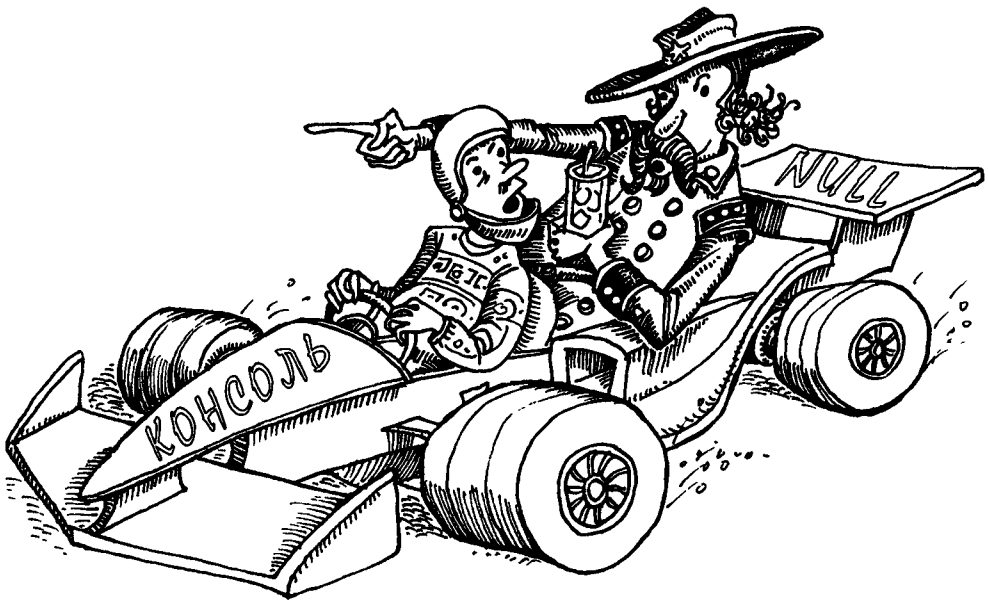
12.3. Выясните, как организовано хранение локального программного обеспечения в вашей организации. Легко ли масштабируется такая система? Удобно ли с ней работать? Аргументируйте свои ответы.

12.4. Назовите самые важные особенности системы управления конфигурацией. Что вы можете сказать о защите распределения конфигурационных файлов по сети?

12.5. ★★ Сконфигурируйте какой-нибудь сетевой инсталлятор по своему усмотрению и установите новую систему с локального сервера. Перечислите действия, которые пришлось предпринять для этого. Какие возникли трудности? Какие слабые места с точки зрения масштабируемости были вами замечены при работе с выбранным инсталлятором?

# Глава 13

# Драйверы и ядро




Ядро — это часть операционной системы, которая организует доступ к аппаратным средствам через абстрактный высокоуровневый программный интерфейс. Ядро отвечает за реализацию многих концепций, которые пользователями и программами пользовательского уровня принимаются как нечто само собой разумеющееся. В частности, на базе низкоуровневых аппаратных возможностей ядро реализует следующие элементы операционной системы Linux:

- процессы (защита адресных пространств, разделение времени и ресурсов процессора);
- сигналы и семафоры;
- виртуальную память (подкачка, страничный обмен, отображение виртуальных адресов в физической памяти);
- файловую систему (файлы, каталоги, пространство имен);
- общий ввод-вывод (специализированное оборудование, клавиатура, мышь, USB);
- межзадачное взаимодействие (каналы и сетевые соединения).

Ядро содержит драйверы устройств, которые организуют взаимодействие с отдельными элементами аппаратного уровня; остальная часть ядра, в основном, не зависит от внешних устройств. Взаимосвязь ядра и драйверов устройств аналогична связи между ядром и процессами пользовательского уровня. Когда процесс просит ядро "прочесть первые 64 байт файла **`/etc/passwd`**", оно (ядро или, вернее, драйвер файловой системы) транслирует эту просьбу в команду драйвера устройства, например "выбрать блок 3348 из устройства 3". Драйвер представляет эту команду в виде последовательности двоичных кодов, которые записываются в управляющие регистры устройства.

Ядро написано преимущественно на языке C, хотя в некоторых случаях используется язык ассемблера, что позволяет вызывать специализированные функции устройств, недоступные через обычные директивы компилятора.

Одно из преимуществ Linux и других сред с открытым исходным кодом заключается в том, что доступность исходного кода ядра позволяет легко, если это слово здесь вообще применимо, писать собственные драйверы устройств и модули ядра. На заре Linux умение писать такие программные компоненты было необходимым, поскольку для эффективного администрирования операционной системы требовалось адаптировать ее к конкретной аппаратной среде. Разрабатывать программы без специальных знаний для разновидностей системы UNIX еще труднее. (Выражаем благодарность компании IBM за прекрасную документацию по разработке драйверов (и не только драйверов).)

К счастью, администраторам не приходится копаться в коде ядра. Считается, что подобной деятельностью должны заниматься разработчики ядра и драйверов, а администраторам следует сосредоточивать усилия на организации работы пользователей. Администраторы могут настраивать ядро и добавлять существующие модули, как описано в данной главе, но им не обязательно проходить курс программирования на языке C или языке ассемблера. (Это не всегда справедливо.)

## 13.1. Адаптация ядра

Все платформы UNIX, описанные в этой книге, используют монолитные ядра, в которых вся операционная система работает в пространстве ядра, т.е. в разделе памяти, зарезервированном для привилегированных функций операционной системы. В монолитном ядре такие службы, как драйверы устройств, взаимодействие процессов, виртуальная память и планирование (диспетчеризация), выполняются в одном и том же адресном пространстве. Такой подход идет вразрез с архитектурой "микроядра", в которой многие из перечисленных служб выполняются в пользовательском (непривилегированном) режиме (т.е. в виде регулярных процессов). Преимущества и недостатки этих двух архитектур горячо обсуждались годами, но большинство разработчиков ядра соглашаются с тем, что оба подхода заслуживают внимания.

По сути, Linux также представляет собой монолитное ядро, но оно всегда позволяет использовать драйверы пространства пользователя для многих устройств. Такие проекты, как Gelato, UIO, FUSE и FUSD, предоставляют интерфейсы с устройствами в пространстве пользователя. Тем не менее большинство драйверов все еще реализовано в режиме ядра (защищенном режиме работы процессора), большей частью, по причинам быстродействия.

Современные монолитные ядра поддерживают загрузку модулей по запросу, поэтому вы можете включать драйверы устройств и другие функции ядра по необходимости без пересоздания ядра и перезагрузки системы. Драйверы, файловые системы и новые системные вызовы реализованы в большинстве случаев как модули. Память, используемая модулем, выделяется и освобождается по мере загрузки кода или его удаления, что особенно полезно для встроенных систем с ограниченной памятью, поскольку разработчики могут настроить ядро на возможность удаления ненужных устройств.

Ядро может узнать о системном оборудовании различными способами. Самый простой — предоставление ядру явной информации об устройствах, которые необходимо найти (или в отношении которых можно сделать вид, будто их не существует, как иногда бывает). Некоторые ядра самостоятельно ищут устройства либо на этапе начальной загрузки, либо динамически в процессе работы системы. Последний метод наиболее часто

применяется для современных USB-устройств: карт памяти, цифровых камер, принтеров и пр. Linux предоставляет поддержку широкого множества таких устройств. Системы AIX и HP-UX обеспечивают весьма ограниченную поддержку, а Solaris занимает, можно сказать, промежуточное положение.

В табл. 13.1 указано местоположение каталога построения ядра и стандартное название инсталлированного ядра в каждом из наших примеров систем.

**Таблица 13.1. Каталоги построения и названия ядер**

| Система | Каталог построения | Ядро |
|---|---|---|
| Linux | `/usr/src/linux` | `/vmlinuz` или `/boot/vmlinuz` |
| Solaris | –[a] | `/platform/hardware-class-name/unix` |
| HP-UX | `/stand` | `/stand/vmunix` |
| AIX | –[б] | `/usr/lib/boot/unix` |

[a] Администраторы редко занимаются построением ядер Solaris, но когда это случается, они создают произвольный каталог построения.

[б] Ядро системы AIX никогда не перестраивается, даже при добавлении новых модулей и устройств.

## 13.2. Драйверы и файлы устройств

Драйвер устройства — это программа, которая организует взаимодействие системы с аппаратным компонентом определенного типа. Драйвер играет роль переводчика команд конкретного устройства на "язык" API-функций ядра. Благодаря наличию драйверов обеспечивается приемлемый уровень независимости ядра от внешних устройств.

В большинстве случаев драйверы устройств являются компонентами ядра, а не пользовательскими процессами. Тем не менее доступ к драйверу возможен как из ядра, так и со стороны команд пользовательского уровня. Для последних создаются специальные файлы устройств, хранящиеся в каталоге `/dev`. Ядро преобразует операции, выполняемые над этими файлами, в вызовы функций драйвера.

Учитывая темпы появления новых устройств, практически невозможно выпускать основные дистрибутивы операционных систем, в которых интегрирована поддержка всех новинок. Поэтому нередко приходится самостоятельно добавлять в систему новые драйверы.

В каждой системе можно инсталлировать только "свои" драйверы, причем зачастую только те, которые разработаны специально для конкретной версии ядра. Драйверы для других операционных систем (например, Windows) работать не будут. Об этом следует помнить, покупая новое оборудование[1]. Кроме того, различные устройства в различной степени совместимы с дистрибутивами Linux и при работе в ее среде могут предоставлять различные функциональные возможности. Поэтому, выбирая оборудование, следует учитывать опыт его применения в других организациях.

Изготовители оборудования обращают все больше внимания на рынки UNIX и Linux и иногда создают UNIX- и Linux-версии драйверов. Если повезет, к устройству будут прилагаться и драйвер, и инструкции по его инсталляции. Но с наибольшей вероятностью нужный драйвер можно найти на каком-нибудь неофициальном веб-сайте. Как бы то ни было, ниже мы опишем, что происходит в системе при добавлении в нее драйвера устройства.

[1] Проект NDISwrapper для некоторых сетевых устройств позволяет использовать драйверы Windows под управлением Linux. Подробнее см. *sourceforge.net/projects/ndiswrapper*.

## Файлы и номера устройств

Для многих устройств в каталоге **/dev** имеются соответствующие специальные файлы; исключение в современных операционных системах составляют лишь сетевые устройства. Сложные серверы могут поддерживать сотни различных устройств. Solaris с этими сложностями прекрасно справляется за счет использования отдельного подкаталога **/dev** для каждого типа устройства: жесткого диска, компакт-диска, терминала и т.д.

С каждым файлом связаны старший и младший номера устройства. Посредством этих номеров ядро преобразует обращения к файлу устройства в вызовы нужного драйвера.

Старший номер устройства обозначает драйвер, за которым закреплен данный файл (другими словами, он определяет тип устройства). Младший номер устройства обычно указывает на то, к какому конкретно устройству данного типа следует обращаться. Младший номер иногда называют *номером модуля*.

Узнать номера устройств позволяет команда **ls -l**.

```
linux$ ls -l /dev/sda
brw-rw----  1 root disk 8, 0 Jul 13 01:38 /dev/sda
```

В этом примере выдана информация о первом SCSI-диске. Его старший номер равен 8, а младший — 0.

Младший номер иногда используется драйвером для выбора или активизации определенных функций (характеристик) устройства. Например, накопителю на магнитной ленте может соответствовать два файла в каталоге **/dev**, один из которых при закрытии будет автоматически обеспечивать обратную перемотку, а второй — нет. Драйвер волен интерпретировать младший номер устройства по своему усмотрению. Особенности работы драйвера можно узнать на соответствующей man-странице.

Файлы устройств бывают двух типов: блочно-ориентированные (блочные) и байт-ориентированные (символьные). Чтение из блочного устройства и запись в него осуществляются по одному блоку за раз (блок — это группа байтов, размер которой обычно кратен 512), тогда как чтение из символьного устройства и запись в него происходят в побайтовом режиме. Диски и магнитные ленты "ведут двойную жизнь", а терминалы и принтеры — нет. Драйверы устройств предоставляют стандартный интерфейс с ядром. Каждый драйвер оснащен процедурами для выполнения некоторых или всех перечисленных ниже функций.

```
attach     close      dump       ioctl      open       probe
psize      read       receive    reset      select     stop
strategy   timeout    transmit   write
```

Определенные системные функции удобно реализовывать, используя драйвер устройства, даже если связанное с ним устройство не существует. Такие фантомные устройства называют *псевдоустройствами*. Например, пользователю, зарегистрировавшемуся в системе по сети, назначается псевдотерминал (PTY), который с точки зрения высокоуровневого программного обеспечения ничем не отличается от обычного последовательного порта. Благодаря такому подходу программы, написанные в эпоху алфавитно-цифровых терминалов, могут функционировать в мире окон и сетей. Вот еще примеры псевдоустройств: **/dev/zero**, **/dev/null** и **/dev/random**.

Когда программа выполняет операцию над файлом устройства, ядро перехватывает обращение к файлу, ищет в таблице переходов соответствующую функцию и передает управление соответствующей части драйвера. Для выполнения необычных операций, не имеющих прямых аналогов в модели файловой системы (например, выталкивание дискеты из дисковода), используется системный вызов **ioctl**, который передает пользовательскую команду непосредственно драйверу.

## Создание файлов устройств

Файлы устройств можно создавать вручную с помощью команды **mknod**, которая имеет следующий синтаксис.

**mknod** *имя_файла тип старший младший*,

где *имя_файла* — это создаваемый файл устройства, *тип* — тип устройства (**c** — в случае символьного устройства и **b** — в случае блочного), а *старший* и *младший* — старший и младший номера устройства соответственно. Если вручную создаете файл устройства, драйвер которого уже имеется в ядре, просмотрите соответствующую этому драйверу man-страницу и найдите указанные в ней старший и младший номера.

Под управлением Linux система udev динамически управляет созданием и удалением файлов устройств в соответствии с реальным наличием (или отсутствием) устройств. Демон **udevd** слушает (т.е. ожидает) сообщения, исходящие от ядра, относительно изменений в статусе устройств. На основе данных о конфигурации, хранимых в каталогах **/etc/udev** и **/lib/udev**, демон **udevd** может предпринять множество действий при обнаружении устройства или его отсоединении. По умолчанию он просто создает файлы устройств в каталоге **/dev**. Подробно система udev описана в разделе 13.8.

В системах Solaris каталог **/dev** в действительности состоит из символических ссылок на файлы, хранящиеся в каталоге **/devices**, который представляет собой отдельную файловую систему. Файловая система устройств (Device File System — devfs) управляет файлами устройств, содержащихся в каталоге **/devices**. Эти файлы создаются автоматически во время начальной загрузки системы демоном **devfsadmd**, который продолжает выполняться после загрузки системы для обработки уведомлений об обновлениях, поступающих от ядра. Администраторы могут использовать демон **devfsadm** для настройки этого процесса, но большинство администраторов не испытывают необходимости в его использовании.

Ядро системы HP-UX создает файлы устройств во время загрузки системы. Если новое устройство присоединяется позже, администратор должен создавать файлы устройства вручную с помощью команд **mksf**, **insf** и **mknod**. Кроме того, утилита **smh** включает ограниченный интерфейс для просмотра информации об устройствах.

В системе AIX во время загрузки системы выполняется команда **cfgmgr** для конфигурирования устройств и инсталляции драйверов для устройств, которые прежде отсутствовали. Она выводит предупреждения относительно устройств, для которых программное обеспечение или драйверы не установлены. После обнаружения устройства система AIX запоминает его, поместив идентификатор в Object Data Manager, который мы рассмотрим в разделе 13.6. Команда **cfgmgr** создает файлы в каталоге **/dev** для устройств, которые успешно обнаружены и инициализированы.

Если в системе присутствуют описанные выше средства автоматизации (создающие файлы устройств), системным администраторам, использующим текущие версии UNIX и Linux, никогда не приходится вручную управлять файлами устройств с помощью команды **mknod**.

## Соглашения об именовании устройств

Соглашения об именовании устройств — это нечто весьма неопределенное. Во многих случаях применяется древняя методика, которую использовали еще при работе на компьютерах PDP-11 фирмы DEC.

Для устройств, имеющих как блочную, так и символьную идентичности, имя устройства с посимвольным вводом-выводом обычно начинается с буквы "r" (от англ. *raw*). Сравните, например, имена `/dev/da0` и `/dev/rda0`. Альтернативное соглашение предлагает хранить файлы "символьных" устройств в подкаталоге, имя которого начинается с буквы "r" (сравните, например, имена `/dev/dsk/dks0d3s0` и `/dev/rdsk/dks0d3s0`). Однако буква "r" не всегда означает использование исходного, или необработанного, файла устройства (*raw* device file).

🕮 Подробнее о последовательных портах рассказывается в главе 31.

Имена файлов последовательных портов обычно состоят из префикса `tty` и последовательности букв, обозначающих конкретный интерфейс, к которому подключен порт. Иногда для терминала имеется несколько файлов устройств. Дополнительные файлы предназначены для поддержки альтернативных методов управления потоками и протоколов блокировки.

Имена накопителей на магнитной ленте часто включают не только ссылку на сам накопитель, но и указание на то, перематывается ли лента после закрытия устройства. У каждого изготовителя свой алгоритм действий.

Соглашения об именовании устройств для файлов, которые представляют жесткие диски и SSD (Solid-State Disk — полупроводниковые диски), в большинстве систем довольно сложны. Подробнее см. раздел 8.5.

## Сравнение пользовательских ядер с загружаемыми модулями

При инсталляции системы создается базовое ядро, которое рассчитано на запуск большинства приложений в большинстве аппаратных сред. В эту конфигурацию входит множество различных драйверов устройств и дополнительных пакетов. Кроме того, некоторые драйверы можно динамически вставлять в работающее ядро. В Linux система udev может также в реальном времени управлять такими изменениями, как вставка USB-устройства.

Существуют различные "идеологические" направления на предмет того, должны ли промышленные серверы иметь ядра, построенные по специальному заказу. Несмотря на некоторый потенциальный выигрыш в производительности, особенно во встроенных системах, не использующих большой объем памяти, выбор между наложением заплат и обновлением системы обычно является решающим фактором при принятии решений. Если не существует серьезной необходимости выжимать из системы "последние соки" по части производительности, мы рекомендуем использовать ядро из ассортимента поставщика.

Когда дело касается поддержки ядра, умный администратор обычно ведет себя как самый ленивый. Используйте везде, где только возможно, динамические модули. Избегайте строить ядра на заказ, если в этом нет большой необходимости. В системах Linux большинство USB-устройств можно подключать без какого бы то ни было вмешательства администратора.

# 13.3. Конфигурирование ядра Linux

Существует четыре основных метода конфигурирования ядра Linux:

- модификация настраиваемых (динамических) параметров ядра;
- повторное создание ядра (компиляция исходных файлов ядра с возможными модификациями и дополнениями);
- динамическая загрузка новых драйверов и модулей в существующее ядро;
- передача ядру директив на этапе начальной загрузки или GRUB (подробнее об этом рассказывалось в разделе 3.2).

Каждый из перечисленных методов имеет свою область применения. Проще всего настраивать параметры ядра. Именно так обычно и поступают. Компиляция ядра — наиболее сложный метод, и к нему прибегают реже всего. Но любой метод вполне можно освоить, это лишь вопрос практики.

## Конфигурирование параметров ядра linux

Многие модули и драйверы ядра разрабатывались с учетом того, что все в системе подвержено изменению. Для повышения гибкости были созданы специальные информационные каналы, которые позволяют системному администратору динамически корректировать различные параметры, определяющие размеры внутренних таблиц ядра и его поведение в конкретных ситуациях. Все эти каналы расположены в файловой системе **/proc** (также называемой **procfs**), которая представляет собой интерфейс между ядром и приложениями пользовательского уровня. Часто большие приложения пользовательского уровня (особенно такие приложения "инфраструктуры", как базы данных) требуют настройки параметров ядра.

Специальные файлы в каталоге **/proc/sys** позволяют просматривать и динамически изменять значения параметров ядра. Они напоминают стандартные файлы Linux, но на самом деле являются "шлюзами" к ядру. Если в одном из файлов есть значение, которое требуется изменить, можно попытаться осуществить запись в этот файл. К сожалению, не все файлы доступны для записи (независимо от прав доступа), а объем имеющейся документации оставляет желать лучшего. Информацию о некоторых значениях и их назначении можно получить в подкаталоге **Documentation/syscnt** каталога с исходным кодом ядра.

Например, чтобы изменить максимальное число файлов, которые одновременно могут быть открыты в системе, выполните такие действия.

```
linux$ cat /proc/sys/fs/file-max
34916
linux$ sudo sh -c "echo 32768 > /proc/sys/fs/file-max"
```

Со временем вырабатывается привычка к столь нетрадиционному способу взаимодействия с ядром, тем более что он очень удобен. Но сразу предупредим: изменения не сохраняются при перезагрузке.

В табл. 13.2 перечислены некоторые обычно настраиваемые параметры. В разных дистрибутивах значения, действующие по умолчанию, варьируются в широких пределах.

Более надежный способ модификации тех же параметров в большинстве систем реализован в виде команды **sysctl**. Она либо берет значения переменных из командной строки, либо читает список пар *переменная=значение* из файла. По умолчанию файл **/etc/sysctl.conf**, в котором заданы начальные (пользовательские) значения параметров, считывается на этапе начальной загрузки.

**Таблица 13.2. Файлы каталога `/proc/sys`, содержащие некоторые настраиваемые параметры ядра**

| Каталог[a] | Файл/параметр | Назначение и комментарии |
|---|---|---|
| C | `autoeject` | Автоматически выталкивает компакт-диск при размонтировании дисковода CD-ROM |
| F | `file-max` | Задает максимальное число открытых файлов. В системе, которой приходится работать с большим количеством файлов, можно попытаться увеличить это значение до 16384 |
| F | `inode-max` | Задает максимальное число открытых индексных дескрипторов в одном процессе. Этот параметр полезен при создании приложения, которое открывает десятки тысяч дескрипторов файлов |
| K | `ctrl-alt-del` | Перезагружает при нажатии комбинации клавиш <Ctrl+Alt+Del>. Применение этого параметра может диктоваться личными предпочтениями или соображениями повышения уровня безопасности незащищенных серверных консолей |
| K | `printk ratelimit` | Минимальный интервал между сообщениями ядра, в секундах |
| K | `printk_ ratelimit_burst` | Устанавливает количество последовательных сообщений, которые должны быть получены, прежде чем значение минимального интервала между сообщениями `printk` действительно вступит в действие |
| K | `shmmax` | Задает максимальный размер совместно используемой памяти |
| N | `conf/default/ rp_filter` | Включает проверку маршрута к исходному файлу. Этот механизм противодействия имитации соединений вынуждает ядро не обращать внимания на пакеты, полученные по "невозможным" маршрутам |
| N | `icmp_echo_ ignore_all` | Задает игнорирование ICMP-запросов, если равен 1 |
| N | `icmp_echo_ ignore_ broadcasts` | Задает игнорирование широковещательных ICMP-запросов, если равен 1. Практически во всех ситуациях целесообразно устанавливать это значение равным 1 |
| N | `ip_forward` | Разрешает перенаправление IP-пакетов, если равен 1. Если же компьютер Linux используется в качестве маршрутизатора, это значение необходимо устанавливать равным 1 |
| N | `ip_local_port_ range` | Определяет диапазон локальных портов, выделяемый во время конфигурирования соединения. Для того чтобы повысить производительность серверов, которые инициируют много исходящих соединений, диапазон этого параметра следует расширить до 1024–65000 |
| N | `tcp_fin_timeout` | Определяет интервал ожидания (в секундах) заключительного пакета FIN. Для того чтобы повысить производительность серверов, через которые проходит большой объем трафика, следует устанавливать более низкое значение (порядка 20) |
| N | `tcp_syncookies` | Защищает от атак волнового распространения пакетов SYN. Этот параметр необходимо включать при наличии подозрения о наличии атак отказа от обслуживания (DOS). |

[a] F — `/proc/sys/fs`, N — `/proc/sys/net/ipv4`, K — `/proc/sys/kernel`, C — `/proc/sys/dev/cdrom`

Например, команда

```
linux$ sudo sysctl net.ipv4.ip_forward=0
```

отключает перенаправление IP-пакетов. (В качестве альтернативного варианта вы можете просто отредактировать файл `/etc/sysctl.conf` вручную.) Обратите внимание на

то, что символы косой черты в имени файла, хранящегося в структуре каталога **/proc** **/sys**, заменяются точками.

## Сборка ядра Linux

Операционная система Linux развивается столь быстрыми темпами, что рано или поздно ее ядро приходится компилировать заново. Постоянно появляются новые "заплаты" к ядру, драйверы устройств и новые функции. Все это вызывает противоречивые чувства. С одной стороны, удобно иметь под рукой всегда "самое новое и свежее", а с другой — стремление идти в ногу со временем требует постоянных усилий и значительных затрат времени.

Кроме того, если вы используете "стабильную" версию, сборка собственного ядра вряд ли потребуется. Первоначально в Linux была принята схема нумерации ядер, согласно которой вторая часть номера версии указывает на то, является ядро стабильным (четный номер) или находится в стадии разработки (нечетный номер). Например, ядро версии 2.6.6 — стабильное, в отличие от ядра версии 2.7.4. В настоящее время эта схема нумерации соблюдается не всегда. Поэтому за официальной информацией по этому вопросу лучше обратиться на сайт `kernel.org`. Этот сайт является также наилучшим источником исходного кода ядра Linux на тот случай, если конкретный дистрибутив (или его поставщик) не вызывает доверия.

## Не чините то, что еще не поломано

Хороший системный администратор тщательно взвешивает все за и против, планируя обновление ядра. Конечно, самая последняя версия — это всегда модно, но является ли она столь же стабильной, что и текущая версия? Нельзя ли подождать до конца месяца, когда будет инсталлироваться целая группа "заплат"? Важно, чтобы желание быть "на гребне волны" не шло вразрез с интересами пользователей.

Возьмите за правило устанавливать обновления и заплаты, только когда планируемая выгода (обычно определяемая на основе критериев надежности и производительности) окупит усилия и время, затрачиваемые на инсталляцию. Если выгоду оценить затруднительно, это верный признак того, что с обновлением можно подождать. (Безусловно, заплаты, связанные с безопасностью, следует инсталлировать без промедления.)

## Конфигурирование параметров ядра

В данной главе мы используем строку *путь_к_исходным_текстам_ядра* в качестве шаблона любого каталога, который выбран для хранения исходного кода ядра. Большинство дистрибутивов инсталлируют исходные файлы ядра в каталог **/usr/src**. В любом случае, чтобы можно было создавать ядро системы, должен быть инсталлирован пакет его исходных кодов. Об инсталляции пакетов рассказывается в разделе 12.4.

Весь процесс сосредоточен вокруг файла **.config**, находящегося в каталоге исходных кодов ядра. В данном файле приведена вся информация, касающаяся конфигурирования ядра, но его формат очень труден для понимания. Для выяснения смысла различных параметров воспользуйтесь пособием по параметрам конфигурирования ядра, приведенным в файле

*путь_к_исходным_текстам_ядра*/**Documentation/Configure.help**.

Для того чтобы не редактировать файл **.config** напрямую, в Linux предусмотрено несколько целевых модулей утилиты **make**, реализующих различные интерфейсы кон-

фигурирования ядра. Если в системе инсталлирована оболочка KDE, то удобнее всего воспользоваться командой **make xconfig**. Аналогично, если вы используете оболочку GNOME, вероятно, лучше всего использовать команду **make gconfig**. Эти команды отображают окно конфигурации, в котором можно выбрать драйверы устройств, добавляемые к ядру (или компилируемые в качестве загружаемых модулей).

В отсутствие графической среды на помощь придет команда **make menuconfig**, работающая на основе библиотеки **curses**. Наконец, есть старая команда **make config**, которая выводит запрос на изменение каждого конфигурационного параметра и не позволяет изменять ранее установленные значения. Мы рекомендуем пользоваться командой **make xconfig** или **make gconfig**, если используемая среда их поддерживает, и командой **make menuconfig** — в противном случае. Применения команды **make config** лучше избегать (как самой негибкой и самой трудоемкой).

При переносе конфигурации существующего ядра в новую версию ядра (или файловую систему) можно использовать команду **make oldconfig** для считывания ранее использовавшегося файла **.config** и определения только тех параметров, которые изменились или являются новыми.

Все эти команды довольно просты, так как их действие сводится к установке нужных опций. Тем не менее они мало подходят для ситуации, когда нужно поддерживать несколько версий ядра для различных платформ или аппаратных конфигураций.

В любом случае генерируется файл **.config**, имеющий примерно такой вид.

```
# Automatically generated make config: don't edit
# Code maturity level options

CONFIG_EXPERIMENTAL=y

# Processor type and features
# CONFIG_M386 is not set
# CONFIG_M486 is not set
# CONFIG_M586 is not set
# CONFIG_M586TSC is not set
CONFIG_M686=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_X86_TSC=y
CONFIG_X86_GOOD_APIC=y
```

Как видите, содержимое файла не слишком понятно. Не дается никаких пояснений относительно того, что означают параметры **CONFIG_***. По сути, каждая строка **CONFIG_*** указывает на активизацию того или иного параметра ядра. Значение **y** говорит о том, что соответствующий компонент компилируется вместе с ядром; значение **m** означает, что компонент станет загружаемым модулем.

Не все компоненты можно сконфигурировать в виде модулей. Сведений об этом в файле **.config** нет, поэтому нужно самостоятельно покопаться в документации. Не так-то легко узнать и назначение параметров **CONFIG_***.

## Компиляция ядра

Формирование файла **.config** — самый важный этап конфигурирования ядра Linux, но нужно выполнить еще ряд действий, прежде чем будет получено готовое ядро.

Схема всего процесса такова:

- перейти в каталог верхнего уровня, содержащий исходные коды ядра;
- выполнить команду **make xconfig**, **make gconfig** или **make menuconfig**;
- выполнить команду **make dep** (не требуется для ядер версии 2.6.x и более поздних);
- выполнить команду **make clean**;
- выполнить команду **make**;
- выполнить команду **make modules_install**;
- скопировать файл **arch/i386/boot/bzImage** под именем **/boot/vmlinuz**;
- скопировать файл **arch/i386/boot/System.map** под именем **/boot/System.map**;
- отредактировать файл **/etc/lilo.conf** (LILO) или **/boot/grub/grub.conf** (GRUB) и добавить в него конфигурационные параметры для нового ядра.

Выполнять команду **make clean** не всегда нужно, но удаление всего лишнего позволит избежать множества проблем.

## Добавление драйвера устройства в Linux

В Linux драйверы устройств распространяются в одной из трех форм:

- "заплаты" к конкретной версии ядра;
- загружаемого модуля;
- инсталляционного сценария или пакета, устанавливающего соответствующие "заплаты".

Наиболее распространенная форма — инсталляционный сценарий или пакет. Если вам повезло и вы располагаете таким сценарием или пакетом для нового устройства, достаточно выполнить стандартную процедуру установки новой программы.

В большинстве случае установку "заплаты" для конкретной версии ядра можно выполнить с помощью следующей команды.

```
linux # cd путь_к_исходным_текстам_ядра ; patch -p1 < файл_заплаты
```

Если ни один из этих вариантов неприменим, вам, скорее всего, придется вручную интегрировать драйвер нового устройства в дерево каталогов исходного кода ядра. Ниже мы рассмотрим, как вручную добавить к ядру драйвер некоего вымышленного сетевого устройства "snarf". В действительности в Linux этот процесс достаточно утомителен, особенно по сравнению с некоторыми другими версиями UNIX.

В подкаталоге **drivers** дерева файлов исходного кода ядра найдите подкаталог, соответствующий типу добавляемого устройства. Вот как выглядит список существующих подкаталогов.

```
linux$ ls -F путь_к_исходным_текстам_ядра/drivers
acorn/      char/      i2c/        Makefile  net/      s390/     telephony/
acpi/       dio/       ide/        md/       nubus/     sbus/    usb/
atm/        fc4/       ieee1394/   media/    parport/  scsi/     video/
block/      gsc/       input/      message/  pci/      sgi/      zorro/
bluetooth/  hil/       isdn/       misc/     pcmcia/   sound/
cdrom/      hotplug/   macintosh/  mtd/      pnp/      tc/
```

Чаще всего драйверы добавляются в каталоги **block**, **char**, **net**, **scsi**, **sound** и **usb**. В них хранятся драйверы блочных устройств (например, IDE-дисков), символьных устройств (например, последовательных портов), сетевых устройств, SCSI-плат, звуко-

вых плат и USB-устройств соответственно. В других каталогах находятся, в частности, драйверы для самих шин (**pci**, **nubus** и **zorro**); маловероятно, чтобы потребовалось добавлять в них файлы. Некоторые каталоги содержат платформенно-зависимые драйверы (**macintosh**, **s390**, **acorn**).

Поскольку наше устройство "snarf" является сетевым, мы поместим его драйвер в каталог **drivers/net**. Потребуется модифицировать следующие файлы:

- **drivers/net/Makefile**, чтобы драйвер мог быть скомпилирован;
- **drivers/net/Kconfig**, чтобы имя устройства появилось в списке конфигурируемых параметров.

После помещения файлов с расширениями **.c** и **.h** в каталог **drivers/net/snarf** нужно добавить запись о драйвере в файл **drivers/net/Makefile**. Вот эта строка (она располагается ближе к концу файла).

```
obj-$(CONFIG_SNARF_DEV) += snarf/
```

Эта строка добавляет драйвер (хранящийся в каталоге **snarf/**) в процесс компиляции ядра.

Когда запись добавлена в файл **Makefile**, нужно убедиться в том, что устройство будет доступно для конфигурирования при настройке ядра. Все сетевые устройства должны быть перечислены в файле **drivers/net/Kconfig**. В нашем случае требуется записать в него приведенные ниже строки, чтобы драйвер был добавлен либо как загружаемый модуль, либо как часть ядра (в соответствии с тем, что было заявлено в файле **Makefile**).

```
config SNARF_DEV
tristate 'Snarf device support'
```

Первый компонент, следующий за ключевым словом **config**, — имя конфигурационного макроса, которое должно совпадать с компонентом, следующим за ключевым словом **CONFIG** в файле **Makefile**. Ключевое слово **tristate** означает, что драйвер может стать загружаемым модулем. Если это не соответствует действительности, вместо **tristate** необходимо указать ключевое слово **bool**. Следующим элементом инструкции является строка, которая будет отображаться в окне конфигурации. Это может быть произвольный текст, но он должен идентифицировать конфигурируемое устройство.

Как при компиляции в ядро нового драйвера устройства указать ядру, что оно должно использовать новый драйвер? В версиях ядра, предшествовавших версии 2.6, эта задача была весьма трудоемкой и требовала знаний в области программирования. Недавние архитектурные изменения, внесенные в модель драйверов устройств, позволяют драйверам стандартным образом связывать себя с ядром.

Подробное описание этого процесса выходит за рамки данной книги, но в результате драйверы устройств, созданные для версии 2.6 (и последующих), выполняют свою регистрацию с помощью макроса **MODULE_DEVICE_TABLE**. Этот макрос создает соответствующие скрытые связи, чтобы такие утилиты, как **modprobe** (описывается ниже в разделе 13.7), могли активизировать новые устройства в ядре.

## 13.4. Конфигурирование ядра Solaris

Во время загрузки ядро Solaris исследует устройства и инициализирует драйвер для каждого найденного устройства. Оно широко использует загружаемые модули и загружает код только для тех устройств, которые реально присутствуют в системе, если нет причин для других действий.

В зависимости от вашей точки зрения, такая автоматическая конфигурация делает конфигурирование пользовательского ядра в большей или меньшей степени необходимостью в Solaris, по сравнению с другими системами. В идеальном мире ядро должно корректно идентифицировать свою аппаратную среду 100% времени. К сожалению, ненадежное, нестандартное или просто проблемное оборудование (или драйверы) может подчас превратить эти блага цивилизации в источник мучений. Поэтому имеет смысл рассмотреть, как выполнить пользовательскую конфигурацию ядра Solaris и возникнет ли когда-либо необходимость это делать.

## Пространство ядра Solaris

Для того чтобы сделать загрузку модулей по запросу корректной, система Solaris опирается на организацию конкретных каталогов. Система Solaris работает в предположении, что определенные каталоги находятся в определенных местах, и эти каталоги должны содержать конкретные типы модулей:

- **/kernel** — общие модули для компьютеров, которые совместно используют набор команд;
- **/platform/***имя_платформы***/kernel** — модули, специфичные для одного типа компьютеров (например, Sun Fire T200);
- **/platform/***имя_класса_оборудования***/kernel** — модули, специфичные для одного класса оборудования (например, все компьютеры sun4u);
- **/usr/kernel** — аналогично каталогу **/kernel**.

Элементы *имя_платформы* и *имя_класса_оборудования* можно определить с помощью команд **uname -i** и **uname -m** соответственно.

Рассмотрим пример.

```
solaris$ uname -i
SUNW,Sun-Fire-T200
solaris$ uname -m
sun4v
```

Во время загрузки Solaris, чтобы найти ядро, проверяет путь

**/platform/***имя_платформы***/kernel:/kernel:/usr/kernel**

Сначала находятся файлы с именами **unix**, а затем — файлы с именами **genunix**. **Genunix** — это обобщенное ядро, которое представляет независимую от платформы часть базового ядра.

Каждый из перечисленных выше каталогов может содержать несколько стандартных подкаталогов, приведенных в табл. 13.3. Поскольку эти подкаталоги могут существовать внутри любых каталогов ядра, для обозначения любого или всех каталогов ядра мы используем обобщенное имя *ЯДРО*.

**Таблица 13.3. Подкаталоги каталогов ядра Solaris**

| Подкаталог | Его содержимое |
|---|---|
| **drv** | Загружаемые файлы объектов для драйверов устройств. Файлы конфигурации, которые перечисляют тестовые адреса для каждого устройства |
| **misc** | Загружаемые файлы объектов для смешанных программ ядра |
| **cpu** | Специальный ЦПУ-модуль для UltraSPARC |
| **strmod** | Модули STREAMS |

*Окончание табл. 13.3*

| Подкаталог | Его содержимое |
|---|---|
| `sparcv9` | 64-разрядное ядро |
| `fs` | Модули ядра, связанные с файловой системой |
| `exec` | Модули для декодирования существующих форматов |
| `sched` | Планировщики операционной системы |
| `sys` | Загружаемые системные вызовы |
| `genunix` | Обобщенное платформенно-независимое ядро |
| `unix` | Базовое ядро конкретной платформы |

Обычно не приходится изменять какие бы то ни было файлы в этих каталогах, если не нужно инсталлировать драйвер нового устройства. У этого правила может быть одно исключение в виде файлов `.conf` в каталоге *ЯДРО*`/drv`, которые задают параметры конфигурации конкретных устройств. Однако необходимость изменять их возникает редко, и в действительности вы будете это делать только в случае, если есть соответствующее предписание от производителя устройства.

## Конфигурирование ядра с помощью файла /etc/system

В системе Solaris файл `/etc/system` служит в качестве основного файла конфигурации для ядра. В табл. 13.4 перечислены директивы и переменные, которые могут быть включены в этот файл. Директивы — это самостоятельные ключевые слова; переменным же должны быть присвоены значения с помощью директивы `set`.

**Таблица 13.4. Директивы и переменные, используемые в файле `/etc/system`**

| Имя | Тип[a] | Значение |
|---|---|---|
| `rootfs` | D | Задает тип файловой системы корневого раздела |
| `rootdev` | D | Задает местоположение корневого раздела |
| `forceload` | D | Задает драйверы ("модули"), которые должны быть загружены |
| `exclude` | D | Задает модули, которые не должны быть загружены |
| `moddir` | D | Задает новый путь к модулям |
| `set` | D | Устанавливает переменные настройки ядра (например, `maxusers`) |
| `maxusers` | V | Управляет размерами таблицы и другими параметрами |
| `pt_cnt` | V | Устанавливает количество доступных псевдотерминалов (PTY) |
| `max_nproc` | V | Устанавливает максимальное количество процессов |
| `maxuprc` | V | Устанавливает максимальное количество пользовательских процессов |

[a] D — директива, V — переменная

Обращение к файлу `/etc/system` происходит во время загрузки, и если его содержимое будет искажено, то система больше не загрузится. Команда `boot -a` позволяет указать резервную копию файла `/etc/system`, если вы создали таковую. (Если у вас нет резервной копии, а существующий оригинал не работает, можете использовать файл `/dev/null`.)

Рассмотрим пример файла `/etc/system` для простого ядра.

```
rootfs:ufs
rootdev:/sbus@1,f8000000/esp@0,800000/sd@3,0:a
```

Этими строками определяется, что корневая файловая система будет иметь тип UFS (UNIX File System) и что она будет храниться в дисковом разделе **sd3a**. Синтаксис, используемый для задания корневого устройства, идентичен используемому Sun-монитором **openprom**. На разных платформах он разный, поэтому обратитесь к соответствующему руководству по работе с аппаратурой или используйте символические ссылки в **/dev**, которые преобразуют "странные" имена в понятные. Команда **ls -l**, заданная после ссылки, покажет ее точное длинное имя.

```
moddir: /platform/SUNW,Sun-Fire-T200/kernel:/platform/sun4v/kernel:
  /kernel:/usr/kernel
```

Эта строка (физически занимающая две строки) задает путь поиска для загружаемых модулей. Это значение предлагается man-страницей ядра, но оно не действует по умолчанию, поэтому вы должны задать его в явном виде.

```
exclude: lofs
forceload: drv/sd
```

Первая строка исключает зацикленную файловую систему из ядра, а вторая принудительно обеспечивает загрузку общего драйвера SCSI (**sd**).

```
set maxusers=64
```

Эта строка надлежащим образом изменяет размеры таблиц ядра для 64 параллельных регистрационных имен.

## Добавление в систему Solaris драйверов устройств

Solaris-драйверы обычно распространяются как пакеты. Для добавления драйвера устройства в систему используйте утилиту **pkgadd**. Если драйверы не распространяются как пакеты или добавление пакета завершилось неудачно, обычно тогда драйверы добавляют вручную, поскольку все они реализованы как загружаемые модули ядра.

Solaris-драйверы почти всегда распространяются как объектные файлы, а не в виде исходного кода, как это принято в системах Linux. В этом примере мы добавляем в Solaris устройство "snarf". Его драйвер должен быть представлен, как минимум, двумя файлами: **snarf.o** (собственно драйвер) и **snarf.conf** (файл конфигурации). Оба должны находиться в каталоге **/platform/`uname -m`/kernel/drv**.

После копирования конфигурационный файл можно отредактировать, чтобы задать параметры конкретного устройства. Обычно этого делать не требуется, но иногда некоторые параметры доступны для "тонкой" настройки.

После копирования файлов нужно загрузить модуль. Модули подключаются к работающему ядру командой **add_drv** (подробнее о загружаемых модулях речь пойдет ниже в этой главе). В нашем случае команда имеет вид **add_drv snarf**. Вот и все! Более простую процедуру трудно себе представить.

## Отладка Solaris-конфигурации

Поскольку Solaris формирует свой "взгляд на мир" на лету, отладка проблемного компьютера может вызвать трудности. К счастью, Solaris предоставляет ряд инструментов, которые отображают текущую конфигурацию компьютера.

Команда **prtconf** позволяет вывести общую конфигурацию компьютера, включая тип устройства, номер модели, объем памяти и некоторую информацию о сконфигурированных аппаратных устройствах. Строки, которые описывают устройства (драй-

веры), имеют отступы, чтобы обозначить существующие зависимости. Удобная опция **prtconf -D** отображает имя драйвера для каждого устройства.

Как видно из следующего фрагмента результатов выполнения команды **prtconf**, несколько строк содержат текст "driver not attached" (драйвер не присоединен). Это сообщение может иметь несколько значений: нет драйвера для устройства, устройство сконфигурировано, но не присоединено к системе или устройство не используется и драйвер не загружен.

```
solaris$ sudo prtconf
System Configuration:  Sun Microsystems  i86pc
Memory size: 580 Megabytes
System Peripherals (Software Nodes):

i86pc
    scsi_vhci, instance #0
    isa, instance #0
      i8042, instance #0
        keyboard, instance #0
        mouse, instance #0
    lp, instance #0 (driver not attached)
    asy, instance #0 (driver not attached)
    asy, instance #1 (driver not attached)
    fdc, instance #0
      fd, instance #0 (driver not attached)
    pit_beep, instance #0
```

Команда **prtconf -D** показывает, какие драйверы загружены в каталог **/etc/system**.

```
solaris$ sudo prtconf -D
System Configuration:  Sun Microsystems  i86pc
Memory size: 580 Megabytes
System Peripherals (Software Nodes):

i86pc (driver name: rootnex)
   scsi_vhci, instance #0 (driver name: scsi_vhci)
   isa, instance #0 (driver name: isa)
     i8042, instance #0 (driver name: i8042)
       keyboard, instance #0 (driver name: kb8042)
       mouse, instance #0 (driver name: mouse8042)
     lp, instance #0 (driver name: ecpp)
     asy, instance #0 (driver name: asy)
     asy, instance #1 (driver name: asy)
     fdc, instance #0 (driver name: fdc)
       fd, instance #0 (driver name: fd)
     pit_beep, instance #0 (driver name: pit_beep)
```

Команда **sysdef**, помимо информации, выдаваемой командой **prtconf**, также перечисляет драйверы псевдоустройств, настраиваемые параметры ядра и имена файлов загружаемых модулей. Если вы модифицируете стандартное ядро для важного компьютера, включите в его документацию вывод команды **sysdef**.

Команда **modinfo** сообщает информацию о динамически загружаемых модулях. Solaris (среди прочего) динамически загружает драйверы устройств, модули **STREAMS** и драйверы файловых систем. Не удивляйтесь, если результат выполнения команды **modinfo** будет содержать более 200 элементов. Больше информации о команде **modinfo** вы найдете в разделе 13.7.

# 13.5. Конфигурирование ядра HP-UX

Ядро HP-UX — самое монолитное среди наших примеров операционных систем, и для него предпочтительнее загружать большинство модулей статически. Его файл конфигурации отличается сложностью и беспорядочностью. К счастью, HP предоставляет удобное средство конфигурации **kcweb**, которое в случае доступности X Windows и браузера работает с графическим интерфейсом; в противном случае используется режим командной строки. Для того чтобы установить режим командной строки, используйте команду **kcweb -t**.

HP-UX считывает параметры конфигурации ядра (модули и настраиваемые значения) из файла **/stand/system**. Этот файл управляется утилитой **kcweb** и другими инструментами, и администраторы не должны модифицировать его напрямую.

Модули и параметры конфигурации могут быть статическими или динамическими. Для изменения статического значения или инсталляции статического модуля необходимо перестраивать ядро и перезагружать систему.

Динамические модули загружаются и выгружаются по мере их использования, не требуя перезагрузки системы. Подобным образом динамически настраиваемые значения немедленно вступают в силу.

Список наиболее полезных настраиваемых свойств ядра HP-UX приведен в табл. 13.5.

**Таблица 13.5. Настраиваемые значения конфигурации ядра HP-UX**

| Переменная | Тип | Значение по умолчанию | Описание |
|---|---|---|---|
| **maxfiles_lim** | Динамическая | 4096 | Жесткое ограничение на количество открытых файлов для процесса |
| **maxfiles** | Статическая | 2048 | Мягкое ограничение на количество открытых файлов для процесса |
| **maxuprc** | Динамическая | 256 | Максимальное количество пользовательских процессов |
| **nproc** | Динамическая | 4200 | Максимальное количество процессов |
| **nflocks** | Динамическая | 4096 | Максимальное количество блокировок файлов |
| **ninode** | Статическая | 8192 | Максимальное количество открытых индексных дескрипторов (inodes) |
| **npty** | Статическая | 60 | Максимальное количество псевдотерминалов (PTY) |
| **nstrtel** | Статическая | 60 | Максимальное количество устройств, находящихся в сеансе сетевого теледоступа (telnet) |
| **nkthread** | Динамическая | 8416 | Максимальное количество потоков ядра |

Если необходимо внести изменения в статические модули или статические настраиваемые значения, утилита **kcweb** автоматически выполняет команду **mk_kernel** для построения нового ядра. Новое ядро вступает в силу после следующей перезагрузки системы.

# 13.6. Управление ядром AIX

Ядро AIX никогда не требует перестройки. Новые устройства конфигурируются динамически посредством черного IBM-ящика, известного как *администратор объектных данных* (Object Data Manager — ODM). Это довольно загадочная структура. Многие параметры, которые обычно можно настраивать

в других ядрах (например установки совместно используемой памяти), совсем не поддаются настройке в системе AIX. Они самостоятельно управляются ядром. Другие реконфигурируемые параметры управляются последовательностью шести команд настройки.

## Администратор объектных данных

Вместо хранения данных о конфигурации устройств в текстовых файлах или сценариях, AIX собирает их в ODM-базе данных атрибутов/значений. Еще один связующий слой соединяет эти списки свойств с конкретными устройствами (экземплярами драйверов) и связывает эти драйверы с данными о конфигурации.

Цель AIX состоит в поддержке сохраняемости данных, необходимых для конфигурирования устройств. Вместо возможности конфигурировать устройства "на лету" (например, с помощью команд **ifconfig** или **ndd**) и параллельной системы файлов конфигурации и сценариев для выполнения конфигурации во время загрузки, в AIX реализована попытка унифицировать эти функции, чтобы большинство изменений в устройствах происходило автоматически во время перезагрузки системы.

Но если бы вы "приняли красную пилюлю" и заглянули на происходящее внутри системы, сложность механизма вас бы могла испугать. Эта система имеет больше точек входа, чем традиционная система UNIX, а взаимодействие между компонентами не всегда очевидно. Вот как выглядит структура различных слоев этой системы.

- Администратор объектных данных (ODM) представляет собой хранилище конфигурации, подобное системному реестру в Microsoft Windows. В действительности оно еще сложнее, чем реестр Windows, поскольку реализует идею объектных схем и экземпляров, а не просто произвольные списки свойств.
- Программы получают доступ к ODM через библиотечные подпрограммы, но вы также можете работать с базой данных ODM с помощью команд **odmadd**, **odmcreate**, **odmdrop**, **odmshow**, **odmget**, **odmchange** и **odmdelete**[2].
- Семейство команд **chdev**, **lsdev**, **lsattr**, **mkdev**, **rmdev**, **lsconn** и **lsparent** преобразует ODM-данные конфигурации в информацию для конкретных устройств. В действительности AIX-команда **chdev** подобна Solaris- и HP-UX-команде **ndd** (см. раздел 14.13), но по умолчанию **chdev** записывает ваши изменения как в реальный драйвер, так и в ODM-базу данных конфигурации. Даже обычные параметры (такие, как имя хоста системы и детали статической маршрутизации) хранятся как атрибуты устройств (под устройством в данном случае понимается экземпляр "интернет-драйвера").
- Несколько административных утилит обеспечивают внешние интерфейсы для семейства команды **chdev**. Например, команда **mktcpip** ведет себя подобно постоянной команде **ifconfig**, которая преобразует свои аргументы в последовательность вызовов команды **chdev** в сетевых интерфейсах, оказывая влияние как на

[2] Дэн Фостер (Dan Foster), один из наших рецензентов, прокомментировал это так: "Управлять базой данных ODM напрямую с помощью **odm**-команд не рекомендуется, если вы точно не знаете, что именно делаете. Эти команды не выполняют никакого контроля ошибок в данных, которые вы модифицируете, тогда как обычные **ch**-/**mk**-/**rm**-средства проверяют достоверность данных до внесения изменений. Эти **odm**-команды можно сравнить с заряженным автоматом Калашникова (АК-47) без предохранительного механизма: одно неосторожное прикосновение — и несколько автоматных очередей тут же будет выпущено в несчастную цель".

активную, так и на сохраненную конфигурации. (А вы подумали, что ее синтаксис отражает синтаксис команды **ifconfig**? Вы подумали неправильно.)

- ODM — это средство пользовательского уровня, поэтому драйверы не получают к нему непосредственный доступ. Как и в случае с традиционной конфигурацией на базе текстовых файлов, некоторые программы должны считывать ODM-конфигурации во время загрузки и записывать соответствующие значения в работающие драйверы.

К счастью, большинству администраторов не нужно погружаться в сложности ODM благодаря SMIT (System Manager Interface Tool — интерфейсные средства администратора системы) и таким инструментам более высокого уровня, как **mktcpip**.

Одной обязательной утилитой для управления AIX-устройствами является команда **cfgmgr**. Выполняйте ее без аргументов от имени суперпользователя после добавления в систему нового оборудования, и это новое оборудование чудесным образом будет распознано и станет доступным для использования (ну, как правило, так и происходит). Если же драйверы устройств еще не загружены в базу данных ODM, команда **cfgmgr** любезно предложит вам пакет для инсталляции из инсталляционных средств AIX. Подробнее о команде **cfgmgr** можно узнать из соответствующей man-страницы.

## Настройка ядра

В системе AIX имеется шесть категорий настраиваемых значений и предусмотрено шесть соответствующих команд для их настройки. Большинство значений связано с оптимизацией рабочих характеристик. Назначение каждой из этих команд описано в табл. 13.6. Отталкиваясь от стандартного соглашения AIX, эти команды используют общий синаксис. Параметрами можно также управлять посредством интерфейса SMIT, а именно после "магического заклинания" **smit tuning**. Подробнее о каждой команде можно узнать из соответствующей man-страницы.

**Таблица 13.6. Команды установки настраиваемых параметров ядра в системе AIX**

| Команда | Область применения | Примеры аспектов, которые можно конфигурировать |
|---|---|---|
| **vmo** | Виртуальная память | Минимальное количество свободных страниц |
| **ioo** | Ввод-вывод | Асинхронное поведение при вводе-выводе.<br>Конфигурирование JFS2 |
| **schedo** | Планирование процессов | Временные интервалы, выделяемые процессам, и свойства процессов.<br>Управление виртуальными процессами |
| **no** | Сеть | IP-пересылка.<br>Размеры буферов TCP- и UDP-сокетов.<br>Значения времен существования пакетов |
| **nfso** | NFS | Поддержка формата UTF-8.<br>Поддержка делегирования.<br>Максимальное количество соединений NFS |
| **raso** | Надежность | Всего несколько параметров, ни один из которых не представляет большую ценность для администраторов |

Эти команды просты в применении. Например, чтобы разрешить IP-форвардирование, выполните следующую команду.

```
aix$ sudo no -o ipforwarding=1
```


Для того чтобы получить список всех доступных для настройки параметров подсистемы ввода-вывода, введите такую команду.

```
aix$ sudo ioo -a
```

Для того чтобы гарантировать, что ваши изменения сохранятся после перезагрузки, добавьте к любой из перечисленных выше команд флаг **-r**.

# 13.7. Загружаемые модули ядра

Загружаемые модули ядра в настоящее время являются общим фактором практически для всех версий UNIX. Каждый из наших примеров систем в той или иной форме реализует возможность динамической загрузки.

Благодаря наличию загружаемых модулей, появляется возможность подключать и отключать драйверы устройств и другие компоненты ядра непосредственно в процессе работы системы. Это значительно облегчает инсталляцию драйверов, поскольку исполняемый код ядра не нужно менять. Кроме того, уменьшается размер ядра, так как ненужные драйверы просто не загружаются.

Загружаемые драйверы очень удобны, но они не обеспечивают стопроцентную безопасность. При каждой загрузке и выгрузке модуля существует риск нарушить работоспособность ядра. Поэтому во избежание нарушения работы системы не используйте непроверенные модули.

Подобно другим аспектам управления устройствами и драйверами, реализация загружаемых модулей зависит от операционной системы. В следующих разделах описаны команды для систем Solaris и Linux, которые поддерживают больше устройств и предоставляют администраторам больше возможностей по конфигурированию, чем в других наших примерах систем.

## Загружаемые модули ядра в Linux

Система Linux сложнее и одновременно проще системы Solaris в области обработки загружаемых модулей ядра, по крайней мере с точки зрения системного администратора. В Linux почти все программные компоненты можно сделать загружаемыми модулями. Исключение составляет драйвер устройства, на котором расположена корневая файловая система, а также драйвер мыши PS/2.

Загружаемые модули обычно хранятся в каталоге **/lib/modules/***версия*, где последняя часть имени — это версия ядра Linux, сообщаемая командой **uname -r**. Получить список загруженных в данный момент модулей можно с помощью команды **lsmod**.

```
redhat$ sudo /sbin/lsmod
Module                  Size    Used by
ipmi_devintf           13064    2
ipmi_si                36648    1
ipmi_msghandler        31848    2 ipmi_devintf,ipmi_si
iptable_filter          6721    0
ip_tables              21441    1 iptable_filter
...
```

Как видно из списка, в системе установлены модули интеллектуального интерфейса управления платформой (Intelligent Platform Management Interface — IPMI) и брандмауэр **iptables**.

Приведем пример загрузки вручную модуля ядра **snarf**, который был установлен в предыдущем разделе.

```
redhat$ sudo insmod /путь/куда/snarf.ko
```

Загружаемым модулям ядра можно также передавать конфигурационные параметры.

```
redhat$ sudo insmod /путь/куда/snarf.ko io=0xXXX irq=X
```

После того как модуль вручную добавлен к ядру, удалить его можно только по явному запросу или при перезагрузке системы. Для удаления нашего модуля подойдет команда **rmmod snarf**.

Команду **rmmod** можно использовать в любой момент времени, но она будет работать, только если число действующих ссылок на модуль (это число указано в столбце **Used by** (Используется) результата выполнения команды **lsmod**) равно **0**.

Модули ядра Linux могут загружаться полуавтоматически с помощью команды **modprobe**, которая является оболочкой команды **insmod** и распознает межмодульные зависимости, параметры модулей, процедуры инсталляции и выгрузки. Эта команда просматривает файл **/etc/modprobe.conf**, чтобы узнать правила обработки каждого модуля.

Можно динамически создать файл **/etc/modprobe.conf**, соответствующий текущей конфигурации ядра, выполнив команду **modprobe -c**. Эта команда генерирует длинный файл, который выглядит примерно так.

```
#This file was generated by: modprobe -c
path[pcmcia]=/lib/modules/preferred
path[pcmcia]=/lib/modules/default
path[pcmcia]=/lib/modules/2.6.6
path[misc]=/lib/modules/2.6.6
…
# Aliases
alias block-major-1 rd
alias block-major-2 floppy
…
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
…
alias dos msdos
alias plip0 plip
alias ppp0 ppp
options ne io=x0340 irq=9
```

Инструкции **path** указывают на то, где находится конкретный модуль. Можно модифицировать или добавлять записи данного типа, если нужно хранить модули в нестандартных каталогах.

Инструкция **alias** обеспечивает привязку старших номеров блочных устройств, старших номеров символьных устройств, файловых систем, сетевых устройств и сетевых протоколов к соответствующим модулям.

Строки с ключевым словом **options** не генерируются динамически, но администратор должен добавить их вручную. Они задают параметры, передаваемые модулю при за-

грузке. Например, в следующей строке модулю устройства "snarf" сообщается его адрес ввода-вывода и вектор прерываний[3].

```
options snarf io=0xXXX irq=X
```

Команда **modprobe** понимает также инструкции **install** и **remove**. С их помощью указываются команды, которые выполняются, когда соответствующий модуль подключается к работающему ядру или отключается от него.

## Загружаемые модули ядра в Solaris

В Solaris практически каждый элемент системы представляет собой загружаемый модуль. Команда **modinfo** выводит список загруженных на данный момент модулей.

Результат выполнения этой команды выглядит так.

```
solaris$ modinfo
Id  Loadaddr    Size    Info  Rev     ModuleName
 1  ff07e000    3ba0     1     1      specfs (filesystem for specfs)
 2  ff086000    1340     -     1      swapgeneric (root/swap config)
 3  ff082000    1a56     1     1      TS (time sharing sched class)
 4  ff084000     49c     -     1      TS_DPTBL (Timesharing dispatch)
 5  ff095000   15248     2     1         ufs (filesystem for ufs)
 6  ff0b8000    20e0     1     1      rootnex  (sun4c root nexus)
 7  ff084a00     170    57     1         options (options driver)
 8  ff08dc00     2f4    62     1       dma (Direct Memory Access)
 9  ff08c000     968    59     1         sbus (SBus nexus driver)
...
```

В нашей системе Solaris список содержал более 80 строк. Многие элементы, которые в других системах UNIX (например, локальной файловой системе UFS) включены в ядро, в Solaris представляют собой загружаемые драйверы. Такая организация должна упрощать сторонним фирмам написание пакетов, которые можно легко и безшовно интегрировать в ядро (по крайней мере, теоретически).

Как описано выше в этой главе, вы можете добавить в конфигурацию ядра Linux любой драйвер с помощью команды **add_drv**. Эта команда позволяет загрузить драйвер в ядро и создать соответствующие ссылки на устройства (все эти ссылки пересоздаются при каждой загрузке ядра). После выполнения команды **add_drv** добавленный драйвер остается частью системы до тех пор, пока вы не удалите его. Вы можете выгружать драйверы вручную с помощью команды **rem_drv**.

Добавив драйвер с помощью команды **add_drv**, имеет смысл также выполнить команду **drvconfig**. Эта команда переконфигурирует каталог **/devices** и добавит любые файлы, необходимые для только что загруженного драйвера.

Загружаемые модули, которые не доступны посредством файлов устройств, можно загружать и выгружать с помощью команд **modload** и **modunload** соответственно.

[3] Если система установлена на нестандартном персональном компьютере, то довольно сложно подобрать такую конфигурацию, в которой векторы прерываний и адреса ввода-вывода не перекрываются. Для того чтобы узнать текущую конфигурацию системы, нужно просмотреть содержимое файлов ***/proc/interrupts*** и ***/proc/ioports***. Как правило, для современного оборудования ПК всех основных типов перекрытие векторов прерываний и адресов ввода-вывода не создает особых проблем.

# 13.8. Linux-менеджер устройств udev — полезно и приятно

Файлы устройств в течение многих лет представляли для администраторов серьезную проблему. Когда системы поддерживали лишь несколько типов устройств, обслуживание файлов устройств вручную было вполне приемлемым. Но по мере увеличения количества доступных устройств файловая система **/dev** загромождается, зачастую "обростая" файлами, не относящимися к текущей системе. Система Red Hat Enterprise Linux версии 3 включала более 18 000 файлов устройств, по одному для каждого возможного устройства, которое потенциально могло быть подключено к системе! Создание статических файлов устройств быстро становится крупной проблемой, создающей для системного администратора тупиковую ситуацию.

USB, FireWire, PCMCIA и другие интерфейсы с устройствами привносят дополнительные трудности. Например, если пользователь подключается к двум внешним жестким дискам, то для системы было бы удобно распознавать и автоматически монтировать каждый драйвер с постоянным именем устройства. В идеале драйвер, который в исходном положении распознается под именем **/dev/sda**, должен оставаться доступным под тем же именем (**/dev/sda**), несмотря на периодические отключения и безотносительно к активности других устройств и шин (каналов). Наличие таких динамических устройств, как камеры, принтеры, сканеры и другие типы съемных носителей данных, наводит дополнительную "тень на плетень" и более усложняет проблему постоянной идентичности.

Элегантное решение этих проблем пришло в виде менеджера устройств `udev`, представляющего собой реализованную в пространстве пользователя (а не внутри ядра) систему управления устройствами, которая информирует приложения конечного пользователя об устройствах при их подключении или отключении. Менеджер устройств `udev` опирается на описанную ниже систему **sysfs**, которая позволяет узнать, что происходит с системными устройствами. Он использует ряд специальных правил для понимания соответствующих соглашений о назначении имен. Менеджер `udev` автоматически поддерживает файлы устройств в каталоге **/dev** ( причем при минимальном их разрушении). Только те устройства, которые доступны в системе в данный момент, имеют файлы в каталоге **/dev**.

Администраторам Linux следует понимать, как работает системное правило менеджера udev, и знать, как использовать команду **udevadm**. Но прежде чем углубляться в эти детали, рассмотрим сначала базовую технологию **sysfs**.

## Файловая система sysfs: доступ к "сердцу" устройств в Linux

Виртуальная файловая система **sysfs** (была добавлена в ядро Linux в версии 2.6) предоставляет хорошо организованную и подробную информацию обо всех доступных устройствах, их конфигурациях и состоянии. Доступ к драйверу возможен как из ядра, так и со стороны команд пользовательского уровня.

Для выяснения всей информации об устройстве (используемом им прерывании IRQ, количестве блоков, поставленных в очередь на запись контроллером диска, и т.п.) можно исследовать каталог **/sys**, в котором обычно монтируется система **sysfs**. Один из основополагающих принципов формирования этой файловой системы состоит в том,

что каждый файл в каталоге **/sys** должен представлять только один атрибут связанного с ним устройства. Это соглашение обеспечивает определенное структурирование в остальном хаотичного набора данных.

В табл. 13.7 перечислены каталоги, включенные в корневой каталог **/sys**, каждый из которых является подсистемой, регистрируемой с помощью **sysfs**. Состав этих каталогов зависит от дистрибутива.

**Таблица 13.7. Подкаталоги каталога /sys**

| Каталог | Описание |
|---|---|
| **block** | Информация о блочных устройствах (например, о жестких дисках) |
| **bus** | Шины, известные ядру: PCI-E, SCSI, USB и др. |
| **class** | Дерево, организованное функциональными типами устройств, например звуковые и графические карты, устройства ввода и сетевые интерфейсы |
| **dev** | Информация об устройствах, разделенная между символьными и блочными устройствами |
| **devices** | Корректное представление (с точки зрения наследственности) всех обнаруженных устройств |
| **firmware** | Интерфейсы с такими зависимыми от платформы подсистемами, как ACPI |
| **fs** | Каталог для хранения некоторых (но не всех) файловых систем, известных ядру |
| **kernel** | Значение таких внутренних характеристик ядра, как кеш и виртуальная память |
| **module** | Динамические модули, загруженные ядром |
| **power** | Некоторые сведения о состоянии производительности системы (обычно не используется) |

Первоначально информация о конфигурации устройств, если таковая существовала, хранилась в файловой системе **/proc**. Хотя информация о процессах и ядре, связанная со временем выполнения, и в дальнейшем будет храниться в каталоге **/proc**, предполагается, что со временем вся информация об устройствах будет перемещена в каталог **/sys**.

## Исследование устройств с помощью команды udevadm

Команда **udevadm** запрашивает информацию об устройствах, инициализирует события, управляет демоном **udevd** и отслеживает события менеджера udev и ядра. Системные администраторы используют ее, в основном, для создания и тестирования правил, которые описываются в следующем разделе.

Команда **udevadm** в качестве своего первого аргумента ожидает одну из шести команд: **info**, **trigger**, **settle**, **control**, **monitor** или **test**. Особый интерес для системных администраторов представляют две команды: **info**, которая выводит информацию, зависящую от конкретного устройства, и **control**, которая запускает и останавливает работу менеджера устройств udev или заставляет его перезагрузить свои файлы правил. Команда **monitor** отображает события по мере того, как они происходят в системе.

Следующая команда отображает все udev-атрибуты для устройства **sdb**. Здесь результат выполнения команды представлен в усеченном виде, но в действительности он содержит список всех родительских устройств (например, шины USB), которые в дереве устройств являются предками устройства **sdb**.

```
linux$ udevadm info -a -n sdb
...
looking at device '/devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/
   1-1:1.0/host6/target6:0:0/6:0:0:0/block/sdb':
KERNEL=="sdb"
```

```
SUBSYSTEM=="block"
DRIVER==""
ATTR{range}=="16"
ATTR{ext_range}=="256"
ATTR{removable}=="1"
ATTR{ro}=="0"
ATTR{size}=="1974271"
ATTR{capability}=="53"
ATTR{stat}=="       71      986     1561      860        1        0        1
  12        0      592      872"
 ...
```

Все пути в результате выполнения команды **udevadm** (например, **/devices /pci0000:00/**...) даны относительно каталога **/sys**.

Этот результат отформатирован так, чтобы вы могли при построении правил обращаться к менеджеру udev. Например, если бы запись **ATTR{size}=="1974271"** была уникальной для этого устройства, вы могли бы скопировать этот фрагмент в правило в качестве идентифицирующего критерия.

Узнать дополнительные возможности и синтаксис команды **udevadm** вы сможете, обратившись к ее man-странице.

## Создание правил и постоянных имен

Менеджер устройств udev опирается на набор правил, определяющих возможности управления устройствами и присваивания им имен. Стандартные правила хранятся в каталоге **/lib/udev/rules.d**, а локальные — в каталоге **/etc/udev/rules.d**. Вам не нужно редактировать или удалять стандартные правила — их можно игнорировать или переопределить путем создания нового файла стандартных правил с прежним именем в пользовательском каталоге правил.

Главным файлом конфигурации для менеджера udev является файл **/etc/udev /udev.conf**. Файлы **udev.conf** в наших примерах дистрибутивов содержат только комментарии, за исключением одной строки, которая позволяет выполнять регистрацию ошибок.

Жаль, но из-за политических разногласий между дистрибьюторами и разработчиками совместная деятельность дистрибьюторов на ниве создания правил оставляет желать лучшего. Многие имена файлов в каталоге стандартных правил не меняются при переходе от дистрибутива к дистрибутиву, в то время как содержимое этих файлов претерпевает существенные изменения.

Файлы правил именуются в соответствии с шаблоном *nn-описание*.**rules**, где *nn* — обычно двузначное число. Файлы обрабатываются в лексическом порядке, поэтому меньшие числа обрабатываются первыми. Файлы из двух каталогов правил объединяются до того, как демон менеджера udev, **udevd**, проанализирует их. Суффикс **.rules** обязателен — без него файлы игнорируются.

Правила задаются в таком формате.

*условие*, [*условие*, ...] *назначение* [,*назначение* ...]

Здесь элемент *условие* определяет ситуации, в которых должно применяться данное правило. Каждое выражение условия состоит из ключа, знака операции и значения. Например, в качестве потенциального компонента правила может использоваться условие **ATTR{size}=="1974271"** (см. пример выше), отбирающее все устройства, атрибут **size** которых в точности равен значению **1 974 271**.

Большинство ключей, участвующих в задании условия, связаны со свойствами устройств (которые демон **udevd** получает из файловой системы **/sys**), но есть и такие ключи, которые относятся к другим контекстно-зависимым атрибутам, таким как обрабатываемая операция (например, добавление устройства или его удаление из системы). Все элементы *условие* должны сопоставляться в порядке активизации правила.

Ключи, воспринимаемые менеджером устройств udev, представлены в табл. 13.8.

**Таблица 13.8. Ключи, воспринимаемые менеджером устройств udev**

| Ключ условия | Действие |
|---|---|
| **ACTION** | Сопоставляется с типом события, например добавление или удаление |
| **DEVPATH** | Сопоставляется с конкретным путем устройства |
| **KERNEL**[a] | Сопоставляется с именем ядра для устройства |
| **SUBSYSTEM**[a] | Сопоставляется с конкретной подсистемой |
| **DRIVER**[a] | Сопоставляется с драйвером, используемым устройством |
| **ATTR{***имя файла***}**[a] | Сопоставляется с **sysfs**-значениями устройства; *имя файла* — это листок в дереве **sysfs**, который соответствует конкретному атрибуту |
| **ENV{***ключ***}** | Сопоставляется со значением переменной среды |
| **TEST{***omask***}** | Проверяет факт существования файла; элемент *omask* необязателен |
| **PROGRAM** | Выполняет внешнюю команду; условие считается выполненным, если код завершения команды равен 0 |
| **RESULT** | Сопоставляется с результатом последнего вызова посредством ключа **PROGRAM** |

[a] Возможна также смешанная версия. Выполняется поиск пути устройства, который совпадает с заданным значением.

В правилах сопоставления элементы *назначение* задают действия, которые должен предпринять демон **udevd**, чтобы обработать событие. Формат этих элементов подобен формату элементов *условие*.

Самым важным ключом элемента *назначение* является ключ **NAME**, который означает, каким именем менеджер udev должен назвать устройство. Необязательный ключ **SYMLINK** создает символическую ссылку на устройство через путь, заданный в каталоге **/dev**.

Рассмотрим совместную работу этих компонентов на примере флеш-памяти USB. Предположим, что мы хотим сделать имя этого устройства постоянным и обеспечить автоматическое его монтирование и размонтирование.

Вставим нашу флеш-память и посмотрим, как ядро может идентифицировать ее. Возможны два пути. Выполнив команду **lsusb**, мы можем исследовать шину USB напрямую.

```
ubuntu$ lsusb
Bus 001 Device 007: ID 1307:0163 Transcend, Inc. USB Flash Drive
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

В качестве альтернативного варианта можем проверить записи в журнале, представленные в файле **/var/log/messages**. В нашем случае устройство оставляет пространную контрольную запись.

```
Aug  9 19:50:03 ubuntu kernel: [42689.253554] scsi 8:0:0:0: Direct-Access
   Ut163   USB2FlashStorage 0.00 PQ: 0 ANSI: 2
Aug  9 19:50:03 ubuntu kernel: [42689.292226] sd 8:0:0:0: [sdb] 1974271 512-
   byte hardware sectors: (1.01 GB/963 MiB)
```

```
...
Aug  9 19:50:03 ubuntu kernel: [42689.304749] sd 8:0:0:0: [sdb] 1974271 512-
   byte hardware sectors: (1.01 GB/963 MiB)
Aug  9 19:50:03 ubuntu kernel: [42689.307182]  sdb: sdb1
Aug  9 19:50:03 ubuntu kernel: [42689.427785] sd 8:0:0:0: [sdb] Attached SCSI
   removable disk
Aug  9 19:50:03 ubuntu kernel: [42689.428405]  sd 8:0:0:0: Attached scsi
   generic sg3 type 0
```

Представленные выше зарегистрированные сообщения означают, что устройство было распознано как **sdb**, и это позволяет нам просто идентифицировать устройство в файловой системе **/sys**. Теперь мы можем просмотреть файловую систему **/sys** с помощью команды **udevadm** и поискать фрагменты правил, которые служат характеристикой устройства и могут использоваться в правилах менеджера udev.

```
ubuntu$ udevadm info -a -p /block/sdb/sdb1
looking at device '/devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/
   1-1:1.0/host30/target30:0:0/30:0:0:0/block/sdb/sdb1':
     KERNEL=="sdb1"
     SUBSYSTEM=="block"
     DRIVER==""
     ATTR{partition}=="1"
     ATTR{start}=="63"
     ATTR{size}=="1974208"
     ATTR{stat}==" 71 792 1857 808 0 0
   0 0 0 512 808"
looking at parent device '/devices/pci0000:00/0000:00:11.0/0000:02:03.0/
  usb1/1-1/1-1:1.0/host30/target30: 0:0/30:0:0:0/block/sdb':
    KERNELS=="sdb"
    SUBSYSTEMS=="block"
    DRIVERS==""
...
    ATTRS{scsi_level}=="3"
    ATTRS{vendor}=="Ut163  "
    ATTRS{model}=="USB2FlashStorage"
...
```

Результат выполнения команды **udevadm** отображает возможности совпадений с заданными значениями. Одна из них — поле **size**, которое, вероятно, является уникальным для данного устройства. Но если бы размер раздела изменился, устройство не было бы распознано. Мы можем также использовать сочетание двух значений: соглашение ядра о назначении имен в виде **sd** с дополнительной буквой и содержимое атрибута модели **USB2FlashStorage**.

При создании правил для данного конкретного устройства флеш-памяти еще одним вариантом мог бы послужить регистрационный (серийный) номер устройства (здесь он не отображен).

Поместим наши правила для этого устройства в файл **/etc/udev/rules.d/10-local.rules**. Поскольку мы собираемся "убить сразу несколько зайцев", нужно создать ряд правил.

Прежде всего, мы должны позаботиться о создании символических ссылок на устройство в каталоге **/dev**. Следующее правило для идентификации устройства использует наши знания о ключах условий **ATTRS** и **KERNEL**, почерпнутых из команды **udevadm**.

```
ATTRS{model}=="USB2FlashStorage", KERNEL=="sd[a-z]1", SYMLINK+="ate-
   flash%n"
```

Когда наше правило срабатывает, демон **udevd** устанавливает значение **/dev/ate-flashN** в качестве символической ссылки на устройство. В действительности мы не ожидаем в системе появления более одного такого устройства. Если обнаружится несколько копий устройства, они получат уникальные имена в каталоге **/dev**, но точные имена зависят от порядка вставки устройств в систему.

Затем мы используем ключ **ACTION** для выполнения ряда команд всякий раз, когда устройство будет появляться на шине USB. Ключ элемента *назначение* **RUN** позволяет создавать соответствующий каталог точки монтирования и монтировать там наше устройство.

```
ACTION=="add", ATTRS{model}=="USB2FlashStorage", KERNEL=="sd[a-z]1",
  RUN+="/bin/mkdir -p /mnt/ate-flash%n"
ACTION=="add", ATTRS{model}=="USB2FlashStorage", KERNEL=="sd[a-z]1",
  PROGRAM=="/lib/udev/vol_id -t %N", RESULT=="vfat", RUN+="/bin/mount
  -t vfat /dev/%k /mnt/ate-flash%n"
```

Ключи **PROGRAM** и **RUN** выглядят похожими, но напомним, что **PROGRAM** — это ключ условия, который активен во время фазы выбора правила, тогда как **RUN** — это ключ назначения, который является частью действий, инициированных правилом. Второе правило с помощью опции **-t vfat** в команде **mount** еще до операции монтирования проверяет, что флеш-память содержит файловую систему Windows.

Аналогичные правила после удаления устройства из системы позволяют выполнить очистительно-восстановительные операции.

```
ACTION=="remove", ATTRS{model}=="USB2FlashStorage", KERNEL=="sd[a-z]1",
  RUN+="/bin/umount -l /mnt/ate-flash%n"
ACTION=="remove", ATTRS{model}=="USB2FlashStorage", KERNEL=="sd[a-z]1",
  RUN+="/bin/rmdir /mnt/ate-flash%n"
```

Теперь, когда правила подготовлены, мы должны уведомить демон **udevd** о наших изменениях. Команда **control**, используемая в качестве аргумента команды **udevadm**, — одна из немногих, которые требуют полномочий суперпользователя.

```
ubuntu$ sudo udevadm control --reload-rules
```

Опечатки молча игнорируются после перезагрузки, даже с использованием флага **--debug**, поэтому не забудьте дважды проверить синтаксис своих правил.

Вот и все! Теперь, когда флеш-память вставляется в разъем USB-порта, демон **udevd** создает символическую ссылку **/dev/ate-flash1** и монтирует устройство **/mnt/ate-flash1**.

```
ubuntu$ ls -l /dev/ate*
lrwxrwxrwx 1 root root 4 2009-08-09 21:22 /dev/ate-flash1 -> sdb1

ubuntu$ mount | grep at
/dev/sdb1 on /mnt/ate-flash1 type vfat (rw)
```

## 13.9. Рекомендуемая литература


- Bovet, Daniel P. and Marco Cesati. *Understanding the Linux Kernel (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2006.
- Corbet, Jonathan, et al. *Linux Device Drivers (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2005. Эта книга доступна также по адресу: **lwn.net/Kernel/LDD3**.

- Love, Robert. *Linux Kernel Development (2nd Edition)*. Indianapolis, IN: Novell Press, 2005. (Роберт Лав. Разработка ядра Linux, 2-е издание. ИД "Вильямс", 2006 г.)
- McDougall, Richard and Jim Mauro. *Solaris Internals: Solaris 10 and Open-Solaris Kernel Architecture (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2006.


## 13.10. Упражнения

13.1. Опишите, что делает ядро. Объясните разницу между загрузкой драйвера в качестве модуля и его статической компиляцией в ядро.

13.2. Процесс в системе HP-UX завершился аварийным отказом и выдал странное сообщение об ошибке: "слишком много открытых файлов: права доступа к файлам отрицают доступ сервера". Что могло стать причиной такой ошибки? Какие изменения необходимо внести, чтобы решить проблему?

13.3. ★ Предлагают ли системы AIX загружаемые модули ядра? Как разработчик должен добавить в ядро AIX поддержку новой файловой системы или новых системных вызовов? Когда могут потребоваться такие функции?

13.4. ★ Вы недорого купили Ethernet-плату для портативного компьютера, которая позволяет подключаться к сети через параллельный порт. Какие действия необходимо предпринять, чтобы операционная система Linux распознала новую плату? Нужно ли встроить поддержку платы непосредственно в ядро или ее следует реализовать в виде модуля? Почему? (Вопрос на засыпку: если ваша почасовая ставка составляет $80, во сколько обойдется установка этой "дешевой" платы?)

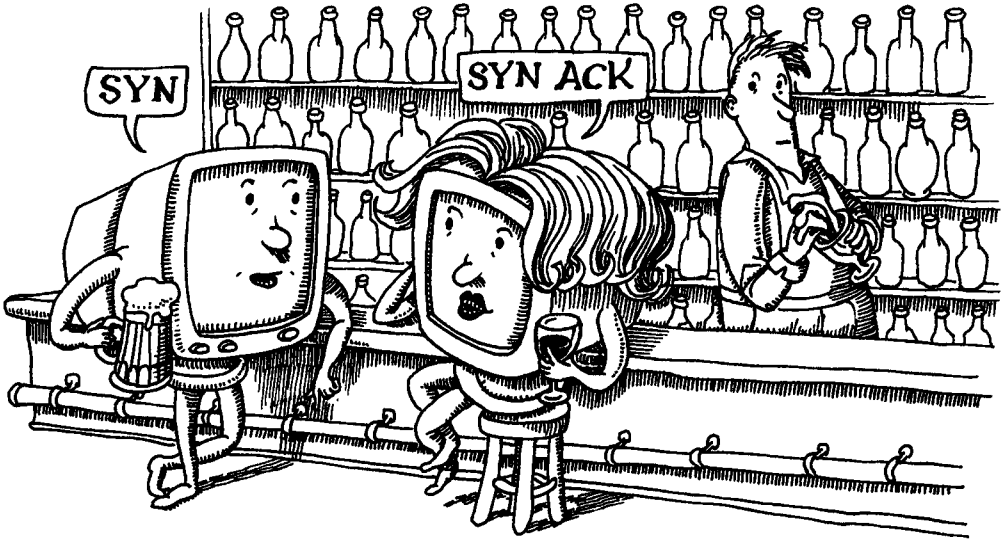
13.5. ★★ Настройте ядро с помощью команды **`make xconfig`** или **`make menuconfig`** и создайте исполняемый образ ядра. Инсталлируйте ядро и перезагрузите систему. Включите выполнение команды **`dmesg`** на этапе инициализации системы и сравните результаты, выдаваемые старым и новым ядрами. (Необходим доступ с правами суперпользователя.)

# ЧАСТЬ II

# Работа в сети

# Глава 14

# Сети TCP/IP




Трудно переоценить важность сетей в современном компьютерном мире, хотя многие все же пытаются это сделать. Во многих организациях — возможно, даже в большинстве из них — компьютеры используются, в первую очередь, для работы в веб и доступа к электронной почте. По оценкам сайта `internetworldstat.com`, к средине 2010 года в Интернете работало более 1,5 млрд пользователей, что составляет более 21% населения Земли. В Северной Америке внедрение Интернета уже достигло 75%.

TCP/IP — это сетевая система (networking system), лежащая в основе Интернета. Система TCP/IP не зависит ни от аппаратного обеспечения, ни от операционной системы, поэтому все устройства, использующие протоколы TCP/IP, могут обмениваться данными ("взаимодействовать"), несмотря на различия.

Система TCP/IP работает в сетях любого размера и любой топологии, независимо от того, соединены они с внешним миром или нет. В этой главе протоколы TCP/IP рассматриваются в политическом и техническом аспектах, связанных с Интернетом, но изолированные сети на уровне протоколов TCP/IP очень похожи друг на друга.

## 14.1. Система TCP/IP и Интернет

История системы TCP/IP тесно связана с историей Интернета и уходит корнями на несколько десятилетий назад. Популярность Интернета во многом обусловлена элегантностью и гибкостью системы TCP/IP, а также тем, что это открытое и некоммерческое семейство протоколов. В свою очередь, широкое распространение системы TCP/IP именно в Интернете позволило этому семейству одержать верх над несколькими конкурирующими семействами, популярными в свое время по политическим или коммерческим причинам.

Прародительницей Интернета была сеть ARPANET, организованная в 1969 году Министерством обороны США. К концу 1980-х годов эта сеть перестала быть научно-исследовательской и наступила эра коммерческого Интернета. В настоящее время Интернет представляет собой совокупность частных сетей, принадлежащих провайдерам интернет-услуг (ISP — internet service provider) и соединяющихся в так называемых "точках обмена трафиком" (peering center).

## Кто сегодня управляет Интернетом

Проектирование Интернета и его протоколов всегда осуществлялось на кооперативных и открытых началах, но его точная структура изменялась по мере того, как Интернет превращался в инструмент открытого пользования и движущую силу мировой экономики. В настоящее время управление Интернетом разделено на административное, техническое и политическое, но границы между этими функциями часто размыты. Основными действующими лицами в управлении Интернетом являются следующие организации.

- ICANN (Internet Corporation for Assigned Names and Numbers — Корпорация по назначению имен и адресов в Интернете). Эта группа с наибольшим правом может быть названа ответственной за Интернет. Только она обладает всеми реальными полномочиями. Корпорация ICANN контролирует выделение адресов и доменных имен в Интернете, а также другие аспекты его деятельности, например номера протокольных портов. Эта организация, основанная как некоммерческая корпорация, расположена в Калифорнии и действует в рамках меморандума по взаимопонимании с Министерством торговли США (`www.icann.org`).
- ISOC (Internet Society — Сообщество пользователей Интернета) — открытая членская организация, представляющая интересы пользователей Интернета. Несмотря на то что эта организация преследует образовательные и политические цели, она широко известна как прикрытие для технического развития Интернет. В частности, организация ISOC является головной организацией по отношению к проблемной группе проектирования Интернета (`ietf.org`), контролирующей всю техническую работу. ISOC является международной некоммерческой организацией. Ее офисы расположены в Вашингтоне, округ Колумбия, и Женеве (`www.isoc.org`).
- IETF (Internet Engineering Task Force — Проблемная группа проектирования Интернета) — эта организация отвечает за разработку и публикацию технических стандартов Интернета, являясь открытым форумом, принять участие в котором может любой желающий (`www.ietf.org`).
- IGF (Internet Governance Forum) — это организация, созданная Организацией Объединенных Наций относительно недавно, в 2005 году, чтобы предоставить возможность для проведения международных и политических дискуссий, посвященных Интернету. Она проводит ежегодные конференции, но ее роль со временем постепенно возрастает, поскольку правительства разных стран пытаются установить более жесткий контроль за Интернетом (`intgovforum.org`).

Среди перечисленных организаций наибольшая ответственность лежит на ICANN: она выступает в качестве руководящего органа Интернета, исправляя ошибки, допущенные в прошлом, и продумывая пути дальнейшего развития Интернета с учетом потребностей пользователей, правительств и бизнеса.

## Сетевые стандарты и документация

Если вы не уснули сразу после того, как прочитали название этого раздела, значит, выпили несколько чашек кофе. Тем не менее, умение разбираться в технической документации, выпускаемой руководящими органами Интернета, чрезвычайно важно для сетевых администраторов, и при этом это не так уж и скучно, как кажется на первый взгляд.

Техническая деятельность сообщества пользователей Интернета находит отражение в серии документов, известных как RFC (Request For Comments — запрос комментариев). В формате документов RFC публикуются стандарты протоколов, предлагаемые нововведения, а также информационные бюллетени. Получая сначала статус черновых проектов (Internet Draft), после бурных дебатов в телеконференциях и на форумах IETF они либо отвергаются, либо получают официальный статус. Свое мнение по поводу предлагаемого стандарта может высказать любой участник обсуждения. Иногда документ RFC представляет собой не стандартизацию протоколов Интернета, а всего лишь констатацию или объяснение некоторых аспектов современной практики.

Документы RFC имеют порядковые номера. На сегодняшний день их более 5600. У каждого документа есть описательное название (например, *Algorithms for Synchronizing Network Clocks* — алгоритмы сетевой синхронизации часов), но, во избежание неоднозначности, на документы чаще всего ссылаются по номерам. Будучи опубликованным, документ RFC никогда не меняется. Изменения и дополнения публикуются в виде новых документов с собственными номерами. Обновление может либо дополнять и разъяснять документ RFC, либо полностью его заменять.

Существует множество источников, распространяющих документы RFC, но сайт `tfc-editor.org` является центральным диспетчерским пунктом и всегда содержит самую свежую информацию. Прежде чем тратить время на чтение документа RFC, выясните его статус на сайте `tfc-editor.org`; возможно, этот документ уже не содержит наиболее актуальную информацию.

Процесс публикации стандартов Интернета подробно описан в документе RFC2026. Другой полезный метадокумент — RFC5540, *40 Years of RFCs* (40 лет существования документов RFC). В нем описаны культурные и технические аспекты публикации документов RFC.

Пусть читателей не пугает обилие технических подробностей в документах RFC. В большинстве из них содержатся вводные описания, резюме и толкования, которые будут полезны системным администраторам. Некоторые документы специально написаны в виде обзора или общего введения. Чтение документов RFC — это, возможно, не самый простой способ разобраться в той или иной теме, но это авторитетный, лаконичный и бесплатный источник информации.

Не все документы RFC написаны сухим техническим языком. Встречаются документы развлекательного содержания (часть из них опубликована первого апреля), среди которых нам особенно нравятся следующие:

- RFC1149 — *A Standard for the Transmission of IP Datagrams on Avian Carriers* (стандарт передачи датаграмм с помощью птиц)[1];
- RFC1925 — *The Twelve Networking Truths* (12 сетевых истин);

[1] Группа энтузиастов системы Linux из города Берген в Норвегии, называющая себя BLUG (Bergen Linux User Group), действительно реализовала протокол CPIP (Carrier Pigeon Internet Protocol — межсетевой протокол голубиной почты) в соответствии с документом RFC1149. Детали см. на их веб-сайте по адресу: `blug.linux.no/rfc1149`.

- RFC3251 — *Electricity over IP* (передача электричества по протоколу IP);
- RFC3514 — *The Security Flag in the IPv4 Header* (защитный флажок в заголовке протокола IPv4).
- RFC4041 — *Requirements for Morality Section in Routing Area Drafts* (моральный кодекс маршрутизатора).

Помимо собственных порядковых номеров, документам RFC могут назначаться номера серий FYI (For Your Information — к вашему сведению), BCP (Best Current Practice — лучший существующий подход) и STD (Standard — стандарт). Перечисленные серии являются подмножествами серии RFC, группирующими документы по важности или назначению.

Документы FYI — это вводные или информационные материалы, предназначенные для широкой аудитории. Как правило, именно с них лучше всего начинать изучать незнакомую тему. К сожалению, эта серия в последнее время стала иссякать и сейчас современных документов FYI очень немного.

Документы BCP описывают рекомендуемые процедуры для администраторов вебсайтов в Интернете. Они содержат административные предписания и представляют большую ценность для системных администраторов.

Документы STD содержат описания протоколов Интернета, прошедших процедуру проверки и тестирования в IETF и формально принятых в качестве стандартов.

Нумерация документов в рамках серий RFC, FYI, STD и BCP ведется раздельно, поэтому один документ может иметь несколько номеров. Например, документ RFC1713, *Tools for DNS Debugging* (инструменты для отладки системы DNS) известен также под номером FYI127.

## 14.2. Дорожная карта сети

Завершив краткий обзор, давайте подробнее рассмотрим, что собой представляют протоколы TCP/IP. TCP/IP — это семейство сетевых протоколов, ориентированных на совместную работу. В состав семейства входит несколько компонентов:

- IP (Internet Protocol — межсетевой протокол) обеспечивает передачу пакетов данных с одного компьютера на другой (RFC791);
- ICMP (Internet Control Message Protocol — протокол управляющих сообщений в Интернете) отвечает за различные виды низкоуровневой поддержки протокола IP, включая сообщения об ошибках, вспомогательные маршрутизирующие запросы и отладочные сообщения (RFC792);
- ARP (Address Resolution Protocol — протокол преобразования адресов) обеспечивает трансляцию IP-адресов в аппаратные адреса (RFC826)[2];
- UDP (User Datagram Protocol — протокол передачи датаграмм пользователя) обеспечивает непроверяемую одностороннюю доставку данных (RFC768);
- TCP (Transmission Control Protocol — протокол управления передачей) обеспечивает надежный дуплексный канал связи между процессами на двух компьютерах с возможностью управления потоками и контроля ошибок (RFC793).

[2] Мы немного грешим против истины, утверждая, что протокол ARP входит в семейство TCP/IP. На самом деле он вполне может использоваться вместе с другими семействами протоколов. Просто этот протокол является неотъемлемой частью стека TCP/IP в большинстве локальных сетей.

Эти протоколы образуют иерархию, или стек, в котором протокол верхнего уровня использует протокол нижележащего уровня. Систему TCP/IP обычно описывают в виде пятиуровневой структуры (рис. 14.1), но реальная система TCP/IP содержит только три из этих уровней.

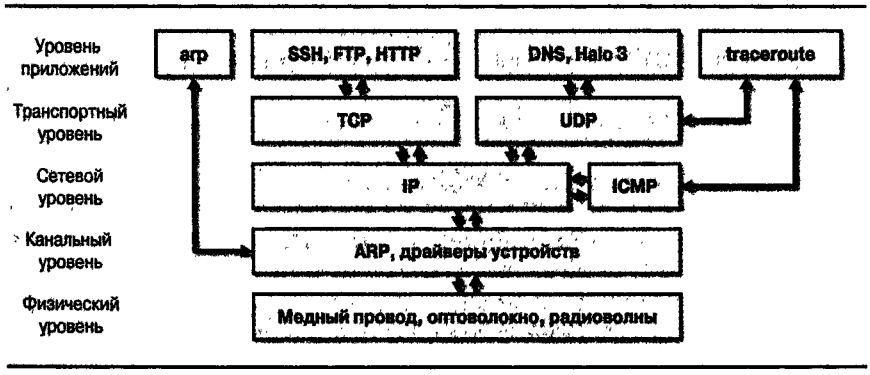



*Рис. 14.1. Семейство TCP/IP*

## Версии IPv4 и IPv6

В течение последних трех десятилетий широкое распространение получила четвертая версия протокола TCP/IP, известная также под именем IPv4. Она использует четырехбайтовые IP-адреса. Современная версия, IPv6, расширила адресное пространство до 16 байт, а также учла опыт использования версии IPv4. В нее не были включены возможности протокола IPv4, которые оказались не очень нужными. Это позволило ускорить работу протокола и облегчило его реализацию. Кроме того, версия IPv6 объединила системы безопасности и аутентификации в рамках одного базового протокола.

Все современные операционные системы и многие сетевые устройства уже поддерживают протокол IPv6. Однако в реальности степень активного использования протокола IPv6 остается практически нулевой.[3] Опыт показывает, что администраторам лучше было бы отложить использование протокола IPv6 на как можно более долгий срок. В конце концов все будут вынуждены перейти на протокол IPv6, но на это потребуется еще несколько лет. В то же время этот переход не настолько далек, чтобы не учитывать его при покупке новых сетевых устройств. Приобретая новое оборудование, настаивайте на том, чтобы оно поддерживало протокол IPv6.

Разработка протокола IPv6 была в большой степени мотивирована беспокойством о том, что адресное пространство протокола IPv4 практически исчерпано. Это действительно так: прогнозы показывают, что современная система выделения адресов в протоколе IPv4 примерно в 2011 году испытает коллапс. (См. сайт `ipv4.potaroo.net`, который обновляется ежедневно.) И все же адаптация протокола IPv6 в Интернете в ближайшее время не предвидится.

Более вероятно, что организации ISP и ICANN (точнее, их филиал IANA (Internet Assigned Numbers Authority — Администрация адресного пространства Интернета)) предпримут временные меры по усилению господства протокола Iv4 в течение несколь-

[3] Компания Google на конференции RIPE 57, которая проходила в октябре 2008 года, отметила, что степень внедрения версии IPv6 (не возможности ее поддержки, а реального использования) равна 0,24%. Нет ни одной страны, в которой степень внедрения версии IPv6 превысила бы 0,76%.

ких следующих лет. Мы ожидаем увеличения степени использования протокола IPv6 в Интернете, но, за исключением крупных провайдеров интернет-услуг, академических сайтов и универсальных провайдеров вроде компании Google, протокол IPv6 в ближайшем будущем не повлияет на работу большинства системных администраторов.

Дефицит адресов в рамках протокола IPv4 особенно остро ощущается за пределами США, поэтому там протокол IPv6 ожидает более теплый прием. В США резкому росту внедрения протокола IPv6 может способствовать потрясающее приложение: новое поколение мобильных телефонов, отображающих адрес IPv6 в телефонный номер. (Системы голосовой связи через Интернет также получат выгоду от более тесного соответствия телефонных номеров и адресов IPv6.)

В книге мы сосредоточились на протоколе IPv4, поскольку именно он является основной версией протокола TCP/IP. Все, что касается версии IPv6, мы выделяем явным образом. К счастью для системных администраторов, версии IPv4 и IPv6 очень похожи. Если вы знаете протокол IPv4, то вы знаете большую часть того, что вам следует знать о протоколе IPv6. Основное различие между этими версиями заключается в том, что они используют разные схемы адресации. В версии IPv6 использовано несколько новых концепций адресации и несколько новых обозначений. Вот и все.

## Пакеты и их инкапсуляция

Система TCP/IP располагает средствами поддержки целого ряда физических сетей и транспортных систем, включая технологии Ethernet, Token Ring, MPLS (Multiprotocol Label Switching), беспроводную технологию Ethernet, а также системы с последовательными соединениями. Управление аппаратными устройствами осуществляется на канальном уровне архитектуры TCP/IP, а протоколам более высоких уровней неизвестно, как именно используются аппаратные средства.

Данные передаются по сети в виде *пакетов*, которые имеют максимальный размер, определяемый ограничениями канального уровня. Каждый пакет состоит из заголовка и полезного содержимого. Заголовок содержит сведения о том, откуда прибыл пакет и куда он направляется. В него могут также включаться контрольные суммы, информация, характерная для конкретного протокола, и другие инструкции, касающиеся обработки пакета. Полезное содержимое — это данные, подлежащие пересылке.

Название базового блока передачи данных зависит от уровня протокола. На канальном уровне это *кадр*, или *фрейм*, в протоколе IP — *пакет*, а в протоколе TCP — *сегмент*. Мы будем придерживаться универсального термина "пакет".

Готовящийся к отправке пакет передается вниз по стеку протоколов, и каждый протокол добавляет в него собственный заголовок. Сформированный пакет одного протокола становится полезным содержимым пакета, генерируемого следующим протоколом. Эта операция называется *инкапсуляцией*. На принимающей стороне инкапсулированные пакеты восстанавливаются в обратном порядке при прохождении вверх по стеку.

Например, датаграмма (пакет UDP), передаваемая по сети Ethernet, упакована в трех различных "конвертах". В среде Ethernet она "вкладывается" в простой физический фрейм, заголовок которого содержит сведения об аппаратных адресах отправителя и ближайшего получателя, длине фрейма и его контрольной сумме (CRC). Полезным содержимым Ethernet-фрейма является IP-пакет. Полезное содержимое IP-пакета — это UDP-пакет, и, наконец, полезное содержимое UDP-пакета состоит из собственно данных, подлежащих передаче. Компоненты такого пакета изображены на рис. 14.2.

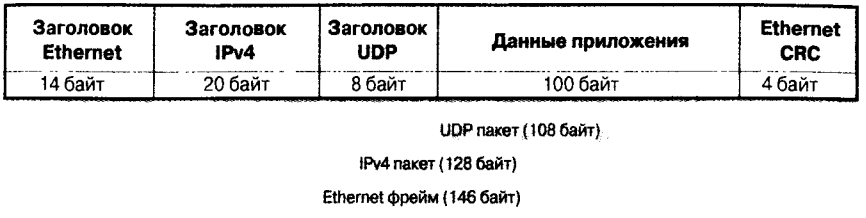



*Рис. 14.2. Стандартный сетевой пакет*[4]

## Стандарты формирования фреймов Ethernet

На канальном уровне добавляются заголовки к пакетам и вставляются разделители между ними. Заголовки содержат информацию об адресах канального уровня и контрольные суммы, а разделители позволяют принимающей стороне понять, где заканчивается один пакет и начинается другой. Процесс добавления вспомогательных битов называется формированием фреймов или кадровой разбивкой.

На самом деле канальный уровень разделен на две части: MAC (подуровень Media Access Control) и LLC (подуровень Link Layer Control). Подуровень MAC работает с аудиовизуальной информацией и передает пакеты по проводам. Подуровень LLC формирует фреймы.

В настоящее время существует единственный стандарт формирования фреймов по технологии Ethernet: DIX Ethernet II. Кроме того, по историческим причинам используется еще несколько стандартов, основанных на стандарте IEEE 802.2, особенно в сетях Novell.

### *Максимальный размер передаваемого блока*

Размер сетевых пакетов ограничивается как характеристиками аппаратных средств, так и требованиями протоколов. Например, объем полезного содержимого стандартного Ethernet-фрейма не может превышать 1500 байт. Предельный размер пакета устанавливается на канальном уровне и называется максимальной единицей передачи (Maximum Transfer Unit, MTU). Стандартные значения параметра MTU для разных видов сетей приведены в табл. 14.1.

**Таблица 14.1. Максимальные размеры передаваемых блоков в сетях различных типов**

| Тип сетевого соединения | Максимальная единица передачи |
|---|---|
| Ethernet | 1500 байт (1492 в спецификации 802.2) |
| FDDI | 4470 байт (4352 для IP/FDDI) |
| Token Ring | Конфигурируется [a] |
| Модемное PPP-соединение | Конфигурируется, обычно 512 или 576 байт |
| Двухточечные каналы ГВС (T1, T3) | Конфигурируется, обычно 1500 или 4500 байт |

[a] Распространенные значения таковы: 552, 1064, 2088, 4508 и 8232. Иногда выбирается значение 1500 для совместимости с Ethernet.

[4] К примеру, в документах RFC, описывающих протоколы, часто используется термин "октет" (octet) вместо "байт" (byte).

В семействе TCP/IP протокол IP отвечает за разбивку пакета на такие фрагменты, чтобы их размер соответствовал требованиям конкретного сетевого соединения. Если пакет проходит через несколько сетей, в одной из них параметр MTU может оказаться меньшим, чем в исходной сети. В этом случае маршрутизатор, пересылающий пакет в сеть с меньшим значением MTU, подвергнет пакет дальнейшей фрагментации.

Фрагментация "на лету" нежелательна в условиях сильной загруженности маршрутизаторов. В протоколе IPv6 эта возможность устранена. Пакеты по-прежнему могут фрагментироваться, но эту задачу должен выполнять вызывающий хост.

Отправитель может обнаружить канал, способный пропустить пакет с наименьшим показателем MTU, установив на пакете флаг "не фрагментировать". Если пакет достигнет промежуточного маршрутизатора, который не способен его пропустить, этот маршрутизатор вернет отправителю сообщение об ошибке ICMP. Пакет ICMP содержит показатель MTU сети, требующей пакеты меньшего размера, и этот показатель затем становится ориентировочным размером для пакетов, отправляемых по данному адресу.

В протоколе TCP путь MTU определяется автоматически, даже в версии IPv4. Протокол UDP такой возможности не имеет и перекладывает дополнительную работу на уровень IP.

Иногда проблема фрагментации оказывается достаточно коварной. Несмотря на то что выявление пути MTU должно автоматически разрешить конфликты, иногда администратор должен вмешаться. Например, в виртуальной частной сети с туннельной структурой необходимо проверять размер пакетов, проходящих через туннель. Обычно их начальный размер — 1500 байт, но когда к ним добавляется туннельный заголовок, размер пакетов становится равным примерно 1540 байт и уже требуется фрагментация. Уменьшение максимального размера передаваемого блока позволяет избежать фрагментации и повысить производительность туннельной сети. Просмотрите **справочную** страницу команды `ifconfig`, чтобы узнать, как настроить параметр MTU сетевой платы.

# 14.3. Адресация пакетов

Подобно письмам и сообщениям электронной почты, сетевые пакеты могут достичь пункта назначения только при наличии правильного адреса. В системе TCP/IP используется сочетание нескольких схем адресации.

- Адреса MAC (media access control) для использования в сетевом оборудовании.
- Сетевые адреса протоколов IPv4 и IPv6 для использования в программном обеспечении.
- Имена компьютеров для использования пользователями.

## Аппаратная адресация (MAC)

Каждый сетевой интерфейс компьютера имеет один MAC-адрес канального уровня, который отличает его от других компьютеров в физической сети, а также один или несколько IP-адресов, идентифицирующих интерфейс в глобальной сети Интернета. Последнее утверждение стоит повторить: IP-адрес идентифицирует *сетевые интерфейсы, а не машины.* (Для пользователей это различие не имеет значения, но администраторы должны об этом знать.)

Самый нижний уровень адресации задается сетевыми аппаратными средствами. Например, Ethernet-устройствам в процессе изготовления назначаются уникальные

шестибайтовые аппаратные адреса. Эти адреса традиционно записываются в виде ряда двухцифровых шестнадцатеричных байтов, разделенных двоеточиями, например 00:50:8D:9A:3B:DF.

Сетевые платы Token Ring также имеют шестибайтовые адреса. В некоторых сетях с двухточечным соединением (например, в PPP-сетях) аппаратные адреса вообще не нужны: адрес пункта назначения указывается непосредственно при установке соединения.

Шестибайтовый Ethernet-адрес разбивается на две части: первые три байта определяют изготовителя устройства, а последние три — выступают в качестве уникального серийного номера, назначаемого изготовителем. Системные администраторы могут выяснить марку устройства, вызывающего проблемы в сети, поискав трехбайтовый идентификатор соответствующих пакетов в таблице идентификаторов изготовителей. Текущая таблица доступна по адресу

```
http://www.iana.org/assignments/ethernet-numbers.
```

Трехбайтовые коды на самом деле представляют собой идентификаторы OUI (Organizationally Unique Identifier — уникальный идентификатор организации), присваиваемые организацией IEEE, поэтому их можно найти непосредственно в базе данных IEEE по адресу

```
http://standards.ieee.org/regauth/oui.
```

Разумеется, отношения между производителями микросхем, компонентов и системы носят сложный характер, поэтому идентификатор изготовителя, закодированный в MAC-адресе, может ввести пользователя в заблуждение.

Теоретически, аппаратные адреса Ethernet должны назначаться на постоянной основе и оставаться неизменными. К сожалению, некоторые сетевые платы допускают программное задание аппаратных адресов. Это удобно при замене испорченных компьютеров или сетевых карт, MAC-адрес которых менять по тем или иным причинам нежелательно (например, если его фильтруют все ваши коммутаторы, если ваш DHCP-сервер выдает адреса на основе MAC-адресов или MAC-адрес был использован как лицензионный ключ для программного обеспечения). Фальсифицируемые MAC-адреса могут также оказаться полезными, если вам необходимо проникнуть в беспроводную сеть, использующую механизм управления доступом на основе MAC-адресов. Однако, чтобы не усложнять ситуацию, мы рекомендуем сохранять уникальность MAC-адресов.

## IP-адресация

На следующем, более высоком, уровне используется Интернет-адресация (чаще называемая IP-адресацией). IP-адреса глобально уникальны[5] и аппаратно независимы.

Соответствие между IP-адресами и аппаратными адресами устанавливается на канальном уровне модели TCP/IP. В сетях, поддерживающих широковещательный режим (т.е. в сетях, позволяющих адресовать пакеты "всем компьютерам данного физического сегмента"), протокол ARP обеспечивает автоматическую привязку адресов без вмешательства системного администратора. В протоколе IPv6 MAC-адреса интерфейсов мож-

[5] В принципе IP-адреса идентифицируют конкретный и уникальный пункт назначения. Однако в особых случаях ситуация усложняется. Механизм NAT (Network Adrresses Translation — преобразование сетевых адресов) использует IP-адреса интерфейсов для того, чтобы обработать трафик на нескольких машинах. Пространство частных IP-адресов присваивает адреса нескольким сайтам, которые могут использовать их одновременно, поскольку они не видимы в Интернете. Адресация в методе Anycast распределяет один и тот же адрес среди нескольких машин.

но использовать как часть IP-адресов, благодаря чему преобразование IP-адресов в аппаратные адреса становится практически автоматическим.

🕮 Подробнее о протоколе ARP рассказывается в разделе 14.6.

## "Адресация" имен машин

Поскольку IP-адреса представляют собой длинные, на первый взгляд, случайные числа, запоминать их трудно. Операционные системы позволяют закреплять за IP-адресом одно или несколько текстовых имен, чтобы вместо 128.9.160.27 пользователь мог ввести "rfc-editor.org". В системах UNIX и Linux это отображение можно осуществить с помощью статического файла (`/etc/hosts`), базы данных LDAP и, наконец, DNS (Domain Name System) — глобальной системы доменных имен. Следует помнить о том, что имя компьютера — это лишь сокращенный способ записи IP-адреса, и он относится к сетевому интерфейсу, а не компьютеру.

🕮 Подробнее о глобальной системе DNS рассказывается в главе 17.

## Порты

IP-адреса идентифицируют сетевые интерфейсы компьютера, но они недостаточно конкретны для идентификации отдельных процессов и служб, многие из которых могут активно использоваться в сети одновременно. Протоколы TCP и UDP расширяют концепцию IP-адресов, вводя понятие *порта*. Порт представляет собой 16-разрядное число, добавляемое к IP-адресу и указывающее конкретный канал взаимодействия. Всем стандартным службам, в частности электронной почте, FTP и HTTP, назначаются "хорошо известные" порты, которые определены в файле `/etc/services`.[6] Для того чтобы предотвратить попытки посторонних процессов замаскироваться под стандартные службы, системы UNIX предоставляют доступ к портам с номерами до 1024 только процессам пользователя `root`. (Взаимодействовать с сервером через порты с небольшими номерами может кто угодно; ограничение распространяется лишь на программы, прослушивающие этот порт.)

## Типы адресов

В протоколе IP поддерживается несколько типов адресов, некоторые из которых имеют эквиваленты на канальном уровне.

- Направленные (unicast) — адреса, которые обозначают отдельный сетевой интерфейс.
- Групповые (multicast) — адреса, идентифицирующие группу узлов.
- Широковещательные (broadcast) — адреса, обозначающие все узлы локальной сети.
- Альтернативные (anycast) — адреса, обозначающие любой из группы узлов.

Режим группового вещания используется, к примеру, в видеоконференциях, где одна и та же последовательность пакетов посылается всем участникам конференции. Протокол IGMP (Internet Group Management Protocol — протокол управления группами узлов Интернета) отвечает за управление совокупностями узлов, идентифицируемыми как один обобщенный адресат.

[6] Полный список присвоенных портов можно найти на веб-странице `iana.org/assignments/port-numbers`.

Групповые адреса в настоящее время в Интернете практически не используются. Тем не менее они были использованы в протоколе IPv6, в котором широковещательные адреса, по существу, представляют собой специализированную форму групповой адресации.

Альтернативные адреса обеспечивают балансирование нагрузки на канальный уровень сети, разрешая доставлять пакеты в ближайший из нескольких пунктов назначения в смысле сетевой маршрутизации. Можно было ожидать, что эти адреса будут напоминать групповые, но фактически они больше похожи на направленные адреса.

Большинство деталей механизма альтернативных адресов скрыто на уровне маршрутизации, а не на уровне протокола IP. Благодаря альтернативной адресации произошло реальное ослабление традиционных требований, чтобы IP-адреса однозначно идентифицировали пункт назначения. С формальной точки зрения, альтернативная реализация предназначена для протокола IPv6, но аналогичный трюк можно осуществить и в протоколе IPv4, например, так, как это сделано для корневых серверов имен DNS.

# 14.4. IP-адреса

За исключением групповых адресов, адреса Интернета состоят из двух частей: сетевой и машинной. Сетевая часть идентифицирует логическую сеть, к которой относится адрес, а машинная — узел этой сети. В протоколе IPv4 адреса состоят из четырех байтов, а граница между сетевой и машинной частями устанавливается административно. В протоколе IPv6 адреса состоят из 16 байт, а сетевая и машинная части всегда состоят из 8 байт.

В протоколе IPv4 адреса записываются в виде группы десятичных чисел (по одному на каждый байт), разделенных точками, например 209.85.171.147. Самый левый байт — старший; он всегда относится к сетевой части адреса.

Если первым байтом адреса является число 127, то оно обозначает *интерфейс обратной связи* ("loopback network") — фиктивную сеть, не имеющую реального аппаратного интерфейса и состоящую из одного компьютера. Адрес 127.0.0.1 всегда ссылается на текущий компьютер. Ему соответствует символическое имя "localhost". (Это еще одно небольшое нарушение требования уникальности IP-адресов, поскольку каждый компьютер интерпретирует адрес 127.0.0.1 как адрес другого компьютера, хотя этим компьютером является он сам.)

Адреса в протоколе IPv6 и их текстовые эквиваленты являются немного более сложными. Они будут рассмотрены далее в подразделе "Адресация в протоколе IPv6".

IP-адрес и другие параметры сетевого интерфейса задаются командой `ifconfig`. Она описывается в разделе 14.10.

## Классы адресов в протоколе IPv4

Исторически IP-адреса группировались в *классы*, которые определялись на основании первых битов самого левого байта. Классы отличались распределением байтов адреса между сетевой и машинной частями. Современные маршрутизаторы используют явные маски для задания сетевой части адреса, причем компоненты адреса могут разделяться не обязательно по границе байтов. Тем не менее традиционные классы все еще используются по умолчанию, если не предоставлена явная маска.

Классы A, B и C обозначают обычные IP-адреса, а классы D и E применяются при групповой адресации и в исследовательских целях. В табл. 14.2 представлены характеристики каждого класса адресов. Сетевая часть адреса помечена буквой C, а машинная — буквой M.

**Таблица 14.2. Классы IP-адресов**

| Класс | Первый байт[a] | Формат | Комментарии |
|---|---|---|---|
| A | 1–126 | C.M.M.M | Самые первые сети или адреса, зарезервированные для Министерства обороны США |
| B | 128–191 | C.C.M.M | Крупные организации, обычно с подсетями; адреса данного класса почти полностью заняты |
| C | 192–223 | C.C.C.M | Небольшие организации; адреса данного класса получить легко, они выделяются целыми блоками |
| D | 224–239 | — | Групповые адреса; не назначаются на постоянной основе |
| E | 240–255 | — | Экспериментальные адреса |

[a] Значение 0 не используется в качестве первого байта обычных IP-адресов. Значение 127 зарезервировано для адресов обратной связи.

В редких случаях в состав локальной сети входит более ста компьютеров. По этой причине полезность адресов класса A или B (которые допускают наличие в одной сети, соответственно, 16777214 и 65534 узлов) весьма сомнительна. К примеру, 127 адресов класса A занимают половину доступного адресного пространства. Кто же знал, что адресное пространство протокола IPv4 станет таким ценным!

## Подсети

Для того чтобы эффективнее использовать адреса, определенную долю машинной части адреса можно "одолжить" для расширения сетевой части, явно указав четырехбайтовую маску подсети ("subnet mask"), или сетевую маску ("netmask"), в которой единицы соответствуют сетевой части, а нули — машинной. Единицы должны занимать левую часть маски и следовать одна за другой без разрывов. Сетевая часть должна занимать не менее восьми битов, а машинная — не менее двух битов. Следовательно, маска подсети в соответствии с протоколом IPv6 допускает только 22 возможных значения.

Например, четыре байта адреса класса B обычно интерпретируются как C.C.M.M. Следовательно, неявная маска подсети класса B в десятичной системе выглядит как 255.255.0.0. Однако адрес с маской 255.255.0.0 можно интерпретировать как C.C.C.M. Использование такой маски превращает одну сеть класса B в 256 разных подсетей, подобных сетям класса C, т.е. в каждую из них может входить 254 компьютера.

Маски подсети, как и любой сетевой интерфейс, задаются с помощью команды **ifconfig**. По умолчанию команда **ifconfig** использует класс адреса для того, чтобы выяснить, какие биты относятся к сетевой части. Если задается явная маска, то эта функция просто отменяется.

🕮 Подробнее о команде **ifconfig** рассказывается в разделе 14.10.

Сетевые маски, не оканчивающиеся на границе байта, труднее декодировать. Они обычно записываются в виде суффикса */XX*, где *XX* — число битов в сетевой части адреса (длина маски). Такую запись иногда называют нотацией CIDR (Classless Inter-Domain Routing — протокол бесклассовой междоменной маршрутизации). Например, адрес 128.138.243.0/26 обозначает первую из четырех сетей с общим компонентом адреса 128.138.243. В трех других сетях последний байт адреса равен 64, 128 и 192. Сетевая маска, связанная с этими сетями, имеет вид 255.255.255.192 или 0xFFFFFFC0. В двоичном представлении это 26 единиц с последующими шестью нулями (рис. 14.3).

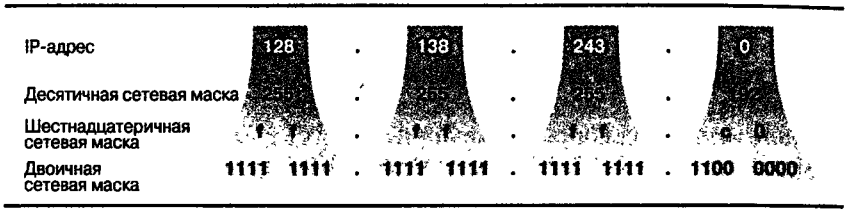



*Рис. 14.3. Структура маски подсети в разных системах счисления*

В сети с маской /26 для нумерации узлов отводится шесть битов (32 – 26 = 6). Таким образом, появляется возможность задать 64 адреса ($2^6$ = 64). В действительности допускается использовать лишь 62 адреса, поскольку адреса, полностью состоящие из нулей или единиц, зарезервированы (для самой сети и широковещательного режима соответственно).

В нашем примере старшие два бита последнего байта адреса могут принимать значения 00, 01, 10 и 11. Таким образом, сеть 128.138.243.0/24 может быть разделена на четыре сети /26:

- 128.138.243.0/26 (0 — **00**000000);
- 128.138.243.64/26 (64 — **01**000000);
- 128.138.243.128/26 (128 — **10**000000);
- 128.138.243.192/26 (192 — **11**000000).

Выделенные полужирным шрифтом биты последнего байта каждого адреса относятся к сетевой части.

## Трюки и инструменты для арифметических вычислений, связанных с подсетями

Манипулировать всеми этими битами в уме трудно, но есть ряд приемов, позволяющих упростить вычисления. Число узлов в сети и значение последнего байта сетевой маски в сумме всегда дают 256.

*последний байт сетевой маски* = 256 – *размер сети*

Так, в рассмотренном выше случае формула даст результат 256 – 64 = 192, где 192 — последний байт маски. Другое правило гласит о том, что значение последнего байта фактического адреса сети (не сетевой маски) должно нацело делиться на число узлов сети. В рассматриваемом примере последние байты равны 0, 64, 128 и 192 — каждое из этих чисел делится нацело на 64.[7]

Имея только IP-адрес (допустим, 128.138.243.100), невозможно сказать, каким будет адрес сети и широковещательный адрес. В табл. 14.3 представлены возможные варианты для масок /16 (выбирается по умолчанию для адресов класса B), /24 и /26 (наиболее приемлемая длина, если имеющееся адресное пространство невелико).

**Таблица 14.3. Пример расшифровки IP-адреса**

| IP-адрес | Сетевая маска | Адрес сети | Широковещательный адрес |
|---|---|---|---|
| 128.138.243.100/16 | 255.255.0.0 | 128.138.0.0 | 128.138.255.255 |
| 128.138.243.100/24 | 255.255.255.0 | 128.138.243.0 | 128.138.243.255 |
| 128.138.243.100/26 | 255.255.255.192 | 128.138.243.64 | 128.138.243.127 |

[7] Разумеется, нуль делится на любое число.

Сетевой (все нули в машинной части) и широковещательный (все единицы в машинной части) адреса сокращают число узлов в каждой локальной сети на 2, поэтому в самой маленькой сети формально должно быть 4 узла: два реальных, соединенных напрямую, и два фиктивных (сетевой и широковещательный). Для того чтобы создать такую сеть, необходимо отвести два бита под машинную часть адреса, т.е. суффикс адреса будет /30, а сетевая маска — 255.255.255.252 или 0xFFFFFFFC. Но есть еще сеть с маской /31, которая трактуется как особый случай (RFC3021): у нее нет сетевого и широковещательного адресов, а оба оставшихся адреса используются для идентификации узлов. Маска такой сети равна 255.255.255.254.

Кришан Джодис (Krischan Jodies) написал полезную программу под названием IP Calculator (она доступна по адресу `jodies.de/ipcalc`), позволяющую выполнять арифметические операции над двоичными, шестнадцатеричными числами и масками. Программа IP Calculator делает все, что требуется пользователю при работе с сетевыми адресами и масками подсети, а также широковещательными адресами и т.п. Кроме того, существует архив формата `tar` для версии калькулятора **ipcalc**, которая запускается из командной строки.

В операционной системе Ubuntu программу **ipcalc** можно установить с помощью команды **apt-get**.

Ниже приведен образец работы программы **ipcalc** (формат вывода для наглядности немного изменен).

```
Address:    24.8.175.69          00110000.00001000.10101111.01000101
Netmask:    255.255.255.0 = 24   11111111.11111111.11111111.00000000
Wildcard:   0.0.0.255            00000000.00000000.00000000.11111111
=>
Network:    24.8.175.0/24        00011000.00010000.10101111.00000000 (Class A)
Broadcast:  24.8.175.255         00011000.00010000.10101111.11111111
HostMin:    24.8.175.1           00011000.00010000.10101111.00000001
HostMax:    24.8.175.254         00011000.00010000.10101111.11111110
```

Эти результаты позволяют не только просто и понятно представить адреса, но и "копировать и вставлять" версии. Очень полезная программа.

В операционной системе Red Hat имеется утилита, тоже называющаяся **ipcalc**, но она довольно неуклюжа и в большинстве случаев предполагает, что IP-адреса относятся к классу, предусмотренному по умолчанию.

Если этот калькулятор вам по каким-то причинам не подойдет, можно воспользоваться стандартной утилитой **bc**, поскольку она может выполнять арифметические операции в любой системе счисления. Установите основания систем счисления для ввода и вывода, используя директивы **ibase** и **obase**. Сначала выполните директиву **obase**, в противном случае ее результат будет интерпретироваться в зависимости от нового основания системы счисления, установленной директивой **ibase**.

## CIDR: протокол бесклассовой междоменной маршрутизации

Как и подсети, прямым расширением которых он является, протокол CIDR основан на использовании явной маски подсети, определяющей границу между сетевой и машинной частями адреса. Но, в отличие от подсетей, протокол CIDR допускает, чтобы сетевая часть была *меньше*, чем того требует класс адреса. Благодаря укороченной маске возникает эффект объединения нескольких сетей для облегчения маршрутизации. По этой причине протокол CIDR иногда называют протоколом формирования суперсетей.

Протокол CIDR упрощает информацию о маршрутизации и устанавливает иерархию в этом процессе. Несмотря на то что протокол CIDR задумывался как временная мера для облегчения маршрутизации в рамках протокола IPv6, он оказался достаточно мощным средством для решения проблемы роста Интернета, возникшей в последнее десятилетие.

Предположим, организации предоставлен блок из восьми адресов класса C, пронумерованных последовательно от 192.144.0.0 до 192.144.7.0 (в нотации CIDR — 192.144.0.0/21). Внутри самой организации адреса могут распределяться так:

- 1 сеть с длиной маски 21 — 2046 узлов, сетевая маска 255.255.224.0;
- 8 сетей с длиной маски 24 — 254 узла в каждой, сетевая маска 255.255.255.0;
- 16 сетей с длиной маски 25 — 126 узлов в каждой, сетевая маска 255.255.255.128;
- 32 сети с длиной маски 26 — 62 узла в каждой, сетевая маска 255.255.255.192 и т.д.

Ценность протокола CIDR заключается в том, что для перечисленных адресов не обязательно иметь 32, 16 или даже 8 записей в таблице маршрутизации. Ведь все они ссылаются на узлы одной и той же организации, поэтому пакеты предварительно нужно доставлять в общий приемный пункт. В таблицу маршрутизации достаточно внести запись 199.144.0.0/21. Протокол CIDR позволяет даже выделять фрагменты адресных пространств классов A и B, благодаря чему увеличивается число доступных адресов.

Внутри сети допускается смешение подсетей с разной длиной маски, если только они не перекрывают друг друга. Это называется формированием подсетей переменного размера. Например, интернет-провайдер, которому выделена адресная область 193.144.0.0/21, может создать группу сетей /30 для коммутируемых PPP-клиентов, несколько сетей /24 — для крупных клиентов и ряд сетей /27 — для более мелких компаний.

Все узлы конкретной сети должны конфигурироваться с помощью одной сетевой маски. Нельзя для одного узла задать длину маски 24, а для другого — 25.

## Выделение адресов

Формально назначаются лишь адреса сетей. Организации должны самостоятельно определять номера компьютеров и закреплять за ними IP-адреса. Деление на подсети также осуществляется произвольно.

Организация ICANN в административном порядке делегировала полномочия по распределению адресов трем региональным организациям, которые предоставляют блоки адресов интернет-провайдерам в рамках своих регионов (табл. 14.4). Провайдеры, в свою очередь, выделяют адреса отдельным клиентам. Только крупные провайдеры могут напрямую посылать запросы в один из регистров, спонсируемых организацией ICAAN.

**Таблица 14.4. Региональные реестры IP-адресов**

| Организация | Веб-адрес | Охватываемый регион |
|---|---|---|
| ARIN | arin.net | Северная часть Карибских островов |
| APNIC | apnic.net | Азия и Океания, включая Австралию и Новую Зеландию |
| AfriNIC | adrinic.net | Африка |
| LACNIC | lacnic.net | Центральная и Южная Америка, часть Карибских островов |
| RIPE NCC | ripe.net | Европа и прилегающие регионы |

Делегирование полномочий от организации ICANN к региональным регистрам обеспечивает должную агрегацию адресов в таблицах маршрутизации магистральных сетей.

Клиенты, получившие адреса от своего провайдера, не обязаны иметь маршрутные записи в магистральных серверах. Достаточно одной записи, ссылающейся на провайдера.

## Частные адреса и система NAT

Другое временное решение проблемы сокращающегося адресного пространства протокола IPv4 заключается в использовании частных областей IP-адресов, описанных в документе RFC1918. Частные адреса используются только внутри сайта и никогда не демонстрируются в Интернете (по крайней мере, непреднамеренно). Преобразование частных адресов в адреса, выделенные интернет-провайдером, осуществляется пограничным маршрутизатором.

Документ RFC1918 определяет, что одна сеть класса A, 16 сетей класса B и 256 сетей класса C резервируются для частного использования и никогда не выделяются глобально. В табл. 14.5 показаны диапазоны частных адресов (каждый диапазон представлен в более короткой нотации протокола CIDR). Из перечисленных диапазонов организации могут выбирать для себя сети нужного размера.

**Таблица 14.5. IP-адреса, зарезервированные для частного использования**

| Класс | Начало | Конец | Диапазон CIDR |
|---|---|---|---|
| A | 10.0.0.0 | 10.255.255.255 | 10.0.0.0/8 |
| B | 172.16.0.0 | 172.31.255.255 | 172.16.0.0/12 |
| C | 192.168.0.0 | 192.168.255.255 | 192.168.0.0/16 |

Изначально идея состояла в том, чтобы узлы могли самостоятельно выбирать класс адресов из указанных возможностей, чтобы правильно определить свой размер. Однако в настоящее время протокол CIDR и подсети стали универсальным инструментом, поэтому для всех новых частных сетей целесообразнее всего использовать адреса класса A (разумеется, с подсетями).

Для того чтобы узлы, использующие частные адреса, могли получать доступ в Интернет, на пограничном маршрутизаторе организации должна выполняться система NAT (Network Address Translation — трансляция сетевых адресов). Эта система перехватывает пакеты и заменяет в них адрес отправителя реальным внешним IP-адресом. Может также происходить замена номера исходного порта. Система хранит таблицу преобразований между внутренними и внешними адресами/портами, чтобы ответные пакеты доставлялись нужному адресату.

Благодаря нумерации портов, появляется возможность подключить несколько исходящих соединений к общему IP-адресу, чтобы внутренние узлы совместно использовали одинаковый внешний IP-адрес. Иногда в организации достаточно иметь один "настоящий" внешний адрес. Например, эта конфигурация по умолчанию установлена в большинстве популярных маршрутизаторов, использующих кабель и модемы DSL.

Узел, использующий систему NAT, запрашивает адреса у провайдера, но большинство адресов теперь используется для внутрисистемных привязок и не назначается отдельным компьютерам. Если узел позднее захочет сменить провайдера, потребуется лишь изменить конфигурацию пограничного маршрутизатора и его системы NAT, но не самих компьютеров.

Существует возможность заставить операционные системы UNIX или Linux выполнять функции NAT, хотя во многих организациях предпочитают делегировать эти обя-

занности маршрутизаторам или устройствам подключения к сети.[8] Вопросы, связанные с конкретными поставщиками, мы обсудим в этой главе позднее.

Неправильная конфигурация системы NAT может привести к тому, что пакеты с частными адресами начнут проникать в Интернет. Они могут достичь узла назначения, но ответные пакеты не будут получены. Организация CAIDA[9], замеряющая трафик магистральных сетей, сообщает о том, что 0,1–0,2% пакетов, проходящих по магистрали, имеют либо частные адреса, либо неправильные контрольные суммы. На первый взгляд показатель кажется незначительным, но на самом деле это приводит к тому, что каждую минуту в сети циркулируют тысячи пакетов. Информацию по статистике Интернета и средствам измерения производительности глобальной сети можно получить на веб-узле `www.caida.org`.

Одной из особенностей системы NAT является то, что узел Интернета не может напрямую подключиться к внутренним машинам организации. Для того чтобы преодолеть это ограничение, в некоторых реализациях системы NAT разрешается создавать туннели, которые поддерживают прямые соединения с выбранными узлами.[10]

Еще одна проблема заключается в том, что некоторые приложения встраивают IP-адреса в информационную часть пакетов. Такие приложения не могут нормально работать совместно с NAT. К ним относятся определенные протоколы маршрутизации, программы потоковой доставки данных и ряд FTP-команд. Система NAT иногда также отключает виртуальные частные сети (VPN — virtual private network).

Система NAT скрывает внутреннюю структуру сети. Это может показаться преимуществом с точки зрения безопасности, но специалисты считают, что на самом деле система NAT не обеспечивает должную безопасность и уж во всяком случае не устраняет потребность в брандмауэре. Кроме того, она препятствует попыткам оценить размеры и топологию Интернета. См. документ RFC4864, *Local Network Protection for IPv6*, содержащий полезную информацию о реальных и мнимых преимуществах системы NAT и протокола IPv4.

## Адресация в стандарте IPv6

В IPv6 адрес имеет длину 128 бит. Изначально столь длинные адреса вводились для того, чтобы решить проблему сокращающегося адресного пространства IPv4. Сегодня они служат также целям маршрутизации и локализации ссылок.

Адреса в протоколе IPv4 никогда не были географически сгруппированы подобно тому, как это имеет место в случае телефонных номеров или почтовых индексов. Теперь, с появлением протокола CIDR, IP-адреса стали группироваться в кластеры. (Разумеется, прилагательное "географический" относится к пространству маршрутизации, а не к физическим координатам.) Протокол CIDR оказался настолько удачным с технической

[8] Разумеется, многие маршрутизаторы в настоящее время могут запускать встроенные ядра системы Linux. Но даже в этом случае их изначальные системы остаются более эффективными и безопасными, чем универсальные компьютеры, которые также пересылают пакеты.

[9] CAIDA (Cooperative Association for Internet Data Analysis — совместная ассоциация по анализу данных в сети Internet) находится в Центре суперкомпьютеров, который расположен в здании Калифорнийского университета в Сан-Диего (`www.caida.org`).

[10] Многие маршрутизаторы поддерживают также стандарты Universal Plug and Play (UPnP), продвигаемые компанией Microsoft. Эти стандарты позволяют внутренним машинам устанавливать собственные динамические туннели NAT. В зависимости от точки зрения пользователя, это может оказаться как удачным решением, так и угрозой безопасности. При желании эту функциональную возможность маршрутизатора легко отключить.

точки зрения, что иерархические переприсваивания сетевых адресов вошли в стандарт IPv6. Интернет-провайдер, действующий по протоколу IPv6, присваивает вам префиксный адрес, который просто приписывается перед локальной частью вашего адреса, обычно перед адресом пограничного маршрутизатора.

Граница между сетевой и машинной частями адреса в протоколе IPv6 зафиксирована на отметке /64, поэтому не может возникнуть никаких недоразумений, связанных с тем, насколько длинной является сетевая часть в адресе "на самом деле". Иначе говоря, в протоколе IPv6 больше нет настоящих подсетей, хотя термин "подсеть" сохранился как синоним "локальной сети". Несмотря на то что сетевые номера всегда имеют длину 64 бит, маршрутизаторы могут не обращать внимания на все 64 бит в процессе принятия решения. Они могут направлять пакеты на основе анализа префиксов, точно так же как это делается в протоколе CIDR.

Первая схема, описанная в документе RFC2374, предусматривала четыре стандартизованных уровня разделения внутри сетевой части адреса в протоколе IPv6. Однако опыт эксплуатации показал, что интернет-провайдеры могут успешно управлять разделением свои собственных адресов по протоколу IPv4, поэтому первоначальный план в документе RFC3587 был отменен. В настоящее время интернет-провайдеры могут свободно устанавливать границы между частями адресов.

Идентификатор узла, состоящий из 64 бит, потенциально может быть выведен из 48-битовых адресов MAC интерфейсов аппаратного обеспечения.[11]

В протоколе IPv6 MAC-адреса видны на уровне протокола IP, что имеет как положительные, так и отрицательные стороны. Положительный момент заключается в том, что конфигурация адресов узла может быть полностью автоматической. Отрицательный аспект состоит в том, что в первой половине MAC-адреса кодируются тип и модель сетевой платы, что облегчает задачу хакерам. Однако разработчики стандарта IPv6 указали на то, что использование MAC-адресов не является обязательным. Они также предложили схемы включения случайного идентификатора в локальную часть адреса.

Перечислим несколько полезных источников информации, касающейся протокола IPv6.

- `ipv6tf.org` — информационный портал, посвященный протоколу IPv6
- `ipv6.org` — часто задаваемые вопросы и техническая информация
- `www.ipv6forum.com` — форум приверженцев протокола IPv6
- RFC3587 — документ *IPv6 Global Unicast Address Format*
- RFC4291 — документ *IP Version 6 Addressing Architecture*

Были предложены разные схемы перехода с версии IPv4 на IPv6, включая использование системы NAT, чтобы прятать адреса IPv6 при передаче пакетов по туннелю через существующие сети IPv4. В настоящее время чаще всего используются туннельные системы 6to4 и Teredo. Вторая система, которая была названа в честь семейства шашелей, буравящих древесину, может использоваться на устройствах, работающих под управлением системы NAT.

[11] Точнее говоря, в средину добавляется адрес MAC с двумя байтами 0xFFFE, дополненными одним битом (шестым битом первого байта, считая слева и начиная с нуля); см. документ RFC4921. Стандарт, регулирующий преобразование 48-битовых адресов MAC в 64-битовые номера интернет-провайдеров, называется EUI-64.

# 14.5. Маршрутизация

Маршрутизация — это процесс направления пакета по лабиринту сетей, находящихся между отправителем и получателем. В системе TCP/IP маршрутизация происходит примерно так, как путешественник, первый раз посетивший страну, находит нужный ему дом, задавая вопросы местным жителям. Первый человек, с которым он заговорит, возможно, укажет ему нужный город. Войдя в город, путешественник спросит другого человека, и тот расскажет, как попасть на нужную улицу. В конце концов наш путешественник подойдет достаточно близко к конечному пункту своих странствий, чтобы кто-нибудь указал ему дом, который он ищет.

Маршрутная информация в системе TCP/IP имеет форму правил (*маршрутов*), например: "Для того чтобы достичь сети A, посылайте пакеты через компьютер C". Часто существует и стандартный маршрут. В нем объясняется, что нужно делать с пакетами, предназначенными для отправки в сеть, маршрут к которой не указан явным образом.

Данные маршрутизации хранятся в одной из таблиц ядра. Каждый элемент этой таблицы содержит несколько параметров, включая сетевую маску (раньше это поле было опциональным, но теперь оно обязательно, если стандартная сетевая маска неверна). Для направления пакета по заданному адресу ядро подбирает наиболее конкретный маршрут (т.е. тот, где самая длинная маска). Если ни один из маршрутов (в том числе стандартный) не подходит, то отправителю возвращается ICMP-сообщение вида "network unreachable" (сеть недоступна).

Слово "маршрутизация" широко употребляется в двух различных смыслах:

- процедура поиска сетевого адреса в специальной таблице для передачи пакета в пункт его назначения;
- процесс построения этой таблицы.

Ниже мы рассмотрим, как осуществляется переадресация пакетов и как вручную добавить или удалить маршрут. Более сложные вопросы, связанные с работой протоколов маршрутизации, создающих и обслуживающих маршрутные таблицы, освещаются в главе 15.

## Таблицы маршрутизации

Таблицу маршрутизации можно просмотреть с помощью команды **netstat -r**. Команда **netstat -rn** запрещает поиск доменных имен в системе DNS, вследствие чего все адреса будут представлены в числовом виде. Команда **netstat** подробно описывается в разделе 21.5, здесь же мы дадим небольшой пример, чтобы у читателей сложилось представление о том, что такое маршруты.

```
% netstat -rn
Kernel IP routing table
Destination     Genmask          Gateway          Fl  MSS   Iface
132.236.227.0   255.255.255.0    132.236.227.93   U   1500  eth0
default         0.0.0.0          132.236.227.1    UG  1500  eth0
132.236.212.0   255.255.255.192  132.236.212.1    U   1500  eth1
132.236.220.64  255.255.255.192  132.236.212.6    UG  1500  eth1
127.0.0.1       255.255.255.255  127.0.0.1        U   3584  lo
```

В рассматриваемой системе есть два сетевых интерфейса: 132.236.227.93 (eth0) — в сети 132.236.227.0/24 и 132.236.212.1 (eth1) — в сети 132.236.212.0/26.

Поле `destination` обычно содержит адрес сети. В поле `gateway` отображается адрес локального сетевого интерфейса или соседнего узла; в ядре системы Linux для вызова шлюза, установленного по умолчанию, в поле `gateway` может быть записан адрес 0.0.0.0.

Например, четвертый маршрут указывает, что для достижения сети 132.236.220.64/26 пакеты следует посылать в шлюз 132.236.212.6 через интерфейс `eth1`. Вторая запись содержит описание стандартного маршрута. Пакеты, не адресованные явно ни в одну из трех указанных сетей (или самому компьютеру), будут направлены в стандартный шлюз 132.236.227.1.

Компьютеры могут посылать пакеты только тем шлюзам, которые физически подключены к той же сети. Локальные компьютеры могут перемещать пакеты только на один шаг в направлении пункта назначения, поэтому в него бессмысленно включать информацию о шлюзах, не являющихся смежными в таблице локальной маршрутизации. Каждый шлюз, через который проходит пакет, принимает следующее решение о его перемещении, анализируя собственную таблицу локальной маршрутизации.[12]

Вести таблицы маршрутизации можно статически, динамически или комбинированным способом. Статический маршрут задается в явном виде с помощью команды **route**. Он должен оставаться в таблице маршрутизации в течение всего времени работы системы. Во многих случаях такие маршруты задаются на этапе начальной загрузки с помощью одного из сценариев запуска системы. Например, команды

```
route add -net 132.236.220.64 netmask 255.255.255.192
    gw 132.236.212.6 eth1
route add default gw 132.236.227.1 eth0
```

добавляют, соответственно, четвертый и второй маршруты из числа тех, что отображаются выше командой **`netstat -rn`** (первый и третий маршруты добавляются командой **`ifconfig`** при конфигурировании интерфейсов `eth0` и `eth1`).

🕮 Полное описание команды **route** см. в разделе 14.10.

Последний маршрут также добавляется на этапе начальной загрузки. Он определяет псевдоустройство, называемое *интерфейсом обратной связи* (loopback interface). Это устройство препятствует пакетам, которые компьютер посылает сам себе, выходить в сеть. Вместо этого они напрямую переносятся из выходной сетевой очереди ядра во входную очередь.

В относительно стабильной локальной сети статическая маршрутизация является достаточно эффективным решением. Эта система проста в управлении и надежна в эксплуатации, но требует знания системным администратором топологии сети на момент начальной загрузки, а также того, чтобы эта топология в периоды между загрузками не изменялась.

Большинство компьютеров локальной сети имеет единственный выход во внешний мир, поэтому маршрутизация осуществляется очень просто: достаточно на этапе начальной загрузки добавить стандартный маршрут. Компьютер может получить стандартный маршрут вместе со своим IP-адресом у сервера DHCP (этот протокол описан в разделе 14.7).

В сетях с более сложной топологией требуется динамическая маршрутизация. Она осуществляется процессом-демоном, который ведет и модифицирует таблицу маршрутизации. Демоны маршрутизации, "обитающие" на различных компьютерах, взаимодействуют друг с другом с целью определения топологии сети и решения вопроса о том, как добраться до удаленных адресатов. Существует несколько таких демонов. Подробно они будут рассмотрены в главе 15.

[12] Маршрутизация IP-источника является исключением из этого правила; см. раздел 14.8.

## Директивы переадресации протокола ICMP

Несмотря на то что протокол IP не предусматривает средств управления маршрутной информацией, в нем имеется механизм контроля нарушений: директивы переадресации протокола ICMP. Если маршрутизатор направляет пакет компьютеру, находящемуся в той же сети, из которой этот пакет был получен, то что-то работает неправильно. Поскольку отправитель, маршрутизатор и маршрутизатор следующего перехода находятся в одной сети, то пакет можно послать не через два перехода, а через один. Маршрутизатор делает вывод о том, что таблицы маршрутизации отправителя являются неточными или неполными.

В этой ситуации маршрутизатор может уведомить отправителя о проблеме с помощью переадресующего ICMP-пакета. В таком пакете сообщается следующее: "Не нужно посылать мне пакеты для узла *xxx*; их следует адресовать узлу *yyy*".

Теоретически, получив директиву переадресации, отправитель может обновить свою таблицу маршрутизации, чтобы следующие пакеты, предназначенные данному адресату, шли по более прямому пути. На практике переадресация не подразумевает этапа аутентификации и поэтому не может считаться доверенной. Маршрутизатор может проигнорировать полученную команду перенаправить трафик в другое место, но чаще всего системы UNIX и Linux пытаются это сделать по умолчанию. В таких ситуациях необходимо уточнить возможные источники команд переадресации в своей сети и отключить их, если они могут создать проблемы.

В Linux допустимость ICMP-директив определяется элементом **accept_redirects** файловой системы **/proc**. О том, как проверять и устанавливать этот и подобные ему параметры, рассказывается в разделе 14.14.

Для того чтобы отключить маршрутизацию по протоколу ICMP, в системе Solaris следует использовать команду **ndd -set /dev/ip ip_ignore_redirect 1**.

Несмотря на то что система HP-UX также использует команду **ndd** для управления стеком протокола IP, в этой реализации протокола нет возможности игнорировать переадресацию по протоколу ICMP. Однако есть возможность удалить команды переадресации из таблицы маршрутизации через секунду, выполнив следующую команду.

```
ndd -set /dev/ip ip_ire_redirect_interval 1000
```

Некоторые версии системы HP-UX позволяют задавать этот параметр равным не меньше 5 или 60 секунд (сам параметр выражается в миллисекундах), но в системе HP-UX есть возможность задавать еще меньше значения.

Для того чтобы отменить ICNP-переадресацию в системе AIX, необходимо выполнить команду **no -p -o ipignoreredirects=1**. Опция -1 делает эту замену постоянной; при временном тестировании эту опцию следует отключить. Более подробная информация приведена в разделе 14.15.

# 14.6. ARP: протокол преобразования адресов

🕮 Протокол ARP определен в документе RFC826.

Несмотря на то что IP-адреса являются аппаратно-независимыми, для фактической передачи данных на канальном уровне должны применяться аппаратные адреса[13]. ARP

[13] Это не касается двухточечных соединений, где получатель иногда подразумевается неявно.

(Address Resolution Protocol — протокол преобразования адресов) определяет, какой аппаратный адрес связан с тем или иным IP-адресом. Этот протокол можно применять в сетях любых типов, которые поддерживают широковещательный режим, но чаще всего его рассматривают в контексте сетей Ethernet.

Когда компьютер А хочет послать пакет компьютеру Б, находящемуся в том же Ethernet-сегменте, он использует протокол ARP для нахождения аппаратного адреса Б. Если же компьютер Б расположен в другой сети, то компьютер А с помощью подсистемы маршрутизации определяет IP-адрес маршрутизатора следующего перехода, а затем с помощью протокола ARP выясняет аппаратный адрес этого маршрутизатора. Так как в ARP применяются широковещательные пакеты, которые не могут выйти за пределы локальной сети[14], этот протокол позволяет находить только адреса устройств, непосредственно подключенных к той же сети.

Каждый компьютер хранит в памяти специальную таблицу, называемую кешем ARP. Кеш содержит результаты последних ARP-запросов. В нормальных условиях многие адреса, необходимые компьютеру, выявляются вскоре после начальной загрузки, поэтому протокол ARP мало влияет на загруженность сети.

Протокол ARP функционирует путем широковещательной рассылки пакетов примерно следующего содержания: "Знает ли кто-нибудь аппаратный адрес для 128.138.116.4?" Устройство, которое разыскивают, узнает свой IP-адрес и посылает ответ: "Да, это IP-адрес одного из моих сетевых интерфейсов, соответствующий Ethernet-адрес — 8:0:20:0:fb:6a".

Исходный запрос включает IP- и Ethernet-адрес запрашивающей стороны, благодаря чему разыскиваемое устройство может ответить, не посылая собственный ARP-запрос. Это позволяет обоим компьютерам узнать адреса друг друга за один сеанс обмена пакетами. Другие компьютеры, "слышавшие" исходное широковещательное сообщение, посланное инициатором запроса, тоже могут записать информацию о его адресах.

Команда **arp** изучает и обрабатывает содержимое кеша ARP. Обычно она используется для добавления и удаления записей, но может также очищать всю таблицу и отображать ее. В частности, команда **arp -a** отображает содержимое кеша. Формат вывода может варьироваться.

Команда **arp**, как правило, применяется в целях отладки и при работе со специальным оборудованием. Например, если два узла сети имеют одинаковый IP-адрес, то на одном из них запись в ARP-таблице будет правильной, а на другом — нет. С помощью команды **arp** можно найти узел-нарушитель.

## 14.7. DHCP: протокол динамического конфигурирования узлов

🕮 Протокол DHCP определен в документах RFC2131 и 2132.

Когда вы добавляете устройство или компьютер в сеть, то обычно получаете свой IP-адрес в локальной сети, настраиваете соответствующий маршрутизатор, заданный по умолчанию, и присоединяетесь к локальному серверу DNS. Все это за вас может сделать протокол DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования узлов).

[14] Маршрутизатор можно сконфигурировать так, чтобы он пропускал широковещательные пакеты в другие сети, но это, в принципе, неудачная идея. Необходимость пропускать широковещательные пакеты является признаком неблагополучия в работе сети или неудачной архитектуры сервера.

Протокол DHCP дает возможность клиенту взять сетевые и административные параметры "в аренду" у центрального сервера, отвечающего за их распространение. Принцип аренды особенно удобен для персональных компьютеров, которые выключены, когда на них никто не работает, и интернет-провайдеров, чьи клиенты подключаются по коммутируемым линиям.

К "арендуемым" параметрам относятся следующие:

- IP-адреса и сетевые маски;
- адреса шлюзов (стандартные маршруты);
- адреса DNS-серверов;
- имена компьютеров, на которых выполняется система Syslog;
- адреса серверов WINS, X-серверов шрифтов, прокси-серверов и NTP-серверов;
- адреса серверов TFTP (для получения файла начальной загрузки) и десятки других (см. RFC2132).
- Экзотические параметры редко используются на практике.

Периодически клиенты должны повторно обращаться к DHCP-серверу с целью продления срока аренды. Если этого не делать, аренда рано или поздно закончится. DHCP-сервер будет тогда волен предоставить адрес (или иной арендованный параметр) другому клиенту. Срок аренды конфигурируется, но обычно он достаточно велик (до нескольких дней).

Даже если вы хотите, чтобы каждый узел имел собственный постоянный IP-адрес, протокол DHCP может значительно сэкономить ваши время и усилия. Когда сервер DHCP сконфигурирован и запущен, клиенты почти автоматически определяют параметры сетевой конфигурации на этапе начальной загрузки, и никакой путаницы не возникает.

## Программное обеспечение DHCP

Организация ISC (Internet Software Consortium — консорциум разработчиков программного обеспечения для Интернета) поддерживает прекрасный открытый источник информации о протоколе DHCP. В настоящее время широко используются версии пакета IVS 2, 3 и 4, которые прекрасно выполняют основные функции. Версия 3 поддерживает резервное копирование серверов DHCP, а версия 4 — протокол IPv6. Сервер, клиент и агента ретрансляции (agent relay) можно загрузить с веб-сайта `ics.org`.

Все основные дистрибутивы системы Linux используют ту или иную версию пакета ISC, хотя его серверную часть, возможно, придется устанавливать явно. В системе Red Hat эта серверная часть называется **`dhcp`**, в системе Ubuntu — **`dhcp3-server`**, а в системе SUSE — **`dhcp-server`**.

Другие системы часто имеют собственные реализации протокола DHCP. К сожалению, к этой категории относятся и все экземпляры системы UNIX.

Мы рекомендуем не вмешиваться в клиентскую часть протокола DHCP, поскольку эта часть кода относительно проста и поставляется заранее сконфигурированной и готовой к использованию. Изменение клиентской части протокола DHCP — непростая задача.

Однако если вам необходимо запустить *DHCP-сервер*, мы рекомендуем использовать пакет ISC, а не пакеты конкретных поставщиков. В обычной гетерогенной среде процедура администрирования сильно упростится, если будет применяться единая стандартная реализация протокола DHCP. Пакет ISC — это надежное, открытое решение, без проблем инсталлируемое в большинстве систем UNIX.

В следующих подразделах мы вкратце рассмотрим особенности протокола DHCP, инсталляцию сервера ISC, реализующего этот протокол, а также конфигурирование DHCP-клиентов.

## Схема работы DHCP

Протокол DHCP — это расширение протокола BOOTP, который был придуман для того, чтобы бездисковые UNIX-станции могли загружаться по сети. Протокол DHCP не ограничивается этими параметрами, вводя понятие "аренды".

DHCP-клиент начинает диалог с DHCP-сервером, посылая широковещательное сообщение вида "помогите мне узнать, кто я"[15]. Если в локальной сети есть DHCP-сервер, он договаривается с клиентом об аренде IP-адреса и других сетевых параметров (сетевая маска, адреса сервера имен и стандартного шлюза). Если же такого сервера нет, то серверы других подсетей могут получить первоначальное широковещательное сообщение через особую часть программного обеспечения DHCP, которая называется агентом ретрансляции.

По истечении половины срока аренды клиент должен ее продлить. Сервер обязан отслеживать адреса, предоставленные в аренду, и сохранять эту информацию при перезагрузке. Предполагается, что клиенты делают то же самое, хотя это не обязательно. Благодаря этому достигается максимальная стабилизация сетевой конфигурации. Теоретически, все программное обеспечение должно быть готово к мгновенному изменению сетевой конфигурации, но большая часть программного обеспечения по-прежнему необоснованно допускает, что сеть остается неизменной.

## Программное обеспечение DHCP, созданное организацией ISC

Демон сервера ISC называется **dhcpd**, а его файл конфигурации — **dhcpd.conf**. Обычно этот файл находится в каталоге **/etc** или **/etc/dhcp3**. Формат файла конфигурации довольно хрупкий; стоит только пропустить двоеточие, как вы получите запутанное и бесполезное сообщение об ошибке.

Для настройки нового DHCP-сервера необходимо также создать пустой файл базы данных. Проверьте резюме в конце справочной страницы о демоне **dhcpd**, чтобы определить правильное место для лицензионного файла вашей системы. Обычно оно находится где-то под каталогом **/var**.

Для заполнения файла **dhcpd.conf** потребуется следующая информация:

- адреса подсетей, для которых демон **dhcpd** должен управлять IP-адресами, и диапазоны выделяемых адресов;
- список статических адресов, которые вы хотите создать, а также аппаратные MAC-адреса получателей;
- начальный и максимальный сроки аренды в секундах;
- остальные параметры, которые сервер должен передавать DHCP-клиентам: сетевая маска, стандартный маршрут, домен DNS, адреса серверов имен и т.д.

На справочной странице (man page), посвященной демону **dhcpd**, дан обзор процесса конфигурации. Точный синтаксис конфигурационного файла описан на справочной

[15] Клиенты инициируют обмен информацией с DHCP-сервером, используя обобщенный широковещательный адрес. В этот момент клиенты еще не знают маски своих подсетей и, следовательно, не могут использовать широковещательный адрес подсети.

странице файла **dhcpd.conf**. Кроме того, следует убедиться, что демон **dhcpd** автоматически запускается на этапе начальной загрузки системы (см. главу 3). Если система не выполняет эту операцию автоматически, полезно сделать запуск демона условным, осуществляемым при наличии файла **dhcpd.conf**.

Ниже показан пример файла **dhcpd.conf**, взятого из Linux-системы с двумя сетевыми интерфейсами: внутренним и внешним, подключенным к Интернету. На компьютере выполняется система NAT для трансляции адресов внутренней сети, которой предоставляются десять IP-адресов. Файл содержит пустую запись для внешнего интерфейса (обязательна) и запись `host` для одного компьютера, которому нужен фиксированный адрес.

```
# глобальные параметры

option domain-name "synack.net";
option domain-name-servers gw.synack.net;
option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
   range 192.168.1.51 192.168.1.60;
   option broadcast-address 192.168.1.255;
   option routers gw.synack.net;
}

subnet 209.180.251.0 netmask 255.255.255.0 {
}

host gandalf {
   hardware ethernet 08:00:07:12:34:56;
   fixed-address gandalf.synack.net;
}
```

🕮 Более подробная информация о системе DNS приведена в главе 17.

Если вы не назначаете статические адреса, как это сделано выше, необходимо проанализировать, как ваша DHCP-конфигурация будет взаимодействовать с системой DNS. Проще всего присвоить каждому динамически выделяемому адресу общее имя (например, `dhcp1.synack.net`) и разрешить, чтобы имена отдельных компьютеров изменялись вместе с IP-адресами. В качестве альтернативы можно сконфигурировать демон **dhcpd** так, чтобы он обновлял базу данных DNS при выделении очередного адреса. Динамические обновления — сложное решение, но оно позволяет сохранять имена компьютеров неизменными.

Агентом ретрансляции в протоколе DHCP, реализованном организацией ISC, является отдельный демон **dhcrelay**. Это простая программа, не имеющая собственного файла конфигурации, хотя дистрибутивы системы Linux часто добавляют средства инсталляции, передающие соответствующие аргументы командной строки для вашего сайта. Агент **dhcrelay** прослушивает запросы DHCP в локальной сети и передает их на указанные вами удаленные серверы DHCP. Это удобно и для централизации управления службами DHCP, и для облегчения резервного копирования серверов DHCP.

Клиент протокола DHCP по версии организации ISC также не имеет конфигурации. Он хранит файл состояний для каждого соединения в каталогах **/var/lib/dhcp** или **/var/lib/dhclient**. Имена файлов совпадают с именами интерфейсов, которые они

описывают. Например, файл **dhclient-eth0.leases** может содержать все сетевые параметры, которые клиент **dhclient** установил для интерфейса **eth0**.

# 14.8. Вопросы безопасности

Теме безопасности посвящена глава 22, но ряд вопросов, касающихся IP-сетей, заслуживает отдельного упоминания. В этом разделе рассмотрим сетевые механизмы, которые традиционно вызывают проблемы безопасности, и опишем пути решения этих проблем. Детали функционирования систем, приводимых в качестве примера, очень отличаются друг о друга (как и методы их изменения), поэтому они описываются в отдельных подразделах.

## Перенаправление IP-пакетов

Если в UNIX- или Linux-системе разрешено перенаправление IP-пакетов, то компьютер может выступать в качестве маршрутизатора. Иначе говоря, он может принимать пакеты от третьей стороны, поступающие на его сетевой интерфейс, сравнивать их со шлюзом или пунктом назначения и передавать по сети.

Если ваша система не имеет несколько сетевых интерфейсов и не предназначена для функционирования в роли маршрутизатора, эту функцию рекомендуется отключить. Узлы, перенаправляющие пакеты, часто оказываются вовлеченными в атаки на системы безопасности, будучи вынужденными выдавать внешние пакеты за свои собственные. Эта уловка позволяет пакетам злоумышленников обходить сетевые сканеры и фильтры.

Лучше всего, чтобы узел использовал несколько сетевых интерфейсов для своего собственного трафика и не пересылал посторонние пакеты.

## Директивы переадресации протокола ICMP

С помощью переадресующих ICMP-пакетов можно злонамеренно менять направление трафика и редактировать таблицы маршрутизации. Большинство операционных систем по умолчанию принимает эти пакеты и следует содержащимся в них инструкциям. Но, согласитесь, вряд ли можно назвать приемлемой ситуацию, когда на несколько часов весь трафик организации перенаправляется конкуренту, особенно если в это время выполняется резервное копирование. Мы рекомендуем так конфигурировать маршрутизаторы (или системы, играющие роль маршрутизаторов), чтобы переадресующие ICMP-пакеты игнорировались и, возможно, фиксировались в журнальном файле.

## Маршрутизация "от источника"

Механизм маршрутизации "от источника" в протоколе IP позволяет отправителю явно указать последовательность шлюзов, через которые должен пройти пакет на пути к получателю. При этом отключается алгоритм поиска следующего перехода, выполняемый на каждом шлюзе для определения того, куда нужно послать пакет.

Маршрутизация "от источника" была частью исходной спецификации протокола IP и служила для целей тестирования. Но она создает проблему с точки зрения безопасности, ведь пакеты часто фильтруются в зависимости от того, откуда они прибыли. Злоумышленник может так подобрать маршрут, что пакет будет казаться прибывшим из внутренней сети, а не из Интернета, поэтому брандмауэр его пропустит. Мы рекомендуем не принимать и не перенаправлять доставленные подобным образом пакеты.

## Широковещательные ICMP-пакеты и другие виды направленных широковещательных сообщений

Пакеты команды `ping`, несущие в себе широковещательный адрес сети (а не конкретного узла), обычно доставляются всем узлам сети. Такие пакеты применяются в атаках типа "отказ от обслуживания", например в так называемых атаках "smurf" (по названию программы, в которой они впервые были применены).

Широковещательные ICMP-пакеты являются "направленными" в том смысле, что они посылаются по широковещательному адресу конкретной удаленной сети. Стандартные подходы к обработке таких пакетов постепенно меняются. Например, в операционной системе Cisco IOS 11.x и более ранних версий по умолчанию осуществлялась переадресация направленных широковещательных пакетов, но в версиях 12.0 и выше этого уже нет. Обычно можно настроить стек TCP/IP так, чтобы широковещательные пакеты, приходящие из других сетей, игнорировались, но, поскольку это нужно сделать для каждого сетевого интерфейса, подобная задача является весьма нетривиальной в крупной организации.

## Подмена IP-адресов

Исходный адрес IP-пакета обычно заполняется функциями сетевых библиотек и представляет собой адрес узла, с которого был отправлен пакет. Но если программа, создающая пакет, работает с низкоуровневым IP-сокетом, она может подставить любой исходный адрес, какой пожелает. Это называется подменой IP-адресов и обычно ассоциируется с попытками взлома сети. В этой схеме жертвой часто становится компьютер, идентифицируемый по поддельному IP-адресу (если он действительно существует). Сообщения об ошибках и ответные пакеты могут переполнить и вывести из строя сеть жертвы.

Поддельные IP-адреса следует запрещать на пограничном маршрутизаторе, блокируя отправляемые пакеты, исходные адреса которых не находятся в подконтрольном диапазоне. Это особенно важно в университетских сетях, где студенты любят экспериментировать и часто подобным образом срывают свою злость.

В то же время, если в локальной сети используются адреса из частного диапазона, то фильтрации могут подвергаться пакеты с частными адресами, пытающиеся "проскочить" в Интернет. Ответ на такие пакеты никогда не будет получен из-за отсутствия соответствующих маршрутов в магистральной сети. Их появление свидетельствует о том, что в системе имеется внутренняя ошибка конфигурации.

Нужно также защищаться от хакеров, подделывающих исходные адреса внешних пакетов, вследствие чего брандмауэр начинает считать, будто они поступили из локальной сети. Помочь этому может эвристический метод, названный как "одноадресная трансляция по обратному пути " (unicast reverse path forwarding — uRPF). В этом случае IP-шлюзы будут отвергать все пакеты, удовлетворяющие следующему условию: интерфейс, через который пришел пакет, не совпадает с тем интерфейсом, через который пакет уйдет, если его исходный адрес будет равен целевому адресу. Для проверки источника сетевых пакетов этот метод использует обычную таблицу IP-маршрутизации. Метод uRPF применяется не только в специализированных маршрутизаторах, но и в ядре системы Linux, в которой этот режим включен по умолчанию.

Если ваш сайт имеет несколько выходов в Интернет, целесообразно разделить внешние маршруты на исходящие и входящие. В этом случае следует отключить режим uRPF, чтобы протокол маршрутизации работал правильно. Если же выход в Интернет только один, безопаснее включить режим uRF.

## Встроенные брандмауэры

Как правило, соединение вашей локальной сети с внешним миром и управление трафиком в соответствии с правилами сайта, осуществляют сетевые фильтры пакетов или брандмауэры (firewalls). К сожалению, компания Microsoft извратила представление о том, как должен работать брандмауэр, на примере своих систем Windows, печально известных своей уязвимостью. Несколько последних выпусков системы Windows содержали свои собственные брандмауэры и "громко возмущались", когда пользователь пытался их отключить.

Все системы, которые мы используем в качестве примеров, содержат программное обеспечение для фильтрации пакетов, но отсюда не следует, что каждой UNIX- или Linux-машине нужен отдельный брандмауэр. Это не так. Механизмы фильтрации пакетов, встроенные в эти системы, позволяют этим машинам выполнять функции сетевых шлюзов.

Однако мы не рекомендуем использовать рабочую станцию как брандмауэр. Даже самая совершенная операционная система слишком сложна, чтобы быть надежной. Специальное программное обеспечение более предсказуемое и более надежное, даже если оно тайно использует систему Linux.

Даже сложное программное обеспечение, предлагаемое, например, компанией Check Point (чьи программы выполняются на узлах под управлением операционных систем UNIX, Linux и Windows), уступает в надежности межсетевым экранам серии Adaptive Security Appliance компании Cisco, хотя имеет почти такую же цену!

Более подробное обсуждение брандмауэров содержится в разделе 22.11.

## Виртуальные частные сети

Многим организациям, имеющим офисы в различных частях света, хотелось бы, чтобы все эти офисы были соединены одной большой частной сетью. К сожалению, стоимость аренды трансконтинентальных и даже транснациональных линий связи делает это нереальным. Таким организациям приходится использовать Интернет в качестве "частного" канала, организуя серию защищенных, зашифрованных "туннелей" между офисами. Сетевые конгломераты подобного рода называются *виртуальными частными сетями* (virtual private network — VPN).

Возможности виртуальных частных сетей необходимы также сотрудникам, которые должны соединяться с офисом из дома или находятся в поездке. Система VPN не решает всех вопросов, связанных с безопасностью и данным специальным соединением, но во многих отношениях она вполне безопасна.

🕮 О протоколе IPsec рассказывается в разделе 22.14.

В ряде частных сетей используется протокол IPsec, который в 1998 году был стандартизирован организацией IETF в качестве низкоуровневого приложения к протоколу IP. В других сетях, таких как OpenVPN, система безопасности VPN реализуется на основе протокола TCP с помощью криптографического протокола Transport Layer Security (TSL), который ранее был известен под названием Secure Sockets Layer (SSL). Протокол TSL ожидает своей очереди на стандартизацию в организации IETF, хотя он все еще не принят окончательно.

Существует масса запатентованных реализаций систем VPN. Эти системы не взаимодействуют ни друг с другом, ни со стандартизованными системами VPN, но это нельзя считать недостатком, если все конечные точки сети находятся под контролем.

С этой точки зрения системы VPN, основанные на протоколе TLS, являются лидерами. Они также безопасны, как и протокол IPsec, но намного проще. Свободная реализация в виде OpenVN также не наносит никакого вреда. (К сожалению, эта система пока не работает под управлением операционных систем HP-UX и AIX.)

Для поддержки пользователей, работающих дома или в поездке, стало общепринятым использование небольшого компонента Java или ActiveX, загружаемого через их веб-браузеры. Эти компоненты устанавливают соединение VPN с сетью предприятия. Этот механизм удобен для пользователей, но следует учесть, что браузеры сильно отличаются друг от друга: некоторые из них реализуют службу VPN с помощью псевдосетевого интерфейса, а другие используют только специальные порты. Впрочем, веб-браузеры последней категории намного малочисленнее, чем знаменитые веб-прокси.

Пожалуйста, проверьте, что вы правильно понимаете технологию, на которой основаны применяемые решения, и не ожидайте невозможного. Истинная служба VPN (т.е. полноценное IP-соединение через сетевой интерфейс) требует наличия административных привилегий и инсталляции программного обеспечения на клиентской машине, независимо от того, какая операционная система на ней установлена: Windows или UNIX. Кроме того, следует проверить совместимость браузера, поскольку механизм, применяемый при реализации систем VPN с помощью одного браузера, часто не поддерживается другим.

## 14.9. PPP: протокол двухточечного соединения

Протокол PPP определен в документе RFC1331.

PPP (Point-to-Point Protocol — протокол двухточечного соединения) представляет базовый канал связи в виде интерфейса виртуальной сети. Однако поскольку базовый канал может не иметь свойств реальной сети, связь ограничена двумя узлами, расположенными на концах соединения, т.е. виртуальная сеть состоит из двух узлов. Протокол PPP по-разному используется как в самых медленных, так и в самых быстрых IP-соединениях, но по разным причинам.

В асинхронной форме протокол PPP широко известен как протокол, обеспечивающий выход в Интернет по телефонным и последовательным каналам. Эти каналы не предназначены для работы с пакетами, поэтому за кодирование и декодирование сетевых пакетов, передаваемых в поток однородных данных, отвечает драйвер PPP. Он добавляет к пакетам заголовки канального уровня и маркеры-разграничители.

В синхронной форме протокол PPP представляет собой протокол инкапсуляции пакетов, используемый в высокоскоростных соединениях, на обоих концах которых установлены мощные маршрутизаторы. Протокол PPP широко применяется для реализации DSL и кабельных модемов при оказании услуг по широкополосному доступу в Интернет. Во втором случае протокол PPP не только превращает базовую сетевую систему (например, ATM (Asynchronous Transfer Mode — асинхронный способ передачи данных) в технологии DSL (Digital Subscriber Line — цифровая абонентская линия)) в форму, подходящую для работы с протоколами IP, но и осуществляет аутентификацию и управлением доступом в рамках самого соединения. Неким сюрреалистическим образом протокол PPP позволяет реализовать семантику, подобную технологии Ethernet, на основе реальной технологии Ethernet. Эта конфигурация известна под названием "PPP over Ethernet" или PPPoE.

Поскольку протокол PPP был разработан группой, он представляет собой протокол инкапсуляции "всего на свете и кухонной мойки". Помимо спецификации установки,

наладки и демонтажа соединения, протокол PPP осуществляет проверку ошибок, аутентификацию, шифрование и сжатие. Эти функциональные возможности позволяют адаптировать его к любым ситуациям.

Протокол PPP, реализующий сетевые технологии на основе телефонных каналов, когда-то представлял большой интерес для системных администраторов, работающих с операционными системами UNIX или Linux, но быстрое распространение широкополосных сетей сделало телефонные конфигурации практически ненужными. В то же время протокол PPP был широко внедрен в разнообразные специальные сетевые устройства. В настоящее время главной областью применения протокола PPP является обеспечение соединений с помощью сотовых модемов.

## 14.10. Основы конфигурирования сети

Процесс подключения нового компьютера к существующей локальной сети состоит всего из нескольких этапов, но каждая система делает это по-своему. Системы обычно предоставляют графический пользовательский интерфейс, позволяющий настроить базовую сетевую конфигурацию, но в более сложных (или автоматизированных) сценариях инсталляции может понадобиться отредактировать файл конфигурации вручную.

Прежде чем подключать компьютер к сети, где есть выход в Интернет, его нужно оснастить средствами защиты (см. главу 22), чтобы ненароком не привлечь в локальную сеть хакеров.

Основные этапы подключения компьютера.

- Назначение компьютеру уникального IP-адреса и сетевого имени
- Настройка компьютера на конфигурирование своих сетевых интерфейсов в процессе начальной загрузки
- Задание стандартного маршрута и, возможно, других параметров маршрутизации
- Настройка DNS-сервера, чтобы к компьютеру можно было получать доступ через Интернет

Если в сети используется сервер DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования узлов), он возьмет на себя перечисленные выше обязанности. Инсталляция новой операционной системы обычно настраивает свою конфигурацию через протокол DHCP, поэтому новые машины могут вообще не потребовать определения сетевой конфигурации. Общая информация о протоколе DHCP приведена в разделе 14.7.

После внесения любого изменения, которое может повлиять на загрузку операционной системы, следует выполнить ее перезагрузку, чтобы проверить, что машина была подключена правильно. Через полгода, когда произойдет сбой питания и машина откажется загрузиться, будет сложно вспомнить, какие изменения вызвали эти проблемы (см. также главу 21).

Процесс проектирования и инсталляции физической сети описан в главе 16. Если вы имеете дело с существующей сетью и в общих чертах знаете, как она организована, то, наверное, читать о физических аспектах сетей имеет смысл только в том случае, когда планируется расширение действующей сети.

В этом разделе мы рассмотрим разные команды и проблемы, связанные с ручной настройкой конфигурации сети. Этого вполне достаточно для применения к любой версии систем UNIX или Linux. В разделах, посвященных конкретным версиям операционных

систем, мы обсудим особенности, которые отличают эти системы от систем UNIX и Linux и друг от друга.

Просматривая базовую конфигурацию сети любой машины, полезно протестировать соединение с помощью базовых инструментов, таких как утилиты **ping** и **tUbuntuoute**. Эти инструменты описаны в главе 21 (более подробная информация приведена в разделе 21.2).

## Присвоение сетевых имен и IP-адресов

Существует множество теорий о том, как лучше всего определить соответствие между именами компьютеров и IP-адресами в локальной сети: с помощью файла **hosts**, протокола LDAP, системы DNS или комбинации этих средств. С одной стороны, возникают конфликтующие цели, такие как возможность масштабирования, согласованность и удобство эксплуатации, а с другой — системы, достаточно гибкие, чтобы позволить загрузку и функционирование компьютеров, когда не все службы доступны. Управление этими факторами описывается в разделе 19.5.

🕮 Система DNS описывается в главе 17.

Еще один важный аспект, который следует учитывать, — возможность изменения адресов в будущем. Если вы не используете частные адреса RFC1918 (см. раздел 14.4), то ваши IP-адреса могут быть изменены при переключении на нового интернет-провайдера. Это довольно неприятная процедура, если администратор должен лично посетить каждый компьютер и вручную сконфигурировать его. Чтобы не усложнять себе жизнь, указывайте в конфигурационных файлах сетевые имена, а привязки к IP-адресам храните только в базе данных DNS и файлах конфигурации DHCP.

Использование файла **/etc/hosts** — старейший и простейший способ преобразования сетевых имен в IP-адреса. Каждая строка файла начинается с IP-адреса и содержит различные символьные имена, под которыми известен адрес. Ниже показан пример содержимого файла **/etc/hosts** для узла **lollipop**.

```
127.0.0.1       localhost
192.108.21.48   lollipop.xor.com lollipop loghost
192.108.21.254  chimchim-gw.xor.com chimchim-gw
192.108.21.1    ns.xor.com ns
192.225.33.5    licenses.xor.com license-server
```

Минимальный вариант этого файла должен содержать первые две строки. Чаще всего в первой записи файла определяется узел **localhost**. Кроме того, в этом файле могут быть записаны IPv6-адреса.

Поскольку файл **/etc/hosts** содержит лишь локальные адреса привязки и должен находиться на каждой клиентской системе, лучше всего зарезервировать его для отображений, требующихся при загрузке (т.е. для адресов самого узла и маршрутизатора, заданного по умолчанию, а также для имен серверов). Для поиска остальных локальных и глобальных адресов лучше использовать систему DNS или протокол LDAP. Иногда в файле **/etc/hosts** хранятся записи, о которых не следует "знать" другим компьютерам и которых нет в DNS.[16]

Команда **hostname** назначает компьютеру сетевое имя. Она обычно вызывается на этапе начальной загрузки из сценариев запуска системы, которые запрашивают назначаемое имя из конфигурационного файла. (Естественно, все поставщики систем называют

[16] Для этой цели можно также использовать разделенную конфигурацию системы DNS (см. раздел 17.9).

этот файл по-разному. Информация о конкретных системах приведена в разделе 14.11.) В большинстве современных систем компьютеру назначается полностью определенное доменное имя, т.е. оно включает как имя узла, так и имя домена DNS, например `anchor.cs.colorado.edu`.

🕮 Более подробная информация о протоколе LDAP приведена в разделе 19.3.

В небольшой организации вполне можно выделять IP-адреса и сетевые имена вручную. Когда же в организации множество сетей и разнородных административных групп, лучше придерживаться принципа централизации. Об уникальности динамически назначаемых сетевых параметров заботится сервер DHCP. В некоторых организациях сетевыми именами и IP-адресами управляют с помощью баз данных LDAP.

## Команда ifconfig: конфигурирование сетевых интерфейсов

Команда **`ifconfig`** используется для подключения и отключения сетевого интерфейса, а также задания его IP-адреса, маски подсети, других опций и параметров. Она обычно выполняется на этапе начальной загрузки. Аргументы командной строки берутся из конфигурационного файла, но могут применяться и для внесения изменений в работающую систему. Будьте осторожны при модификации удаленной системы, так как не всегда есть возможность оперативно исправить ошибки.

В большинстве случаев команда **`ifconfig`** имеет следующий формат.

**`ifconfig`** *`интерфейс [семейство] адрес опции ...`*

Например, команда

```
ifconfig eth0 128.138.240.1 netmask 255.255.255.0 up
```

задает IPv5-адрес и сетевую маску, связанную с интерфейсом `eth0`, и приводит интерфейс в состояние готовности.

Параметр *интерфейс* обозначает аппаратный интерфейс, к которому применяется команда. Обычно он представляет собой двух- или трехсимвольное имя, за которым следует число, но в системе Solaris этот параметр может быть длиннее. Наиболее распространенные имена — `ie0`, `le0`, `le1`, `ln0`, `en0`, `we0`, `qe0`, `nme0`, `eth0` и `lan0`. В системе Linux интерфейс обратной связи называется `lo`, а в системах Solaris, HP-AX и AIX — `lo0`. В большинстве систем команда **`ifconfig -a`** перечисляет сетевые интерфейсы системы и их текущие установки. В системе HP-UX для этой цели используется команда **`netstat -i`**.

В системе Solaris необходимо сначала "присоединить" сетевые интерфейсы с помощью команды **`ifconfig`** *`интерфейс`* **`plumb`** и лишь затем настраивать конфигурации и выводить на экран с помощью команды **`ifgonfig -a`**.

Параметр *семейство* сообщает команде **`ifconfig`**, какой именно протокол ("семейство адресов") вы хотите конфигурировать. Вы можете установить несколько протоколов для одного интерфейса и использовать их одновременно, но конфигурировать их необходимо по отдельности. Основными опциями являются **`inet`** для протокола IPv4 и **`inet6`** — для протокола IPv6. По умолчанию используется параметр **`inet`**. Система Linux поддерживает также множество унаследованных протоколов, таких как AppleTalk и Novell IPX.

Параметр *адрес* задает IP-адрес интерфейса. Имя узла также допускается в качестве адресного параметра, но на этапе загрузки по имени узла должен определяться его адрес.

Для основного машинного интерфейса это значит, что имя узла должно находиться в локальном файле **hosts**, поскольку другие методы разрешения имен зависят от инициализированной сети.

Ключевое слово **up** указывает на активизацию интерфейса, а ключевое слово **down** — на его отключение. Когда команда **ifconfig** назначает интерфейсу IP-адрес, как в описанном выше примере, ключевое слово **up** подразумевается неявно и может быть опущено.

Команда **ifconfig** понимает множество опций. Мы рассмотрим лишь самые основные. Как всегда, детали, касающиеся конкретной системы, можно узнать на страницах интерактивного руководства. Все опции имеют символические имена. Некоторые опции требуют аргументов, которые должны следовать сразу за названием опции через пробел.

Опция **netmask** задает маску подсети для данного интерфейса. Эта опция обязательна, если подсеть формируется не на основании класса адреса (A, B или C). Маску можно указывать в точечной нотации либо в виде четырехбайтового шестнадцатеричного числа, начинающегося с префикса **0x**. В любом случае единичные биты являются частью номера сети, а нулевые биты — частью номера узла.

Опция **broadcast** задает широковещательный IP-адрес интерфейса в шестнадцатеричной или точечной записи. По умолчанию в широковещательном адресе все биты машинной части равны единице. В приведенном выше примере использования команды **ifconfig** автоматически конфигурируется широковещательный адрес 192.168.1.255.

В качестве широковещательного адреса можно использовать любой IP-адрес, допустимый в сети, к которой подключен компьютер. В некоторых организациях широковещательный адрес специально выбран нестандартным для предотвращения сетевых атак, основанных на широковещательной команде **ping**. Однако это рискованно и, вероятно, излишне. Сбой при конфигурации каждого широковещательного адреса компьютера может вызвать широковещательный шторм, в ходе которого пакеты будут блуждать от машины к машине, пока не будет исчерпано время TTL (Time to live — время жизни пакета данных в протоколе IP. — *Примеч. ред.*)[17]

Лучший способ избежать проблем с широковещательными запросами состоит в том, чтобы запретить пограничному маршрутизатору перенаправлять их, а отдельным узлам — отвечать на них. В главе 22 будет рассказано о том, как реализовать подобные ограничения.

Система Solaris интегрирует команду **ifconfig** с клиентским демоном протокола DHCP. Команда **ifconfig** *интерфейс* **dhcp** конфигурирует именованный интерфейс с помощью параметров, арендованных у локального DHCP-сервера, затем запускает программу **dhcpagent**, чтобы управлять этими параметрами в течение долгого времени. Другие системы используют команду **ifconfig** независимо от протокола DHCP, при этом программное обеспечение DHCP функционирует на отдельном уровне.

[17] Широковещательный шторм (broadcast storm) возникает из-за того, что для транспорта пакетов должен использоваться один и тот же широковещательный адрес, независимо от того, какой адрес установлен. Например, машина X считает, что широковещательный адрес равен A1, а машина Y считает, что он равен A2. Если машина X посылает пакет по адресу A1, то машина Y получит пакет (поскольку адрес назначения на уровне соединения — это широковещательный адрес), затем увидит, что этот пакет предназначен ни ей, ни широковещательному адресу (поскольку машина Y полагает, что широковещательный адрес равен A2), и может возвратить этот пакет обратно в сеть. Если обе машины находятся в состоянии машины Y, то пакет будет циркулировать в сети, пока не закончится его время жизни. Широковещательные штормы могут разрушить полосу пропускания, особенно в больших коммутируемых сетях.

Кроме того, с помощью команды **ifconfig** *интерфейс* можно получить конфигурацию отдельного интерфейса.

```
solaris$ ifconfig e1000g0
e1000g0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
    index 2 inet 192.168.10.10 netmask ffffff00 broadcast 192.168.10.255
redhat$ ifconfig e1000g0
eth0  Link encap:Ethernet  HWaddr 00:02:B3:19:C8:86
      inet addr:192.168.1.13 Bcast:192.168.1.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:206983 errors:0 dropped:0 overruns:0 frame:0
      TX packets:218292 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      Interrupt:7 Base address:0xef00
```

Отсутствие коллизий в интерфейсе Ethernet во втором примере может указывать на слабо загруженную сеть или, что более вероятно, на коммутируемую сеть. В сети общего доступа (организованной с помощью концентраторов, а не коммутаторов) этот показатель должен быть ниже уровня 5% от числа отправленных пакетов. Большое число коллизий свидетельствует о загруженности сети, которую, возможно, требуется разбить на подсети или перевести на коммутируемую инфраструктуру.

Теперь, когда мы рассмотрели процедуру ручного конфигурирования сетевого интерфейса, осталось выяснить, как задавать параметры команды **ifconfig** на этапе начальной загрузки системы. Кроме того, требуется убедиться в том, что новые значения введены правильно. Для этого необходимо отредактировать один или несколько конфигурационных файлов. Информация, касающаяся конкретных систем, приведена далее в этой главе.

И еще одно замечание по поводу команды **ifconfig**: вы можете назначать интерфейсу несколько адресов, используя "интерфейсы виртуальной сети" или "IP-псевдонимы" (IP-aliases). Администраторы могут делать это для того, чтобы позволить одной машине служить хостом для нескольких веб-сайтов (см. раздел 23.3).

## Параметры сетевого оборудования

Довольно часто сетевое устройство имеет настраиваемые параметры, зависящие от типа передающей среды. Чрезвычайно распространенный пример — современные платы Ethernet, которые могут поддерживать скорость передачи 10, 100, 1000 и даже 10000 Мбит/с как в полудуплексном, так и в дуплексном режимах. Большинство устройств по умолчанию находится в режиме автоматического согласования, когда обе стороны — сетевая плата и вышестоящее устройство (обычно порт коммутатора) — пытаются угадать параметры друг друга.

По историческим причинам, процедура автоматического согласования была разработана так, что она напоминает попытку вставить ключ в замочную скважину с завязанными глазами. Следствием неудачного согласования является высокий коэффициент потери пакетов (особенно больших пакетов).

Если в сети начинают происходить загадочные потери пакетов, в первую очередь отключите везде режим автоматического согласования. Зафиксируйте скорость и режимы работы сетевых интерфейсов как на серверах, так и на портах коммутатора, к которым они подключены. Автоматическое согласование удобно в динамических сетях организаций, где многие сотрудники имеют портативные компьютеры, и совершенно не нужно для статически подключенных компьютеров.

Точный метод, с помощью которого задаются параметры автоматического согласования, меняется от системы к системе. Обсуждение этих вопросов мы начнем в разделе 14.11.

## Команда route: конфигурирование статических маршрутов

Команда **route** определяет статические маршруты — явно заданные элементы таблицы маршрутизации, которые обычно не меняются даже в тех случаях, когда запускается демон маршрутизации. При подключении нового компьютера к локальной сети достаточно указать стандартный маршрут.

Маршрутизация рассматривается нами в этой главе и следующей. И хотя информация по основам маршрутизации и команде **route** приведена здесь, полезно прочитать также несколько первых разделов главы 15.

Маршрутизация осуществляется на сетевом уровне. Когда приходит пакет, предназначенный для другого узла, его целевой IP-адрес сравнивается с записями в таблице маршрутизации ядра. При совпадении (хотя бы частичном) с каким-нибудь маршрутом в таблице пакет направляется по IP-адресу следующего шлюза, связанного с данным маршрутом.

Но есть два особых случая. Во-первых, пакет может быть адресован компьютеру, включенному в ту же сеть, что и узел-отправитель. В этом случае адрес следующего шлюза соответствует одному из интерфейсов локального компьютера, и пакет посылается прямо в пункт назначения. Маршруты такого типа добавляются командой **ifconfig** при конфигурировании интерфейса.

Во-вторых, может вообще не оказаться маршрута, совпадающего с адресом пункта назначения. В этом случае применяется маршрут по умолчанию, если таковой имеется, иначе отправителю посылается ICMP-сообщение "network unreachable" (сеть недоступна) или "host unreachable" (узел недоступен).

Многие локальные сети имеют единственный выход во внешний мир, поэтому им требуется только один маршрут, указывающий на этот выход. В магистральной сети Интернета нет стандартных маршрутов — это конец пути.

Каждая команда **route** добавляет или удаляет один маршрут. К сожалению, команда **route** — это одна из немногих команд системы UNIX, которые одинаково работают во всех системах, но в каждой из них имеют свой синтаксис. Прототип команды **route** выглядит следующим образом.

```
# route add -net 192.168.45.128/25 zulu-gw.atrust.net
```

Эта команда добавляет маршрут в сеть 192.168.45.128/25 через шлюз маршрутизатора `zulu-gw.atrust.net`, который должен быть либо соседним узлом, либо одним из локальных интерфейсов узла. (Система Linux требует указывать имя **gw** перед адресом шлюза.) Естественно, маршрутизатор должен уметь преобразовывать имя `zulu-gw.atrust.net` в IP-адрес. Если ваш DNS-сервер находится на другой стороне шлюза, используйте числовой IP-адрес!

Система Linux в качестве пункта назначения для маршрута допускает использование имени интерфейса (например, `eth0`). Это эквивалентно указанию первичного IP-адреса интерфейса в качестве адреса шлюза. Иначе говоря, I-стек пытается осуществить непосредственную доставку на этот интерфейс, а не перенаправлять пакет на отдельный шлюз. Адреса шлюзов на этом маршруте в результатах работы команды **netstat -r** выглядят как 0.0.0.0. Для того чтобы увидеть, где именно проходит маршрут, необходимо поискать в столбце `Iface` имя интерфейса.

Сети назначения традиционно задаются отдельными IP-адресами и сетевыми масками, но в настоящее время все версии команды **route**, за исключением ее версии в системе HP-UX, понимают обозначения CIDR (например, 128.138.176.0/20). Система обозначений CIDR яснее, и пользователю остается лишь побеспокоиться о некоторых вопросах синтаксиса, зависящего от системы. Даже система Linux допускает обозначения CIDR, хотя в справочной системе на странице, посвященной команде **route**, об этом ничего не сказано.

В системе Solaris есть остроумная опция **-p** для команды **route**, которая позволяет сохранить изменения даже после перезагрузки. Кроме таблицы маршрутизации, принадлежащей ядру, изменения записываются в файл **/etc/inet/static_routes** и хранятся там в момент загрузки.

Перечислим еще несколько приемов.

- Если вы хотите увидеть имена, а не числа, то для того, чтобы проверить существующие маршруты, используйте команды **netstat -nr** или **netstat -r**. Числа часто удобнее при отладке, поскольку имя может быть искажено. Пример работы команды **netstat** приведен в разделе 14.5.
- Для указания стандартного маршрута системы используйте ключевое слово **default** вместо адреса или сетевого имени.
- Для того чтобы удалить записи из таблицы маршрутизации, используйте команды **route delete** или **route del**.
- Для инициализации таблицы маршрутизации и начала работы в системе UNIX используйте команды **route -f** или **route flush**.
- Маршруты в протоколе IPv6 задаются так же, как и маршруты в протоколе IPv4. Для того чтобы указать команде **route**, что используется адресное пространство протокола IPv6, необходимо использовать опции **-inet6** или **-A inet6**.
- Файл **/etc/networks** задает преобразование имен в сетевые номера, аналогично тому как файл **hosts** отображает имена узлов в IP-адреса. Такие команды, как **route**, ожидающие сетевой номер, могут принимать имя, если оно указано в файле **networks**. Сетевые имена можно также перечислить в базах данных NIS или DNS (см. документ RFC1101).
- Для установки маршрута, специфичного для заданного IP-адреса, можно использовать команду **route add -host**. По существу, эта команда аналогична заданию маршрута с маской 255.255.255.255, но при этом он отдельно отмечается в таблице маршрутизации.

## Конфигурирование DNS

Для того чтобы сконфигурировать компьютер в качестве DNS-клиента, достаточно отредактировать файл **/etc/resolv.conf**. DNS-служба при этом, строго говоря, не нужна (см. раздел 19.5), но трудно представить себе ситуацию, в которой без нее можно было бы вообще обойтись.

Файл **/etc/resolv.conf** содержит список DNS-доменов, просматриваемых при анализе неполных имен (например, "anchor" вместо `anchor.cs.colorado.edu`), и список IP-адресов серверов имен, в которых осуществляется поиск имен. Ниже показан пример этого файла; подробнее о нем рассказывается в разделе 17.3.

```
search cs.colorado.edu colorado.edu
nameserver 128.138.242.1
nameserver 128.138.243.151
nameserver 192.108.21.1
```

Первым должен быть приведен "ближайший" стабильный сервер имен, так как он опрашивается в первую очередь. Всего можно задать три записи `nameserver`. Желательно указывать более одного сервера. Период тайм-аута DNS-запроса к конкретному серверу имен достаточно велик, поэтому, если первый сервер не отвечает, пользователи это заметят. Если локальный узел получает адреса своих DNS-служб через протокол DHCP, то клиентское программное обеспечение протокола DHCP записывает адреса, полученные в аренду, в файл **`resolv.conf`**. Поскольку конфигурация DHCP в большинстве систем задана по умолчанию, обычно нет необходимости редактировать файл **`resolv.conf`** самостоятельно, если DHCP-сервер настроен корректно.

Многие сайты используют реализацию DNS-сервера Active Directory, созданную компанией Miscosoft. Она прекрасно работает с файлом **`resolv.conf`** в системах UNIX и Linux, и нет никакой необходимости придумывать что-либо другое.

## 14.11. Сетевое конфигурирование в различных системах

В ранних версиях систем UNIX настройка сетевой конфигурации осуществлялась путем редактирования сценариев системной загрузки и непосредственного изменения содержащихся в них команд. Современные системы содержат сценарии, доступные только для чтения; они охватывают множество сценариев конфигурирования и осуществляют выбор, основываясь на повторном использовании информации, содержащейся в других системных файлах, или на данных, содержащихся в файлах конфигурации.

Несмотря на то что разделение конфигурации и реализации — хорошая идея, каждая система осуществляет ее немного иначе, чем другая. Формат и способы использования файлов **`/etc/hosts`** и **`/etc/resolv.conf`** в системах UNIX и Linux довольно хорошо согласованы, но этого нельзя со всей определенностью сказать обо всех системах.

Многие системы обеспечивают графический пользовательский интерфейс для выполнения основных задач, связанных с конфигурированием, однако соответствие между визуальным интерфейсом и файлами конфигурации часто остается неясным. Кроме того, графический пользовательский интерфейс часто игнорирует сложные варианты конфигурации и остается довольно неудобным для удаленного и автоматизированного администрирования. В следующих разделах мы раскритикуем некоторые варианты конфигурирования, опишем то, что происходит за кулисами, и раскроем детали сетевого конфигурирования для каждой из популярных операционных систем. В частности, мы рассмотрим следующие вопросы.

- Основная конфигурация
- Конфигурация DHC-клиента
- Динамическое конфигурирование и настройка
- Безопасность, брандмауэры, фильтрация и конфигурация NAT
- Трюки

Однако в каждом разделе не обязательно обсуждать все операционные системы.

Следует иметь в виду, что в большинстве случаев сетевое конфигурирование осуществляется в момент загрузки, поэтому эта тема частично перекрывается с информацией, представленной в главе 3.

## 14.12. Сетевое конфигурирование в системе Linux

Система Linux — одна из систем, в которых новые сетевые функциональные возможности реализуются в первую очередь. Семейство систем Linux иногда изменяется настолько быстро, что теряет возможность взаимодействия с остальной сетевой инфраструктурой. Например, реализация в системе Linux явного уведомления о перегруженности (explicit cogestion notification — ECN), описанного в документе RFC2481, противоречит некорректным установкам, заданным по умолчанию в устаревшем брандмауэре компании Cisco, что вызывает удаление всех пакетов, содержащих набор битов ECN.

Разработчики системы Linux любят чинить ее на скорую руку и часто реализуют функциональные возможности и алгоритмы, пока не имеющие принятых стандартов. Примером этого является включение сменных алгоритмов контроля за перегрузкой в ядро системы Linux версии 2.6.13. Некоторые функциональные возможности предусматривают варианты для сетей с потерями (lossy networks), высокоскоростных глобальных вычислительных сетей (wide area network — WAN) с большими потерями пакетов, спутниковыми связями и т.д. Стандартный механизм "reno", реализованный в протоколе TCP (медленный старт, избегание перегрузок, быстрая повторная пересылка и быстрое восстановление), по-прежнему используется по умолчанию, но в конкретной среде может оказаться более приемлемым другой вариант.

После внесения любых изменений в файл, управляющий сетевой конфигурацией в момент загрузки, необходимо либо перезагрузить систему, либо отключить сетевой интерфейс, а затем снова включить его, чтобы изменения вступили в силу. В большинстве систем Linux для этой цели можно использовать команды **ifdown** *интерфейс* или **ifup** *интерфейс*, хотя их реализации не являются идентичными. (В системе SUSE команды **ifup** и **ifdown** выполняются, только если работа в сети не находится под управлением демона NetworkManager.)

### Демон NetworkManager

Поддержка работы в мобильной сети, осуществляемая в системе Linux, некоторое время была довольно неорганизованной, пока в 2004 году не появился демон NetworkManager. Он состоит из службы, предназначенной для непрерывного функционирования, а также приложения для работы с областью уведомлений, позволяющего конфигурировать индивидуальные сетевые интерфейсы. Кроме разнообразных проводных сетей, демон NetworkManager поддерживает также работу беспроводных сетей, систем беспроводного широкополосного доступа и частных виртуальных сетей (VPN). Он постоянно осуществляет оценку доступных сетей и переключает службу на "предпочтительные" сети, как только те становятся доступными. Наиболее предпочтительными являются проводные сети, за ними следуют обычные беспроводные сети.

Все это довольно сильно изменило сетевую конфигурацию системы Linux. Помимо того, что она стала более изменчивой по сравнению с традиционной статической конфигурацией, теперь она запускается и управляется пользователем, а не системным администратором. Демон NetworkManager широко внедрен в дистрибутивы систем Linux,

включая все экземпляры, рассматриваемые в книге, но для того чтобы избежать нарушения существующих сценариев и настроек, он обычно предусматривается как "параллельная вселенная" в дополнение к традиционной сетевой конфигурации, которая использовалась в прошлом.

Система SUSE предлагает сделать выбор между миром демона NetworkManager и устаревшей конфигурацией, управляемой посредством программного пакета YaST. Система Ubuntu запускает демона NetworkManager по умолчанию, сохраняя при этом статически конфигурируемые сетевые интерфейсы за пределами демона NetworkManager. Система Red Hat Enterprise Linux вообще не запускает демона NetworkManager по умолчанию.

Демон NetworkManager, в основном, используется в ноутбуках, поскольку их сетевое окружение может часто изменяться. Для серверов и настольных систем демон NetworkManager не обязателен и фактически может даже усложнять администрирование. В этом окружении он должен игнорироваться или отключаться.

## Сетевое конфигурирование в системе Ubuntu

Как показано в табл. 14.6, система Ubuntu задает конфигурацию сети в файлах **/etc/hostname/** и **/etc/network/interfaces**, а справочная информация записана в файле **/etc/network/options**.

**Таблица 14.6. Сетевая конфигурация системы Ubuntu, записанная в каталоге /etc**

| Файл | Содержимое |
|---|---|
| **hostname** | Имя компьютера |
| **network/interfaces** | IP-адрес, сетевая маска, стандартный маршрут |

Имя компьютера задается в файле **/etc/hostname/**. Имя, записанное в этом файле, должно быть полностью квалифицированным; оно используется в самых разных контекстах, некоторые из которых требуют полной квалификации.

IP-адрес, сетевая маска и стандартный шлюз задаются в файле **/etc/network/interfaces**. Каждый интерфейс задается строкой, начинающейся с ключевого слова `iface`. За этой строкой могут следовать строки, задающие дополнительные параметры. Рассмотрим пример.

```
auto lo eth0
iface lo inet loopback
iface eyh0 inet static
    address 192.168.1.102
    netmask 255.255.255.0
    gateway 192.168.1.254
```

Команды **ifup** и **ifdown** читают этот файл и, соответственно, подключают или отключают интерфейсы, вызывая низкоуровневые команды (например, **ifconfig**) с соответствующими параметрами. Строка `auto` задает интерфейсы, которые должны включаться при загрузке или при выполнении команды **ifup -a**.

Ключевое слово `inet` в строке `iface` определяет семейство адресов наподобие **ifconfig**. Ключевое слово `static` обозначает способ описания интерфейса и указывает на то, что IP-адрес и сетевая маска интерфейса `eth0` будут заданы непосредственно. Для статических интерфейсов строки `address` и `netmask` являются обязательными. В первых версиях ядра Linux необходимо было указывать также адрес сети, но современные ядра "умнее" и могут самостоятельно определить его на основании IP-адреса и маски. В строке `gateway` определяется адрес стандартного шлюза.

## Сетевое конфигурирование в системе SUSE

Система SUSE осуществляет выбор между демоном NetworkManager и традиционной системой конфигурирования. Выбор происходит в пакете YaST; вы можете также использовать графический пользовательский интерфейс пакета YaST. Здесь будем предполагать, что используется традиционная система. Кроме конфигурирования сетевых интерфейсов, пакет YaST обеспечивает непосредственные пользовательские интерфейсы для ввода файла **/etc/hosts**, статических маршрутов и конфигурации DNS. Описание файлов конфигурации приведено в табл. 14.7.

**Таблица 14.7. Сетевые конфигурационные файлы SuSE из каталога /etc/sysconfig/network**

| Файл | Содержимое |
|---|---|
| **ifcfg-интерфейс** | Имя компьютера, IP-адрес, сетевая маска и др. |
| **ifroute-интерфейс** | Определение маршрута, связанного с интерфейсом |
| **routes** | Стандартный маршрут, статические маршруты |
| **config** | Множество переменных, которые используются относительно редко |

За исключением параметров DNS и имени компьютера, система SUSE устанавливает большинство сетевых опций в файлах **ifcfg-*интерфейс***, находящихся в каталоге **/etc/sysconfig/network**. Каждому интерфейсу в системе должен соответствовать отдельный файл.

Кроме IP-адресов, шлюза и информации о широкополосном доступе для интерфейса, файлы **ifcfg-*** могут содержать другие сетевые настройки. В качестве примера можно открыть файл **ifcfg.template**, в котором содержатся ясные комментарии о возможных параметрах. Рассмотрим пример, содержащий наши комментарии.

```
BOOTPROTO='static'          #Подразумевается статический IP-адрес
IPADDR='192.168.1.4/24'     #/24 определяет переменные NETWORK и NETMASK
NAME='AMD PCnet - Fast 79C971' #Используется для запуска
                               #и остановки интерфейса
STARTMODE='auto'            #Запускается автоматически во время загрузки
USERCONTROL='no'            #Отключает управление со стороны графического
                            #пользовательского интерфейса kinternet/cinternet
```

Глобальная статическая информация о маршрутизации в системе SUSE (включая стандартный маршрут) хранится в файле **route**. Каждая строка в этом файле напоминает команду **route** с пропущенными необязательными именами и содержит информацию о пункте назначения, шлюзе, сетевой маске, интерфейсе и других необязательных параметрах, которые должны храниться в таблице маршрутизации и предназначаются для демонов маршрутизации. Файл **route** для компьютера, конфигурация которого показана выше и который имеет только один стандартный маршрут, содержит следующую строку.

```
default 192.168.1.254 - -
```

Маршруты, являющиеся уникальными для каждого конкретного интерфейса, хранятся в файле **ifroute-*интерфейс***, в котором номенклатура компонента *интерфейс* совпадает с номенклатурой файлов **ifcfg-***. Содержимое этих файлов имеет тот же формат, что и содержимое файла **route**.

## Сетевое конфигурирование в системе Red Hat

Графический пользовательский интерфейс для сетевой конфигурации системы Red Hat называется **system-config-network**; он также доступен из меню System->Administration под именем Network. Этот инструмент обеспечивает простой пользовательский интерфейс для конфигурирования индивидуальных сетевых интерфейсов и статических маршрутов. Он также имеет панели для установки туннелей IPsec, конфигурирования системы DNS и добавления записей в файл **/etc/hosts**.

Файлы конфигурации, которые можно редактировать с помощью графического пользовательского интерфейса, перечислены в табл. 14.8.

**Таблица 14.8. Сетевые конфигурационные файлы SuSE из каталога /etc/sysconfig/network**

| Файл | Содержимое |
|---|---|
| **network** | Имя компьютера, стандартный маршрут |
| **static-routes** | Статические маршруты |
| **network-scripts/ifcfg-ifname** | Параметры интерфейсов: IP-адреса, сетевые маски и т.д. |

Имя компьютера записывается в файл **/etc/sysconfug/network**, который также содержит имя домена DNS и информацию о стандартном шлюзе.

Например, ниже приведено содержимое файла **network** для компьютера с единственным интерфейсом Ethernet.

```
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=redhat.toadranch.com
DOMAINNAME=toadranch.com      ### необязательное поле
GATEWAY=192.168.1.254
```

Данные, связанные с интерфейсом, хранятся в файле **/etc/sysconfig/network-scripts/ifcfg-***ifname*, где параметр *ifname* — имя сетевого интерфейса. Эти файлы конфигурации задают IP-адрес, сетевую маску, сеть и широковещательный адрес для каждого интерфейса. Они также содержат строку, указывающую, должен ли интерфейс включаться в момент загрузки.

Типичный компьютер имеет файлы для интерфейса Ethernet (eth0) и интерфейс обратной связи (lo). Рассмотрим пример.

Обычно в системе присутствуют файлы для Ethernet-платы (`eth0`) и интерфейса обратной связи (`lo`). Вот каким будет содержимое файлов **ifcfg-ifcfg-eth0** и **ifcfg-lo** для узла `redhat.toadranch.com`, описанного выше в файле **network**.

```
DEVICE=eth0
IPADDR=192.168.1.13
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
```

и

```
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
```

```
ONBOOT=yes
NAME=loopback
```

Настройки интерфейса eth0 на основе протокола DHCP выглядят еще проще.

```
DEVICE=eth0
BOOTROTO=dhcp
ONBOOT=yes
```

После изменения информации о конфигурации в файле **/etc/sysconfig** следует выполнить команду **ifdown** *ifname*, а затем **ifup** *ifname*, где *ifname* — имя соответствующего интерфейса. Для одновременной конфигурации нескольких интерфейсов можно использовать команду **service network restart**, которая перезагружает сеть. (На самом деле эта команда представляет собой упрощенный способ вызова утилиты **/etc/rc.d/init.d/network**, которая вызывается в момент загрузки с аргументом **start**.)

Сценарии запуска Red Hat также могут конфигурировать статические маршруты. Любой маршрут, внесенный в файл **/etc/sysconfig/static-routes**, добавляется в таблицу маршрутизации на этапе начальной загрузки. Записи этого файла содержат аргументы для команды **route add**, хотя и в смешанном порядке (интерфейс указан первым, а не последним).

```
eth0 net 130.225.204.48 netmask 255.255.255.248 gw 130.225.204.49
eth1 net 192.38.8.0 netmask 255.255.255.224 gw 192.38.8.129
```

Интерфейс указывается первым, но на самом деле он перемещается в конец командной строки **route**, устанавливая связь маршрута с конкретным интерфейсом. (Этот же прием используется в графических пользовательских интерфейсах, где маршруты конфигурируются как часть настройки каждого интерфейса.) Оставшаяся часть строки содержит аргументы команды **route**. Пример команды **static-routes**, приведенный выше, генерирует следующие команды.

```
route add -net 130.225.204.48 netmask 255.255.255.248 gw 130.225.204.49 eth0
route add -net 192.38.8.0 netmask 255.255.255.224 gw 192.38.8.129 eth1
```

Ядра современных систем семейства Linux не используют параметр `metric` команды **route**, но позволяют вводить его в таблицу маршрутизации для использования демонами маршрутизации.

## Настройка сетевого оборудования в системе Linux

Команда **ethtool** запрашивает и устанавливает параметры сетевого интерфейса, характерные для среды, например скорость связи и дуплекс. Она представляет собой замену старой команды **mii-tool**, но в некоторых системах используются обе команды.

Запросить состояние интерфейса можно, просто назвав его имя. Например, интерфейс eth0 (типичная сетевая карта на материнской плате персонального компьютера), описанный ниже, допускает автонастройку и в настоящий момент работает на предельной скорости.

```
ubuntu# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:     10baseT/Half     10baseT/Full
                                  100baseT/Half    100baseT/Full
                                  1000baseT/Half   1000baseT/Full
        Supports auto-negotiation: Yes
```

```
Advertises link modes:    10baseT/Half     10baseT/Full
                          100baseT/Half    100baseT/Full
                          1000baseT/Half   1000baseT/Full
Advertises auto-negotiation: Yes
Speed: 1000Mb/s
Duplex: Full
Port: MII
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
Supports Wake-on: pumbg
Wake-on: g
Current message level: 0x00000033 (51)
Link detected: yes
```

Для того чтобы перевести этот интерфейс в полнодуплексный режим на скорости 100 Мбит/с, используется следующая команда.

```
ubuntu# ethtool -s eth0 speed 100 duplex full
```

Если вы пытаетесь определить, надежна ли автонастройка в вашей среде, то может оказаться полезной команда **ethtool -r**. Она вызывает мгновенную перенастройку параметров связи.

Еще одним полезным вариантом является опция -k, которая показывает, какая из задач, связанных с протоколом, была назначена сетевому интерфейсу, а не выполняется ядром. Большинство интерфейсов может вычислять контрольные суммы, а также выполнять сегментацию. Если вы сомневаетесь, что сетевой интерфейс надежно выполняет эти задания, то лучше их отключить. Для включения и отключения этих функций используется команда **ethtool -K** в сочетании с другими опциями. (Опция **-k** демонстрирует текущее состояние, а опция **-K** устанавливает его.)

Любые изменения, сделанные с помощью команды **ethtool**, являются временными. Если вы хотите, чтобы они стали постоянными, то следует убедиться, что команда **ethtool** выполняется как часть конфигурации сетевого интерфейса. Для этого лучше всего сделать ее частью конфигурации каждого интерфейса; если просто выполнять какие-то команды **ethtool** во время загрузки, то ваша конфигурация не будет правильно учитывать случаи, когда интерфейсы запускаются вновь без перезагрузки системы.

Работая с системой Red Hat, можно включить в файл конфигурации интерфейса, расположенный в каталоге **/etc/sysconfig/network-scripts**, строку `ETHTOOL_OPTS=line`. В качестве аргумента команда **ifup** передает команде **ethtool** целую строку.

Система SUSE выполняет команду **ethtool** аналогично системе Red Hat, но опция называется `ETHTOOL_OPTIONS` и файлы конфигурации каждого интерфейса хранятся в каталоге **/etc/sysconfig/network**.

В системе Ubuntu команду **ethtool** можно выполнить в рамках сценария `post-up`, заданного в конфигурации интерфейса в каталоге **/etc/network/interfaces**.

## Опции протокола Linux TCP/IP

Система Linux помещает представление каждой настраиваемой переменной ядра в виртуальную файловую систему **/proc**. Сетевые переменные находятся в каталоге **/proc/**

**sys/net/ipv4**. Приведем для иллюстрации сокращенный список наиболее интересных из них.

```
ubuntu$ cd /proc/sys/net/ipv4; ls -F
...                          ...                       tcp_no_metrics_save
conf/                        tcp_congestion_control    tcp_orphan_retries
icmp_echo_ignore_all         tcp_dma_copybreak         tcp_reordering
icmp_echo_ignore_broadcast   tcp_dsack                 tcp_retrans_collapse
                             tcp_ecn                   tcp_retries1
icmp_ratelimit               tck_fack                  tcp_retries2
icmp_ratemask                tcp_fin_timeout           tcp_rfc1337
igmp_max_memberships         tcp_frto                  tcp_rmem
igmp_max_msf                 tcp_frto_response         tcp_sack
inet_peer_gc_maxtime         tcp_keepalive_intvl       ...
inet_peer_gc_mintime         tcp_keepalive_probes      tcp_stdurg
inet_peer_maxttl             tcp_keepalive_time        tcp_synack_retries
inet_peer_minttl             tcp_low_latency           tcp_syncookies
inet_peer_threshold          tcp_max_orphans           tcp_syn_retries
ip_default_ttl               tcp_max_ssthresh          tcp_timestamps
ip_default_ttl               tcp_max_syn_backlog       ...
ip_forward                   tcp_max_tw_buckets        udp_mem
...                          tcp_mem                   udp_rmem_min
neigh/                       tcp_moderate_rcvbuf       udp_wmem_min
route/                       tcp_mtu_probing
```

Многие переменные, имена которых содержат слова `rate` и `max`, используются для противодействия DoS-атакам (denial of service attacks). Подкаталог **conf** содержит переменные, устанавливаемые для каждого интерфейса. Он содержит подкаталоги **all** и **default** и отдельный подкаталог для каждого интерфейса (включая интерфейсы loopback). Каждый подкаталог содержит один и тот же набор файлов.

```
ubuntu$ cd conf/default; ls -F
accept_redirects      disable_policy          promote_secondaries
accept_redirects      disable_xfrm            proxy_arp
arp_accept            force_igmp_version      rp_filter
arp_announce          forwarding              secure_redirects
arp_filter            log_martians            send_redirects
arp_ignore            mc_forwarding           shared_media
bootp_relay           medium_id               tag
```

Например, если изменить переменную в подкаталоге **conf/eth0**, то это изменение будет относиться только к данному интерфейсу. Если изменить значение в каталоге **conf/all**, то можно ожидать, что соответствующее изменение произойдет во всех существующих интерфейсах, но не факт, что это произойдет на самом деле. Для каждой переменной существуют собственные правила принятия изменений с помощью каталога **all**. Некоторые изменения применяются к существующим значениям по правилу OR, некоторые — по правилу AND, а остальные — по правилам MAX и MIN. Кроме исходного кода ядра, этот процесс больше не описан ни в одном документе, поэтому таких неопределенных ситуаций лучше всего избегать, ограничивая изменения отдельными интерфейсами.

Если изменить переменную в каталоге **conf/default**, то новое значение будет относиться ко всем интерфейсам, которые будут конфигурироваться позднее. С другой стороны, лучше всего использовать значения, принятые по умолчанию, лишь для справки;

это позволит восстановить нормальное функционирование системы в случае неудачного конфигурирования.

Каталог **/proc/sys/net/ipv4/neigh** содержит также подкаталоги для каждого интерфейса. Файлы в каждом из этих подкаталогов управляют ARP-таблицей и выявлением "соседей" данного интерфейса по протоколу IPv6. Приведем список этих переменных; переменные, имена которых начинаются с букв gc (garbage collection — сборка мусора), определяют, каким образом устаревают и аннулируются записи в ARP-таблице.

```
ubuntu$ cd neigh/default; ls -F
anycast_delay                gc_stale_time          proxy_delay
app_solicit                  gc_thresh1             proxy_qlen
base_reachable_time          gc_thresh2             retrans_time
bade_reachable_time_ms       gc_thresh3             retrans_time_ms
delay_first_probe_time       locktime               ucast_solicit
gc_interval                  mcast_solicit          unres_qlen
```

Для того чтобы увидеть значение переменной, следует применить команду **echo** к соответствующему имени файла. Например, команда

```
ubuntu# cat icmp_echo_ignore_broadcast
0
```

показывает, что значение этой переменной равно нулю, т.е. широковещательная проверка связи не игнорируется. Для того чтобы установить это значение равным единице (и тем самым предотвратить атаку DoS типа Smurf), следует выполнить команду

```
ubuntu# sudo sh -c "echo 1 > icmp_echo_ignore_broadcasts"[18]
```

в каталоге **/proc/sys/net**.

Обычно вы зарегистрированы в той же сети, которую пытаетесь настроить, поэтому будьте осторожны! Вы можете так запутать настройки, что придется перегружать систему с консоли, а это может оказаться неудобным, если система, например, находится в Пойнт-Барроу, штат Аляска, и за окном январь. Прежде чем даже просто подумать о настройке головного компьютера (production machine), проведите тестирование на своей настольной системе.

Для того чтобы указанные изменения стали постоянными (или, говоря точнее, чтобы они восстанавливались при каждой перезагрузке системы), добавьте соответствующие переменные в каталог **/etc/sysctl.conf**, который считывается командой **sysctl** во время загрузки. Формат файла **sysctl.conf** выглядит так: *переменная=значение*, а не **echo value > variable**, как это принято в оболочке при изменении переменных вручную. Именами переменных являются относительные пути к каталогу **/proc/sys**; кроме того, если хотите, можно использовать точки, а не косые черты. Например, каждая из строк

```
net.ipv4.ip_forward=0
net/ipv4/ip_forward=0
```

в файле **/etc/sysctl/conf** отключает пересылку IP-пакетов на заданный хост.

Некоторые подкаталоги каталога **/proc** документированы лечше, чем другие. Лучше всего обратиться к разделу 7 справочной системы протокола. Например, команда

[18] Если попытаться выполнить эту команду в форме **sudo echo 1 > icmp_echo_ignore_broadcasts**, то будет сгенерировано сообщение "в разрешении отказано", потому что ваша оболочка пытается открыть выходной файл до выполнения команды **sudo**. Вы же хотите применить команду **sudo** как к команде **echo**, так и к перенаправлению. Следовательно, вы должны создать корневую подоболочку (root subshell), в которой необходимо выполнять всю команду целиком.

`man 7 icmp` документирует четыре из шести возможных вариантов. (При этом необходимо, чтобы справочные страницы о ядре системы Linux были инсталлированы заранее.)

Кроме того, полезные комментарии можно найти в файле `ip_sysctl.txt`, находящемся в дистрибутивах исходного кода ядра. Если исходный код ядра не инсталлирован, то отправьте в поисковую систему запрос ip-sysctl-txt.

## Переменные ядра, связанные с безопасностью

В табл. 14.9 описано стандартное поведение Linux-систем в отношении различных сетевых методик обеспечения безопасности. Все они кратко рассмотрены выше. Мы рекомендуем установить соответствующие параметры так, чтобы система не отвечала на широковещательные ICMP-запросы, не подчинялась директивам переадресации и не принимала пакеты, маршрутизируемые "от источника".

**Таблица 14.9. Режимы безопасности в системе Linux, заданные по умолчанию**

| Функциональная возможность | Реализация на простом узле | Реализация на шлюзе | Управляющий файл (в каталоге /proc/sys/net/ipv4) |
|---|---|---|---|
| Пересылка IP-пакетов | Отключена | Включена | **ip_forward** для всей системы **conf**/*интерфейс*/**forwarding** для каждого интерфейса [a] |
| Переадресующие ICMP-пакеты | Принимаются | Игнорируются | **conf**/*интерфейс*/**accept_redirects** |
| Маршрутизация "от источника" | Переменная | Переменная | **conf**/*интерфейс*/**accept_source_route** |
| Широковещательное тестирование | Игнорируется | Игнорируется | **icmp_echo_ignore_broadcasts** |

[a] В качестве параметра *интерфейс* может быть задано имя конкретного интерфейса или ключевое слово **all**.

## Система Linux NAT и фильтрация пакетов

В системе Linux традиционно реализуется лишь отдельная разновидность системы NAT (Network Address Translation — трансляция сетевых адресов), называемая PAT (Port Address Translation — трансляция адресов портов). В ней не используется диапазон частных адресов, как в истинной системе NAT, а все соединения коммутируются по одному адресу. Впрочем, с практической точки зрения это не имеет особого значения.

Утилита `iptables` реализует не только систему NAT, но и фильтрацию пакетов. В ранних версиях системы Linux эта функциональная возможность была относительно запутанной, но утилита `iptables` позволила намного точнее отделить систему NAT от фильтрации пакетов.

Фильтрации пакетов посвящен раздел 22.11. Если система NAT используется для обеспечения доступа в Интернет с локальных компьютеров, то при ее работе *необходимо* использовать полный набор фильтров брандмауэра. Тот факт, что "система NAT не осуществляет реальную маршрутизацию" не делает шлюз Linux NAT более безопасным, чем маршрутизатор Linux. Для краткости мы опишем только реальную конфигурацию системы NAT, однако следует помнить, что это лишь малая часть полной конфигурации.

Для того чтобы осуществлялось IP-маскирование, нужно включить перенаправление IP-пакетов и скомпилировать ядро, задав переменную ядра в каталоге `/proc/sys/net/ipv4/ip_forward` равной 1. Кроме того, необходимо вставить соответствующие модули ядра.

```
ubuntu$ sudo /sbin/modprobe iptable_nat
ubuntu$ sudo /sbin/modprobe iptable_conntrack
ubuntu$ sudo /sbin/modprobe iptable_conntrack_ftp
```

Существует много других модулей, обслуживающих соединения; их более полный список можно найти в подкаталоге **/lib/modules**.

Команда **iptables** для маршрутизации пакетов с помощью системы NAT имеет следующий вид.

```
sudo iptables -t -nat -A -POSTROUTING -0 eth1 -j SNAT -to 63.173.189.1
```

В данном примере интерфейсом для связи с Интернетом является интерфейс **eth0**. Он не появляется в этой командной строке непосредственно. Вместо него в командной строке используется его IP-адрес в качестве аргумента **--to**. Интерфейс **eth1** связан с внутренней сетью.

Оказывается, все пакеты, направляемые от интернет-хостов во внутреннюю сеть имеют IP-адрес интерфейса **eth0**. Хост, выполняющий систему NAT, получает входные пакеты, ищет их настоящие адреса, заменяет эти адреса соответствующими адресами внутренней сети и "отпускает их с миром".

# 14.13. Работа в сети под управлением системы Solaris

Система Solaris поставляется с огромным набором сценариев запуска. На одной презентации мы выиграли отрывной календарь системного администратора, в котором на каждой странице были помещены тривиальные вопросы. Вопрос для 1 января звучал так: "Назовите все файлы, которые следует открыть, если вы хотите изменить имя компьютера и IP-адрес машины, на которой выполняется система Solaris". Ответ состоял из имен шести файлов. Такая модульность представляет собой эксцентричную крайность. Однако вернемся к сетевой конфигурации системы Solaris.

## Основная сетевая конфигурация системы Solaris

Система Solaris одну часть файлов сетевой конфигурации прячет в каталоге **/etc**, а другую — в каталоге **/etc/inet**. Многие из них дублируются посредством таинственных символьных связей, благодаря которым реальные файлы могут находиться в каталоге **/etc/inet**, а связи — в каталоге **/etc**.

Для того чтобы задать имя компьютера, введите его в файл **/etc/nodename**. Изменение вступит в силу при перезагрузке компьютера. Некоторые сайты используют короткое имя компьютера, в то время как другие — полностью квалифицированное доменное имя.

🕮 Информация о переключении службы имен приводится в разделе 19.5.

Имя файла **/etc/defaultdomain** означает, что его можно использовать только в данном домене DNS, но на самом деле он задает доменное имя NIS или NIS+. Домен DNS задается в каталоге **/etc/resolv.conf**, как обычно.

Для определения порядка, в котором производится просмотр имен компьютеров в базе данных **/etc/hosts** и в службах NIS, NIS+ и DNS, система Solaris использует файл **/etc/nsswitch.conf**. Для того чтобы не усложнять загрузку, рекомендуем сначала про-

сматривать файл **hosts**, а затем службу DNS. Строка в файле **nsswitch.conf** должна выглядеть следующим образом.

```
hosts:  files dns
```

Эта конфигурация задается по умолчанию, если компьютер получает адреса от DNS-серверов с помощью протокола DHCP.

Использование сетей в системе Solaris происходит либо в традиционном режиме, либо в режиме "Network Auto-Magic" (NWAM), в котором работа в сети управляется автономно демоном **nwamd**. Режим NWAM прекрасно подходит для рабочих станций, но он предусматривает ограниченные возможности для конфигурирования и допускает только один активный интерфейс в каждый момент времени. Дальнейшее описание относится к традиционному режиму.

Для того чтобы увидеть, какой режим работы сети установлен, следует выполнить команду **svcs svc:/network/physical**. В файле конфигурации должно быть две строки: для режима NWAM и для традиционного режима ("по умолчанию"). Для переключения конфигурации необходимо выполнить команду **svcadm**. Например, следующие команды изменяют режим работы системы с NWAM на традиционный.

```
solaris$ svcs svc:/network/physical
STATE          STIME      FMRI
disabled       Mar_31     svc:/network/physical:default
online         Mar_31     svc:/network/physical:nwam
solaris$ sudo svcadm disable svc:/network/physical:nwam
solaris$ sudo svcadm enable svc:/network/physical:defauld
```

Система Solaris конфигурирует IP-адрес для каждого сетевого интерфейса с помощью файла **/etc/hostname.***интерфейс*, где *интерфейс* — это обычное имя интерфейса. Эти файлы могут содержать либо имя компьютера, которое записано в файле **hosts**, либо IP-адрес. Значение, записанное в файле **hostname.***интерфейс*, используется как параметр *адрес* команды **ifconfig**, поэтому безопаснее всего использовать адрес, даже несмотря на то, что имя файла конфигурации подразумевает, что используется имя компьютера.

В файл **hostname.***интерфейс* можно также записать любые специальные опции команды **ifconfig**. Эти опции должны находиться в той же строке, что и имя компьютера или его IP-адрес; все это порождает длинную командную строку **ifconfig**. Сценарии запуска пытаются выяснить IP-адреса всех интерфейсов, используя не соответствующие файлы **hostname**, а протокол DHCP.[19]

В исходной поставке загрузочные файлы системы Solaris содержат опции **netmask+** и **broadcast+** команды **ifconfig**. Плюсы означают, что маску подсети, по которой определяется широковещательный адрес, следует искать в файле **/etc/netmasks**. Этот файл содержит сетевые номера и соответствующие им значения масок подсетей. В этом файле должна быть представлена любая сеть, которая разделяется на подсети иначе, чем сети класса A, B или C. Рассмотрим пример файла **netmasks**.

```
# CS Department network masks database
# Network       netmask
# =====         =======
#
128.138.0.0     255.255.255.192  # default for dept.
#
```

[19] Сетевые интерфейсы в системе Solaris должны находиться в пределах области видимости команды **ifconfig plumb**, чтобы сделать их доступными. При ручном конфигурировании эту команду, возможно, придется выполнять самостоятельно.

```
128.138.192.64  255.255.255.192  # drag
128.138.192.192 255.255.255.192  # csops
128.138.193.0   255.255.255.224  # bcrg
128.138.193.32  255.255.255.224  # database
128.138.193.0   255.255.255.192  # slip
...
```

В первой строке задается стандартная маска /26 для адреса 128.138.0.0 класса B, которая затем заменяется конкретными масками, отличающимися от стандартной. Здесь перечислены все сети, даже те, которые используют стандартную маску и поэтому могут не указываться. В системах, из которых был взят этот пример, файл **netmasks** хранится на центральном узле и распределяется по всем остальным машинам. Ни на одном отдельно взятом компьютере нет интерфейсов для всех этих сетей.

В предыдущих версиях системы Solaris сетевые сценарии загрузки были записаны в файлах каталога **/etc/unit.d** (преимущественно в файлах **rootusr**, **inetinit**, **sysid.net** и **inetsvc**). В версии Solaris 10 способ управления загрузочными файлами и системными службами был коренным образом пересмотрен. Сценарии были переработаны и теперь хранятся в каталоге **/lib/svc/method**. Обзор механизма Service Management Facility в системе Solaris содержится в разделе 3.6.

Если существует файл **/etc/defaultrouter**, то предполагается, что он содержит идентификатор (который, в свою очередь, может представлять собой имя машины или адрес) стандартного шлюза и дальнейшее конфигурирование маршрутов не выполняется. Как обычно, предпочтительнее указывать адреса; если же задано имя, то для него должна существовать запись в файле **/etc/hosts** или на сервере DNS в локальной сети.

Если стандартный шлюз не указан, то система Solaris пытается запустить демон **routed** (который на самом деле называется **in.routed**), но в системе Solaris 10 и более поздних версиях необходимо запустить демон **routed** явным образом с помощью команды **svcadm enable routing/route**. Для определения текущего состояния служб используется команда **svcs route**.

Будь осторожны! Если машина имеет несколько интерфейсов или существует файл **/etc/gateways**, то демон **routed** запускается в режиме сервера ("болтливом"). Как правило, это не соответствует желаниям пользователей. Для того чтобы предотвратить "верещание", можно установить флаг "бесшумного" режима.

```
solaris# svccfg -s routing/route:default setprop routing/quiet_mode = true
```

## Примеры конфигураций системы Solaris

Рассмотрим несколько команд, необходимых для подключения сетевого интерфейса в системе Solaris и добавления маршрута к стандартному шлюзу.

```
solaris$ sudo ifconfig e1000g0 plumb
solaris$ sudo ifconfig e1000g0 192.108.21.48 netmask 255.255.255.0 up
solaris$ sudo route add default 192.108.21.254
```

Следующие примеры демонстрируют, как выяснить статус сетевого интерфейса и содержимое таблиц маршрутизации. Команды, вызываемые с помощью утилиты **sudo**, должны запускаться как корневые (root). Последний пример иллюстрирует особенность команды **route** в системе Solaris, отсутствующую в других архитектурах: аргумент **get** позволяет получить информацию о следующем переходе на пути к пункту назначения. Рассмотрим несколько вариантов, отформатированных так, чтобы их было удобно читать.

```
solaris$ ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING, MULTICAST,IPv4,VIRTUAL>
```

```
        mtu 8232 index 1 inet 127.0.0.1  netmask ff000000
e1000g0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
        index 2 inet 192.108.21.48 netmask ffffff00 broadcast 192.108.21.255
solaris$ sudo ifconfig e1000g0
e1000g0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
        index 2 inet 192.108.21.48 netmask ffffff00 broadcast 192.108.21.255
        ether 0:14:4f:e:e6:1c
```

Обратите внимание: когда команда **ifconfig** запускается как корневая, она выводит аппаратный адрес интерфейса, а когда ее выполняет пользователь, она этого не делает.

```
solaris% netstat -nr
Routing Table: Ipv4
Destination      Gateway          Flags    Ref   Use   Interface
------------     -------------    -----    ---   ---   ---------
default          192.108.21.254     UG       1    9959
192.108.21.0     192.108.21.48      U        1    4985   e1000g0
127.0.0.1        127.0.0.0          UH       1     107   lo0

solaris$ sudo route get google.com
    route to: gw-in-f100.google.com
destination: default
       mask: default
    gateway: 192.108.21.254
interface: e1000g0
    flags: <UP,GATEWAY,DONE,STATIC>
recvpipe  sendpipe   ssthresh   rtt,ms   rttvar,ms   hopcount   mtu   expire
   0         0           0         0          0          0      1500     0
```

## Конфигурирование протокола DHCP в системе Solaris

В системе Solaris существует DHCP-клиент, и она заслуживает награду за самую простую и разумную процедуру конфигурирования этого клиента.

```
solaris$ sudo ifconfig интерфейс dhcp
```

Это действительно работает! Команда **inconfig** вызывает программу **dhcpagent**, чтобы получить параметры интерфейса от сервера DHCP и конфигурировать этот интерфейс в соответствии с ними. В командной строке **ifconfig** можно указывать разные опции, чтобы задать интерфейс первичным, установить тайм-аут, увеличить срок аренды параметров или вывести состояние интерфейса. Для того чтобы аннулировать конфигурацию DHCP-клиента вручную, необходимо выполнить следующую команду.

```
solaris$ sudo ifconfig интерфейс drop
```

Все это прекрасно, но иногда необходимо автоматически опрашивать сервер DHCP на этапе начальной загрузки. Это можно сделать, вообще не указав никаких файлов конфигурации для интерфейса (тем самым полагаясь на автоматическое конфигурирование, как в программе NetworkManager в системе Linux) либо создав файл **/etc/dhcp.***интерфейс* в дополнение к файлу **/etc/hostname.***интерфейс*. Если хотите, можете записать в файл **dhcp.***интерфейс* дополнительные опции, которые будут переданы команде **ifconfig**.

Для того чтобы интерфейс можно было активизировать, должен, как и раньше, существовать файл **hostname.***интерфейс*, хотя его можно оставить пустым. Если этот файл не пуст, то сценарии загрузки сначала статически конфигурируют интерфейс, используя содержимое этого файла, а затем изменят его конфигурацию с помощью сервера DHCP.

Программа **dhpagent** управляет интерфейсом в соответствии с протоколом DHCP. Кроме выполнения других задач, она согласовывает с сервером продление срока аренды и отмену арендованных параметров, когда они уже не нужны. Если конфигурация интерфейса, который был настроен с помощью DHCP-сервера, впоследствии изменяется вручную, то программа **dhpagent** прекращает им управлять.

Программа **dhcpagent** собирает параметры, арендованные у DHCP-сервера (стандартный маршрут, домен, имена серверов и так далее), но не влияет на большинство из них напрямую. Вместо этого она делает их доступными с помощью команды **dhcpinfo**. Сценарии управления службами обращаются к команде **dhcpinfo** за разной информацией, которая используется в качестве аргументов команды **route**, записывается в файл **resolv.conf** и т.д.

Программа **dhcpagent** направляет ошибки в системный журнал с помощью демона и ранжирует информацию по приоритетам. Вывести содержание системного журнала при заданном уровне отладки можно с помощью флага **-d**.

Для того чтобы увидеть конфигурацию конкретного интерфейса, можно просмотреть файлы в каталоге **/etc/dhcp**. Однако существование файла *интерфейс*.**dhc** еще не означает, что программа **dhcpagent** в настоящее время управляет данным интерфейсом, — срок аренды мог закончиться.

## Команда ndd: протокол TCP/IP и настройка интерфейса в системе Solaris

Команда **ndd** в системе Solaris позволяет изменить конфигурацию стека протоколов TCP/IP в выполняющейся системе. Возможно, "изменить конфигурацию" — слишком сильное выражение. На самом деле каждый модуль предоставляет доступ к своим параметрам для проверки и, в некоторых случаях, для уточнения "на ходу".

Основная синтаксическая конструкция этой команды выглядит следующим образом.

**ndd [-set]** *устройство* **?** | *переменная* [*значение*]

Если указан аргумент **?** (его следует защитить от оболочки символом **\?**), то команда **ndd** возвращает список переменных, распознаваемых драйвером указанного устройства. Если задать имя *переменной*, то команда **ndd** вернет ее значение. Если же использовать флаг **-set** и задать *значение*, то оно будет присвоено указанной *переменной*.

К сожалению, на справочной странице, посвященной команде **ndd**, не перечисляются все возможные имена устройств и ничего не говорится о том, что для доступа к одним устройствам (например, **ip** и **hme**) команда **ndd** должна выполняться как корневая, а для доступа к другим (например, **tcp** и **udp**) — нет. Табл. 14.10 представляет собой краткую шпаргалку по работе с командой **ndd**.

**Таблица 14.10. Устройства, с которыми может работать команда ndd в системе Solaris**

| Устройство | Описание | Имена переменных |
|---|---|---|
| **/dev/tcp** | Переменные протокола TCP | `tcp_*` |
| **/dev/udp** | Переменные протокола UDP | `udp_*` |
| **/dev/ip** | Переменные протокола IP | `ip_* è ip6_*` |
| **/dev/icmp** | Переменные протокола ICMP | `icmp_*` |
| **/dev/rawip** | Идентично устройству /dev/icmp | `icmp_*` |
| **/dev/arp** | Переменные протокола ARP | `arp_*` |

Связанные с интерфейсами имена переменных в категории **/dev/ip** управляют перенаправлением (IP-forwarding) на конкретные сетевые интерфейсы. Например, переменная `e1000g0:ip_forwarding` управляет перенаправлением на интерфейс **/dev/e1000g0**. Наряду с ней существует и глобальная переменная `ip_forwarding`.

Если у вас есть доступ к машине HP-UX, выполните команду **ndd** с флагом **-h** (для получения справки), и вы увидите имена устройств, переменных и назначение каждой переменной. Многие имена переменных в системах HP-UX и Solaris совпадают, поэтому можно ограничиться минимальной **man**-страницей, посвященной команде **ndd**.

С помощью команды **ndd** можно также устанавливать такие опции, связанные с интерфейсами, как скорость связи, автосогласование и поддержка jumbo-пакетов. Для этого необходимо применить команду **ndd** непосредственно к файлу устройства, связанного с интерфейсом (например, **/dev/e1000g0**). К сожалению, способ, которым система Solaris реализует эту возможность, делает имена параметров конфигурации зависимыми от конкретного драйвера, поэтому не существует универсального рецепта, например, для фиксации скорости работы сетевого интерфейса на уровне 100 Мбит/с.

Для того чтобы изменить скорость, необходимо идентифицировать соответствующие переменные, задающие "производительность" (обычно они называются `*_cap`), и отключить их (установить равными нулю). Кроме того, следует отключить переменную `*autoneg_cap`. Например, следующий сценарий устанавливает с помощью переменной **/dev/e1000g0** полнодуплексный режим и скорость, равную 100 Мбит/с.

```
#!/bin/sh
ndd -set /dev/e1000g0 adv_autoneg_cap 0
ndd -set /dev/e1000g0 adv_1000fdx__cap 0
ndd -set /dev/e1000g0 adv_100fdx_cap 1
ndd -set /dev/e1000g0 adv_100hdx_cap 0
ndd -set /dev/e1000g0 adv_10fdx_cap 0
ndd -set /dev/e1000g0 adv_10hdx_cap 0
```

## Безопасность в системе Solaris

В табл. 14.11 описано поведение системы Solaris в некоторых щекотливых ситуациях, касающихся безопасной работы в сети. Краткое описание этих ситуаций приведено в разделе 14.8. Большинство этих настроек можно изменить с помощью команды **ndd**.

**Таблица 14.11. Поведение системы Solaris, связанное с сетевой безопасностью**

| Функциональная возможность | По умолчанию | Переменная команды ndd |
|---|---|---|
| Перенаправление IP-пакетов | Запрещено | `ip_forwarding` |
| Переадресующие ICMP-пакеты | По ситуации | Не может изменяться [a] |
| Направленная маршрутизация | Игнорируется | `ip_forward_src_routed` |
| Широковещательные ping-пакеты (ответ) | Разрешено | `ip_respond_to_echo_broadcast` |
| Широковещательные ping-пакеты (перенаправление) | Запрещено | `ip_forward_directed_broadcast` |

[a] Можно модифицировать только время существования.

## Брандмауэры и фильтрация в системе Solaris

Как уже говорилось, не следует использовать системы UNIX, Linux или Windows в качестве брандмауэра или шлюза NAT. Для этого лучше использовать специальное ап-

паратное обеспечение. В системе Solaris этому правилу следовать легче, поскольку в нее вообще не входит никакое программное обеспечение для фильтрации. Тем не менее в большинство дистрибутивов сейчас включается свободно распространяемое программное обеспечение IPFilter, созданное Дарреном Ридом (Darren Reed). Оно представляет собой хороший выбор для фильтрации IP-пакетов в системе Solaris.

Пакет IPFilter реализует фильтрацию IP-пакетов и прозрачное перенаправление портов. Он представляет собой открытое, свободно распространяемое программное обеспечение и работает на компьютерах, имеющих архитектуру SPARC или Intel. Пакет IPFilter содержит утилиты **ipf** (для конфигурирования брандмаэура), **ipfstat** (для распечатки инсталлированных правил фильтрации) и **ipnat** (для реализации механизма NAT).

Детали функционирования пакета IPFilter приведены в разделе 22.13, а пока обсудим функциональные возможности механизма NAT, которыми обладает пакет IPFilter.

## Механизм NAT в системе Solaris

Для того чтобы механизм NAT заработал, вы должны сообщить ядру внутренние и внешние адреса, а также диапазон портов, которые будут использоваться для расширения адресного пространства. Обсуждение общих принципов механизма NAT и способов перехода из частного в публичное адресное пространство содержится в разделе 14.4.

Для настройки механизма NAT необходимо указать правила для команды **ipnat**. Эти правила аналогичны правилам, установленным для утилиты **ipf** при реализации пакета фильтрации. Однако следует быть осторожным: как и правила **ipf**, правила **ipnat** упорядочены, но в обратном порядке. Просто будьте внимательными: выбирается *первое* применимое правило, а не последнее.

Ниже приведено несколько правил **ipnat**. Для того чтобы привести их в действие во время загрузки, они записаны в файле **/etc/ipf/ipnat.conf**.

```
map eth1 192.168.1.0/24 -> 128.138.198.0/26 portmap tcp/udp 20000:65000
map eth1 192.168.1.0/24 -> 128.138.198.0/26
```

Мы предположили, что **eth1** — это наш интерфейс для выхода в Интернет и что наша внутренняя сеть пронумерована как частное адресное пространство класса C. Эти правила отображают адреса сети /24 в адреса сети /26. Поскольку сеть /26 может вместить в себя только четверть машин, которые могут поместиться в сети /24, потенциально возможно, что адреса назначения могут выйти за пределы этой конфигурации. Однако пункт `portmap` расширяет диапазон адресов, разрешая использовать каждый из 45 тыс. разных портов.

Первое правило, сформулированное выше, охватывает весь трафик по протоколам TCP и UDP, но не учитывает протокол ICMP, поскольку этот протокол не использует концепцию порта. Второе правило перехватывает ICMP-сообщения и пытается отправить их назад на правильную машину. Если ядро не может однозначно определить адресата конкретного ICM-сообщения, оно посылает его как широковещательный пакет; машины, которые получат его вне контекста, могут его просто игнорировать.

На своем домашнем компьютере вы можете иметь только один IP-адрес, который присвоен вам вашим ISP- или DHCP-сервером. Если вашему компьютеру присвоен статический IP-адрес, то в строке `map` просто присвойте в качестве целевой сети пункт назначения /32 и задайте достаточно широкий диапазон портов, чтобы удовлетворить потребности всех ваших локальных машин. Если при каждом соединении вы получаете новый динамический адрес, то в строке `map` следует указать пункт назначения 0/32. Это позволит утилите **ipnat** считывать адрес непосредственно с сетевого интерфейса.

В качестве примера приведем строку, которую можно использовать для единственного динамически присваиваемого адреса.

```
map eth1 192.168.1.0/24 -> 0/32 portmap tcp/ip 20000:65000
```

Для проверки этой конфигурации выполните следующую команду.

```
solaris$ sudo ipnat -CF -f /etc/ipf/ipnat.config
```

Эти опции сначала отменяют все существующие правила, а затем загружают полный набор правил из файла **/etc/ipf/ipnat.conf**.

## Особенности сетевого конфигурирования

Результат выполнения команды **ifconfig -a** зависит от того, выполняется ли она как корневая команда или как команда пользователя. В первом случае, помимо IP-адресов и параметров, отображаются также адреса Ethernet канального уровня.

Система Solaris разрешает менять адреса канального уровня (MAC-адреса) сетевого интерфейса с помощью команды **ifconfig** и семейства адресов **ether**. Эта возможность может оказаться полезной, если вам необходимо проникнуть в беспроводную сеть с ограниченными MAC-адресами.

# 14.14. Работа в сети под управлением системы HP-UX

Конфигурирование операционной системы HP-UX не вызывает трудностей: все параметры конфигурации хранятся в файле **/etc/rc.config.d/netconf**. Значения параметров из этого файла (а также других файлов в каталоге **rc.config.d**) считываются в переменные окружения на этапе загрузки и используются сценарием **/sbin/rc**. Файл **netconf** содержит многочисленные комментарии, которые сообщают вам, что именно должно быть записано в ту или иную переменную и для чего.

## Базовое конфигурирование сетей в системе HP-UX

Для того чтобы назначить машине имя и сконфигурировать ее первый сетевой интерфейс, отредактируйте файл **netconf**, присвоив значения следующим переменным.

```
HOSTNAME
INTERFACE_NAME[0]

IP_ADDRESS[0]
SUBNET_MASK[0]
```

Например:

```
HOSTNAME=disaster
INTERFACE_NAME[0]=lan0
IP_ADDRESS[0]=192.108.21.99
SUBNET_MASK[0]=255.255.255.0
```

Второй интерфейс будет иметь индекс 1. О его наличии свидетельствует значение переменной NET_CARDS, равное 2.

Файл **netconf** содержит также переменные для конфигурирования статических маршрутов и запуска демона маршрутизации. Для того чтобы задать стандартный маршрут, необходимо задать значения следующих переменных.

```
ROUTE_DESTINATION[0]=default
ROUTE_MASK[0]=""
ROUTE_GATEWAY[0)=192.108.21.254
ROUTE_COUNT[0]=1
```

Переменная ROUTE_MASK необходима для сети, в которой маска подсети отличается от стандартной маски, используемой в данном классе адресов. Переменная ROUTE_COUNT должна равняться нулю, если в качестве шлюза используется локальный компьютер, и единице, если шлюз расположен на удаленной машине. Параметры остальных статических маршрутов задаются в переменных ROUTE_* с индексами [1], [2] и т.д. Эти значения передаются непосредственно команде **route**. Например, переменной ROUTE_DESTINATTON можно присвоить ключевое слово **default**, как показано выше, а также выражения **net** *адрес_сети* или **host** *адрес_машины*.

В системе HP-UX используется демон **gated**, а не **routed**. Для использования демона **gated** необходимо задать переменную GATED равной 1, а в массив GATED_ARGS записать список аргументов, передаваемых демону при его выполнении. Подробнее демон **gated** описывается в главе 15. Много полезной информации содержится на справочной странице, посвященной маршрутизации (**man routing**).

Многие поля в файле **netconf** могут содержать либо имя компьютера, либо IP-адрес. Если указано имя компьютера, то оно обязательно должно присутствовать в файле **/etc/hosts**. На этапе загрузки система HP-UX просматривает только файл **/etc/hosts** и не использует никаких других поисковых механизмов. В этом файле сначала должны перечисляться полностью квалифицированные доменные имена, затем короткие имена и псевдонимы.

В системе HP-UX используется команда **lanscan**, с помощью которой можно получить информацию о сетевых интерфейсах, существующих в компьютере. В данной ситуации работает команда **ifconfig** *интерфейс*, а не **ifconfig -a**. Имена сетевых интерфейсов начинаются с префикса "lan" или "snap". Префикс "lan" обозначает канальный уровень Ethernet, а префикс "snap" — спецификацию IEEE 802.3. Первый интерфейс обозначается как lan0, второй — lanl и т.д.

В системе HP-UX, как и в системе Solaris, принята концепция "подключения" интерфейсов. Однако все интерфейсы "подключаются" автоматически, когда команда **ifconfig** назначает им IP-адреса.

SMH — это системная административная утилита системы HP-UX, значительно облегчающая управление системой UNIX. Она оснащена меню и может использоваться для конфигурирования сетевых интерфейсов, а также выполнения многих других административных задач.

## Примеры конфигураций в системе HP-UX

Для того чтобы вручную подключить сетевой интерфейс в системе HP-UX и задать стандартный маршрут, нужно выполнить команды следующего вида.

```
hp-ux$ sudo ifconfig lan0 192.108.21.99 netmask 0xffffff00
hp-ux$ sudo route add default 192.108.21.254 1[20]
```

[20] В версии HP-UX поле для счетчика переходов не требуется; если оно явно не указано, то по умолчанию равно нулю. В предыдущих версиях поле для счетчика было обязательным.

Команда **lanscan** выводит список сетевых интерфейсов, существующих в системе, и параметры драйверов, которые ими управляют. Команда **lanscan -v** отображает чуть больше информации. Ниже показан ряд примеров, слегка измененных так, чтобы поместиться на странице. Поле MAC со значением ETHER означает, что устройство должно называться lan0, а не snap0. Команда **ifconfig** подтверждает, что это соответствует действительности.

```
$ lanscan
Hardware    Station    Crd    Hdw     Net-Int    NM    MAC    HP-DLPI   DLPI
Path        Address    In#    State   NamePPA    ID    Type   Support   Mjr#
8/0/20/0    0x001      0      UP     lan0 snap0  1     ETHER  Yes       130

$ ifconfig lan0
lan0:  flags=843<UP,BROADCAST,RUNNING,MULTICAST> inet 192.108.21.99
    netmask ffffff00 broadcast 192.108.21.255

$ ifconfig snap0
ifconfig: no such interface
```

Команда **netstat -i** отображает имена сетевых интерфейсов, а команда **netstat -nr** выводит таблицы маршрутизации.

```
$ netstat -i
Name      Mtu    Network       Address             Ipkts    Opkts
lan0      1500   192.108.21.0  disaster.xor.com    6047     3648
lo0       4136   127.0.0.0     localhost.xor.com    231      231

$ netstat -nr
Routing tables
Dest/Netmask      Gateway         Flags    Refs    Use      Int    Pmtu
127.0.0.1         127.0.0         UH       0       231      lo0    4136
192.108.21.99     192.108.21.99   UH       8                lan0   4136
192.108.21.0      192.108.21.99   U        2         0      lan0   1500
127.0.0.0         127.0.0.1       U        0         0      lo0    4136
default           192.108.21.254  UG       0         0      lan0   1500
```

Программа **lanadmin** отображает статистику сетевого трафика для каждого обнаруженного интерфейса. С ее помощью можно управлять интерфейсами и следить за их работой. Эта программа оснащена меню, облегчающим доступ к требуемой информации. Ниже показан пример, в котором демонстрируются данные об интерфейсе lan0.

```
% lanadmin
  LOCAL AREA NETWORK ONLINE ADMINISTRATION,  Version 1.0
         Copyright 1994 Hewlett Packard Company.
                All rights are reserved.

Test Selection mode.

   lan        = LAN Interface Administration
   menu       = Display this menu
   quit       = Terminate the Administration
   terse      = Do not display command menu
   verbose    = Display command menu

Enter command: lan

LAN Interface test mode. LAN Interface PPA Number = 0
clear         = Clear statistics registers
```

```
display      = Display LAN Interface status/statistics
end          = End LAN Interface Admin., go up 1 level
menu         = Display this menu
ppa          = PPA Number of the LAN Interface
quit         = Terminate the Admin,  return to shell
reset        = Reset LAN Interface, execute selftest
specific     = Go to Driver specific menu

Enter command: display

LAN INTERFACE STATUS DISPLAY
 Thu, Mar 2,2000    00:41:24
PPA Number                      = 0
Description                     = lan0 Intel PCI Pro 10/100Tx Server Adapter
Type (value)                    = ethernet-csmacd(6)
MTU Size                        = 1500
Speed                           = 100000000
Station Address                 = 0x00306eea9237
Administration Status (value)   = up(1)
Operation Status (value)        = up(1)
...
Inbound Unicast Packets         = 45691
Inbound Non-Unicast Packets     =  2630
...
Deferred Transmissions          = 0
Late Collisions                 = 0
Excessive Collisions            = 0
...
```

## Конфигурирование протокола DHCP в системе HP-UX

Как и в случае других параметров сетевой конфигурации, включить протокол DHCP на этапе загрузки можно, задав переменные в файле **/etc/rc.config.d/netconf**. В данном случае переменные хранятся в массиве DHCP_ENABLE: индекс [0] означает первый интерфейс, индекс [1] — второй и т.д. Например, запись

```
DHCP_ENABLE[C]=1
```

переводит первый интерфейс в режим DHCP. Интерфейс получит свой IP-адрес, маску подсети и другие сетевые параметры от DHCP-сервера, расположенного в локальной сети. Если задать переменную равной нулю, то протокол DHCP будет отключен; в этом случае вам придется назначать интерфейсу статический адрес в файле **netconf**. Если запись DHCP_ENABLE отсутствует, считается, что соответствующая переменная по умолчанию равна единице.

Основную задачу по взаимодействию с DHCP-сервером выполняет сценарий **/sbin/auto_parms**. Программа **dhcpdb2conf** записывает параметры DHCP, записанные сценарием **auto_parms** в файл **netconf**, из которого извлекается конфигурационная информация на этапе загрузки.

## Динамическое переконфигурирование и настройка в системе HP-UX

Как и в системе Solaris, для настройки различных сетевых параметров (более 100) в системе HP-UX можно применять команду **ndd**. В интерактивном режиме команда **ndd**

меняет значения "на ходу". Для того чтобы изменения стали постоянными, следует добавить их в файл **/etc/rc.config.d/nddconf**, который считывается на этапе загрузки.

🕮 Более подробно команда **ndd** описана в разделе 14.13.

В системе HP-UX довольно полезной является опция **-h** (вызов справки) команды **ndd**. Если не указаны дополнительные аргументы, то команда **ndd -h** выводит список всех настраиваемых параметров. Если указано имя переменной, то команда **ndd -h** отображает информацию о назначении переменной, а также о ее минимальном и максимальном значениях и значении, задаваемом по умолчанию. Рассмотрим пример.

```
$ ndd -h | grep source
ip_forward_src_routed - Controls forwarding of source routed packets

$ ndd -h ip_forward_src_routed
ip_forward_src_routed:
  Set to 1 to forward source-routed packets; set to 0 to
  disable forwarding.  If disabled,  an ICMP Destination
  Unreachable message is sent to the sender of source-
  routed packets needing to be forwarded.   [0,1]  Default:  1
```

Вывод результатов выполнения команды **ndd** свидетельствует о том, что в данной версии HP-UX по умолчанию поддерживается направленная маршрутизация пакетов. Это может быть требованием ядра, но фактически файл **/etc/rc/config.d/nddconfig** позволяет отключить эту возможность.

```
TRANSPORT_NAME[2]=ip
NDD_NAME[2]=ip_forward_scr_routed
NDD_VALUE[2]=0
```

Двойки в этих записях означают, что в файле **nddconfig** задается значение третьей из десяти возможных переменных. Если вы захотите изменить следующую переменную, то будет достаточно скопировать эти три строки и заменить индекс 2 на 3. К сожалению, с помощью файла **nddconf** можно задать только десять параметров.

Для того чтобы просмотреть и модифицировать значение переменной `ip_forward_src_routed`, воспользуемся командами **ndd -get** и **ndd -set** (их синтаксическая конструкция слегка отличается от принятой в системе Solaris).

```
$ ndd -get /dev/ip ip_forward_src_routed
0
$ sudo ndd -set /dev/ip ip_forward_src_routed 1
$ ndd -get /dev/ip ip_forward_src_routed
1
```



В табл. 14.12 описано стандартное поведение системы HP-UX в ситуациях, связанных с безопасностью работы в сети. Краткое описание последствий этого поведения приведено в разделе 14.8. Большинство настроек можно изменить с помощью команды **ndd**.

**Таблица 14.12. Поведение системы HP-UX, связанное с сетевой безопасностью**

| Функциональная возможность | По умолчанию | Управление с помощью команды ndd |
|---|---|---|
| Перенаправление IP-пакетов | Динамическое [а] | Устанавливает параметр `ip_forwarding`: 0 — отключить, 1 — включить, 2 — динамический режим |
| Переадресующие ICMP-пакеты | По ситуации [б] | Для того чтобы отключить эту возможность, устанавливает параметр `ip_ire_redirect_interval` равным 0 |

*Окончание табл. 14.12*

| Функциональная возможность | По умолчанию | Управление с помощью команды ndd |
|---|---|---|
| Направленная маршрутизация | Игнорируется | Для того чтобы включить эту возможность, устанавливает параметр `ip_forward_src_routed` равным 1 |
| Широковещательные ping-пакеты (перенаправление) | Игнорируется | Устанавливает параметр `ip_forward_directed_broadcast` |
| Широковещательные ping-пакеты (ответ) | Блокируется | Устанавливает параметр `ip_respond_to_echo_broadcast` |

[a] Если параметр больше единицы, то интерфейс включен, иначе — выключен.

[б] Записи о перенаправлении по умолчанию хранятся до пяти минут.

Как и система Solaris, система HP-UX включает в себя пакет IPFilter для фильтрации и реализации механизма NAT, разработанный Дарреном Ридом. Этот пакет описан в разделах 14.13 и 22.13. В системе HP-UX пакет IPFilter ничем не отличается от своего аналога в системе Solaris, хотя система HP-UX конфигурирует его при загрузке несколько иначе. Вместо использования команды **svcadm** для подключения пакета IPFilter, в системе HP-UX следует редактировать файл **/etc/rc.config.d/ipconfig** и включить требуемые опции. Файлы конфигурации для утилит **ipf** и **ipnat** должны находиться в каталоге **/etc/opt/ipf**, а не **/etc/ipf**.

Версия демона **inetd** в системе HP-UX содержит встроенные функции работы с протоколом TCP, которые можно конфигурировать посредством файла **/var/adm/inetd.sec**.

Если вас интересует, как именно система HP-UX может оказаться незащищенной, прочитайте статью Кевина Стивса (Kevin Steves) о действиях, которые необходимо предпринять, чтобы превратить систему HP-UX 11 в крепость, стоящую на входе в незащищенную сеть: `tynyurl.com/5sffy2`. Этот документ немного устарел (он опубликован в 2002 г.), но в нем прекрасно описаны все функциональные возможности, предоставляемые системой HP-UX, которые следует отключить, если вы хотите защитить свой компьютер в открытой среде Интернета. Мы бы хотели, чтобы появились аналогичные документы, посвященные другим операционным системам.

## 14.15. Работа в сети под управлением системы AIX


Вместо того чтобы хранить информацию о сетевой конфигурации в текстовых файлах или сценариях, система AIX скрывает ее в базе данных атрибутов и значений Object Data Manager (ODM). Ассоциацию этих списков свойств с конкретными устройствами (на самом деле, с экземплярами устройств) и связывание драйверов с информацией о конфигурации осуществляет другой слой.

В общих чертах база данных Object Data Manager описана в разделе 13.6. Ее общая схема довольно сложная. Кроме того, она обеспечивает доступ к сетевой конфигурации на разных уровнях. В табл. 14.13 приведено несколько команд системы AIX для задания сетевых адресов интерфейсов. В основном, они отличаются тем, что влияют либо на конфигурацию во время выполнения системы, либо на конфигурацию во время загрузки, либо на обе конфигурации.

**Таблица 14.13. Восемь способов задания IP-адресов интерфейса в системе AIX**

| Команда | Действует ли на текущую конфигурацию? | Действует ли на конфигурацию загрузки? |
|---|---|---|
| `smitty mktcpip` (и заполняет форму) | Да | Да |
| `mktcpip -i en3 -a 192.168.0.1` | Да | Да |
| `chdev -l en3 -a netaddr=192.168.0.1` | Да | Да |
| `chdev -l en3 -a netaddr=192.168.0.1 -P` | Нет | Да |
| `chdev -l en3 -a netaddr=192.168.0.1 -T` | Да | Нет |
| `ifconfig en3 inet 192.168.0.1` | Да | Нет |
| `odmchange -o CuAt -q'name=en3 AND attribute=netaddr'<config`[a] | Нет | Да |
| `echo 'Hey! I set the network address!'` | Нет | Нет |

[a] Команда **odmchange** требует на вход список атрибутов. Задать их значения в командной строке невозможно.

Точнее говоря, команда **mktcpip** делает немного больше, чем обычная установка параметров конфигурации, — она также запускает сценарий **rc.tcpip** для запуска соответствующего сетевого демона.

Для настройки сетевой конфигурации существует довольно полный набор инструментов под названием SMIT, который вы можете и должны использовать в большинстве случаев. Опции конфигурации протокола TCP/IP можно найти в пункте меню Communications Applications and Services.

Большинство системных администраторов никогда не испытывают необходимости углубляться ниже уровня **chdev/lsattr**. Однако этот уровень может оказаться полезным для просмотра списка важных параметров конфигурации устройства. Например, следующий запрос демонстрирует конфигурируемые параметры для сетевого интерфейса en3.

```
aix$ lsattr -H -E -l en3
attribute       value           description                         settable
alias4                          IPv4 Alias including Subnet Mask      True
alias6                          IPv6 Alias including Prefix Length    True
arp             on              Address Resolution Protocol (ARP)     True
authority                       Authrozed Users                       True
broadcast       192.168.10.255  Broadcast Address                     True
mtu             1500            Maximum IP Packet Size                True
netaddr         192.168.10.11   Internet Address                      True
netaddr6                        IPv6 Internet Address                 True
netmask         255.255.255.0   Subnet Mask                           True
prefixlen                       Prefix Lenght for IPv6 Address        True
remmtu          576             Max. Packet Size for REMOTE Nets      True
security        none            Security Level                        True
state           up              Current Interface Status              True
tcp_mssdflt                     Set TCP Maximum Segment Size          True
tcp_nodelay                     Enable/Disable TCP_NODELAY Option     True
tcp_recvspace                   Set Socket Buffer Space for Receiving True
tcp_sendspace                   Set Socket Buffer Space for Sending   True
```

Флаг **-H** позволяет выводить столбцы с заголовками, флаг **-E** запрашивает текущие ("настоящие", в отличие от заданных по умолчанию) значения, а флаг **-I** идентифицирует тестируемое устройство. Многие устройства, которыми можно управлять с помощью

команды **chdev** и других команд, не перечислены в каталоге **/dev**. Для того чтобы увидеть полный список устройств, выполните команду **lsdev -C**.

Для того чтобы задать значение, используйте команду **chdev**. Например, для того чтобы задать значение MTU для интерфейса **en3** больше 1450, выполните следующую команду.

```
aix$ sudo chdev -l en3 mtu=1450
```

## Команда no: настройка сетевых параметров в системе AIX

Система AIX разделяет системные опции протокола TCP/IP на отдельные непересекающиеся и постоянные пары "атрибут/значение", доступ к которым обеспечивает команда **no**, а не **chdev**. (Отличие заключается в том, что команда **no** предназначена для конфигурирования системы, а команда **chdev** — для конфигурирования конкретных драйверов или устройств.)

Для того чтобы увидеть полный список всех доступных переменных (в настоящее время их больше 125), выполните команду **no -a**. Некоторые переменные, относящиеся к безопасности, перечислены в табл. 14.14.

**Таблица 14.14. Переменные для настройки системы AIX, связанные с протоколами TCP/IP и безопасностью**

| Команда | Описание | Значение по умолчанию |
|---|---|---|
| bcastping | Разрешает ответ на широковещательные ping-пакеты | 0 |
| directed_broadcast | Разрешает перенаправление широковещательных пакетов | 0 |
| ipforwarding | Разрешает перенаправление IP-пакетов | 0 |
| ipignoreredirects | Игнорирует перенаправляющие ICMP-пакеты | 0[a] |
| ipsrcrouteforward | Разрешает отправлять IP-пакеты с заданным маршрутом | 1[a] |
| ipsrcrouterecv | Разрешает принимать IP-пакеты с заданным маршрутом | 0 |
| ipsrcroutesend | Блокирует IP-пакеты с заданным маршрутом | 1 |

[a] Изменять не рекомендуется.

Для того чтобы установить переменную, используйте следующую команду.

**no -p -o** *переменная=значение*

Например, для того чтобы предотвратить перенаправление пакетов с заданным маршрутом стеком протоколов TCP/IP, выполните следующую команду.

```
aix$ sudo no -p ipsfcrouteforward=0
```

Опция **-p** вводит изменения в действие как немедленно, так и после перезагрузки.

# 14.16. Рекомендуемая литература


- Stevens W. Richard. *TCP/IP Illustrated, Volume One: The Protocols*. Reading, MA: Addison-Wesley, 1994.
- Wright Gary R. and W. Richard Stevens. *TCP/IP Illustrated, Volume Two: The Implementation*. Reading, MA: Addison-Wesley, 1995.

Эти два тома являются прекрасным исчерпывающим руководством по протоколам TCP/IP, хотя уже немного устарели.

- Stevens W. Richard. *UNIX Network Programming.* Upper Saddle River, NJ: Prentice Hall, 1990.
- Stevens W. Richard. *UNIX Network Programming, volume 1: The Sockets Networking API (3rd Edition).* Upper Saddle River, NJ: Prentice Hall, 2003.
- Stevens W. Richard. *UNIX Network Programming, volume 2: Interprocess Communications (2nd edition)* Upper Saddle River, NJ: Prentice Hall, 1998.

  Эти книги являются библиями для студентов, изучающих сетевое программирование. В первом издании описан только интерфейс сокетов Беркли. В третьем издании рассмотрены также интерфейс STREAMS и стандарт IPv6. Все три издания написаны в четком и понятном стиле Рича Стивенза.
- Tanenbaum Andrew. *Computer Networks (4th Edition).* Upper Saddle River, NJ: Prentice Hall PTR, 1996.

  Эта книга по-прежнему является классическим учебником по сетям. Она содержит подробнейшее описание физического и канального уровней стека сетевых протоколов. Предыдущие издания были ориентированы на протоколы ISO, но в последнем издании описана современная структура сети Интернет.
- Salus Peter H. *Casting the Net, From ARPANET to INTERNET and Beyond.* Reading, MA: Addison-Wesley, 1995.

  Это захватывающая история о том, как сеть ARPANET превратилась в Интернет. Книга написана историком, который многие годы общался с профессионалами UNIX, поэтому кажется, будто ее писал один из них!
- Comer Douglas. *Internetworking with TCP/IP, vol. 1: Principles, Protocols, and Architectures, 4th Edition.* Upper Saddle River, NJ: Prentice Hall, 2000[21].

  Книги Дугласа Камера долгое время считались эталонными справочниками по протоколам TCP/IP. Новое издание содержит описание современных сетевых технологий, а также всех протоколов семейства TCP/IP. Оно предназначено в качестве учебного пособия для студентов и является хорошим источником справочного материала.
- Hunt Craig. *TCP/IP Network Administration (3rd Edition).* Sebastopol, CA: O'Reilly & Associates, 2002.

  Эта книга ориентирована на администраторов UNIX-систем. Половина книги посвящена протоколам TCP/IP, а остальной материал касается средств UNIX более высокого уровня, таких как электронная почта и удаленный доступ.
- Farrel Adrian. *The Internet and Its Protocols: A Comparative Approach.* San-Francisco, CA: Morgan Kaufmann Publishers, 2004.
- Kozierak Charles M. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference.* San-Francisco, CA: No Starch Press, 2005.

  Превосходное собрание документов об истории Интернета и его разнообразных технологиях можно найти на веб-сайте `isoc.org/internet/history`.

[21] Дуглас Э. Камер. *Сети TCP/IP, т. 1: Принципы, протоколы и структура, 4-е изд.* — ИД "Вильямс", 2002. — *Примеч. ред.*

# 14.17. Упражнения

14.1. Почему прием директив переадресации протокола ICMP может привести к несанкционированному доступу в сеть?

14.2. Что такое значение MTU сетевого канала? Что произойдет, если это значение будет слишком большим? А слишком низким?

14.3. ★ Сеть 134.122.0.0/16 разбита на подсети с длиной маски 19.

а) сколько будет подсетей /19? Перечислите их. Каковы их сетевые маски?

б) сколько узлов может быть в каждой подсети?

в) определите, какой подсети принадлежит адрес 134.122.67.124?

г) каким будет широковещательный адрес каждой подсети?

14.4. ★ С узла 128.138.2.4 в сети 128.138.2.0/24 нужно послать пакет узлу 128.138.129.12 в сети 128.138.129.0/24. Предполагается, что

- узел 128.138.2.4 имеет стандартный шлюз 128.138.2.1;
- узел 128.138.2.4 только что загрузился и еще не отправлял и не принимал пакеты;
- все остальные компьютеры сети работают уже долгое время;
- маршрутизатор 128.138.2.1 имеет прямое соединение с узлом 128.138.129.1, который является шлюзом в подсеть 128.138.129.0/24.

а) перечислите все этапы отправки пакета. Покажите исходные и целевые MAC- и IP-адреса передаваемых пакетов.

б) если бы адрес сети был 128.138.0.0/16, изменился бы ответ? Если да, то каким бы он был?

в) если бы сеть 128.138.2.0 имела маску /26, а не /24, изменился бы ответ? Если да, то каким бы он был?

14.5. ★ Срок аренды параметров задается на сервере DHCP. Допустим, что количество IP-адресов превышает количество потенциальных клиентов. Можете ли вы продлить срок аренды на как можно дольше (например, на несколько недель)? Почему да или почему нет? Что можно сказать о других параметрах протокола DHCP?

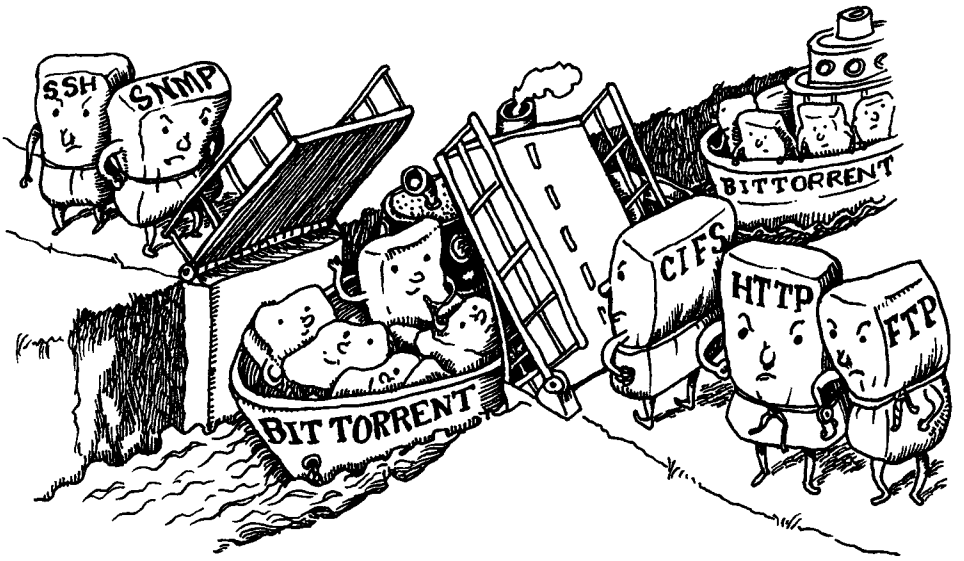
14.6. ★★ Как учесть рассмотренные в этой главе проблемы безопасности в только что инсталлированной Linux-системе? Проверьте, все ли проблемы решены в Linux-системах вашей компьютерной лаборатории. (Необходим доступ с правами пользователя root.)

14.7. ★★ Какие действия необходимо предпринять для подключения нового компьютера к сети в вашей лаборатории? В ответе используйте параметры конкретной среды. Предполагается, что на новом компьютере уже установлена операционная система Linux.

14.8. ★★ Приведите текст конфигурационного файла DHCP-сервера, который выделяет адреса в диапазоне 128.138.192.[1-55]. Задайте срок аренды равным двум часам и сделайте так, чтобы узел с MAC-адресом 00:10:5A:C7:4B:89 всегда получал IP-адрес 128.138.192.55.

# Глава 15

# Маршрутизация




Управление потоками данных — непростая задача. В главе 14 вкратце рассказывалось о перенаправлении IP-пакетов. В настоящей главе мы подробнее изучим этот процесс и познакомимся с различными сетевыми протоколами, благодаря которым маршрутизаторы автоматически находят наиболее эффективные маршруты. Эти протоколы принимают на себя основную нагрузку по управлению таблицами маршрутизации, существенно упрощая задачу администраторам. Они также позволяют быстро перенаправлять трафик в случае выхода из строя маршрутизатора или сетевого сегмента.

Важно отличать реальное перенаправление IP-пакетов от управления таблицей маршрутизации, хотя в обоих случаях употребляют один и тот же термин — "маршрутизация". Перенаправление пакетов — простая процедура, тогда как вычисление маршрутов происходит по довольно запутанному алгоритму. Мы познакомимся только с одноадресной маршрутизацией, так как при многоадресной (когда пакеты направляются группе подписчиков) возникает ряд новых проблем, рассмотреть которые, учитывая объем книги, не представляется возможным.

Для того чтобы составить правильное представление о маршрутизации, в большинстве случаев достаточно информации, изложенной в главе 14 "Сети TCP/IP". Если сетевая инфраструктура уже налажена, можно просто задать единственный статический маршрут (как описано в разделе 14.5 "Маршрутизация") и все готово — у вас есть информация обо всем Интернете. Если же вы стремитесь справиться с сетью, имеющей сложную топологию, или используете операционные системы UNIX или Linux как часть своей сетевой инфраструктуры, тогда эта глава о протоколах и инструментах динамической маршрутизации окажется полезной для вас.

В основе маршрутизации лежит принцип нахождения "следующего перехода". В любой конкретной точке требуется определить только *следующий* узел или маршрутизатор,

куда будет отправлен пакет на пути к пункту назначения. Это в корне отличается от многих старых протоколов, где перед отправкой пакета в сеть требовалось задать точный его маршрут (принцип маршрутизации "от источника")[1].

# 15.1. Подробнее о маршрутизации пакетов

Прежде чем приступать к изучению алгоритмов маршрутизации, рассмотрим подробнее, как используются таблицы маршрутизации. Предположим, сеть имеет топологию, изображенную на рис. 15.1.

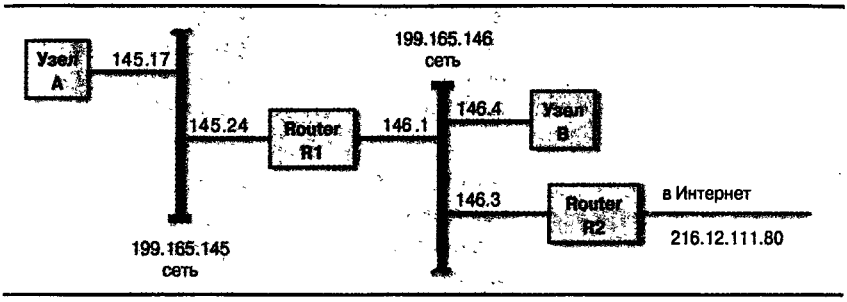



*Рис. 15.1. Демонстрационная сеть*

Маршрутизатор R1 соединяет две сети, а маршрутизатор R2 — одну из этих сетей с внешним миром. Пока будем считать, что R1 и R2 — универсальные компьютеры, а не специализированные маршрутизаторы. (Во всех приводимых примерах предполагается использование операционной системы Linux и протокола IPv4, но в системе UNIX и в протоколе IPv6 используются похожие команды и принципы.) Посмотрим, как выглядят таблицы маршрутизации и некоторые конкретные сценарии перенаправления пакетов. Таблица узла A выглядит следующим образом.

```
A% netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window irtt Iface
199.165.145.0   0.0.0.0         255.255.255.0   U       0 0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U       0 0        0 lo
0.0.0.0         199.165.145.24  0.0.0.0         UG      0 0        0 eth0
```

🕮 Более подробная информация о команде **ifconfig** приведена в разделе 14.9.

Узел A имеет самую простую конфигурацию среди всех четырех компьютеров. Первые два маршрута описывают собственные сетевые интерфейсы узла. Они необходимы, чтобы пакеты, направляемые непосредственно в подключенные сети, не маршрутизировались особым образом. Устройство `eth0` — это Ethernet-интерфейс узла A, а `lo` — интерфейс обратной связи (виртуальный сетевой интерфейс, эмулируемый ядром). Обычно такие записи автоматически добавляются командой **`ifconfig`** при конфигурировании сетевого интерфейса.

Некоторые системы интерпретируют "маршрут обратной связи" как узловой маршрут к конкретному IP-адресу, а не ко всей сети. Поскольку 127.0.0.1 — это единственный

[1] Теоретически IP-пакеты также можно отправлять, выполняя маршрутизацию от источника, но это почти никогда не делается. Из соображений безопасности, подобный режим чаще всего недоступен.

IP-адрес, существующий в сети обратной связи, по существу, неважно, как именно он определен. Единственные изменения, которые можно заметить в таблице маршрутизации, заключаются в том, что в столбце `Destination` вместо адреса `127.0.0.0` будет стоять адрес `127.0.0.1`, а в столбце `Flags` — буква `H`.

🕮 Сетевые маски обсуждаются в разделе 14.4.

Между узловым и сетевым маршрутами нет принципиальной разницы. Когда ядро операционной системы просматривает таблицу маршрутизации, они интерпретируются совершенно одинаково; единственное различие состоит в том, что они имеют разную длину неявной маски.

Стандартный маршрут узла A задает перенаправление всех пакетов, не адресованных интерфейсу обратной связи или сети 195.165.145, на маршрутизатор R1, адрес которого в данной сети — 199.165.145.24. Флаг `G` указывает на то, что данный маршрут ведет к шлюзу, а не к одному из локальных интерфейсов узла A. Шлюзы должны находиться на расстоянии одного перехода.

🕮 Различным типам адресации посвящен раздел 14.3.

Предположим теперь, что узел A посылает пакет узлу B с адресом 199.165.146.4. IP-подсистема ищет маршрут к сети 199.165.146 и, не найдя такового, отправляет пакет по стандартному маршруту, т.е. маршрутизатору R1. На рис. 15.2 приведена структура пакета, проходящего по сети Ethernet (в заголовке Ethernet указаны MAC-адреса соответствующих интерфейсов компьютеров A и R1 в сети 145).

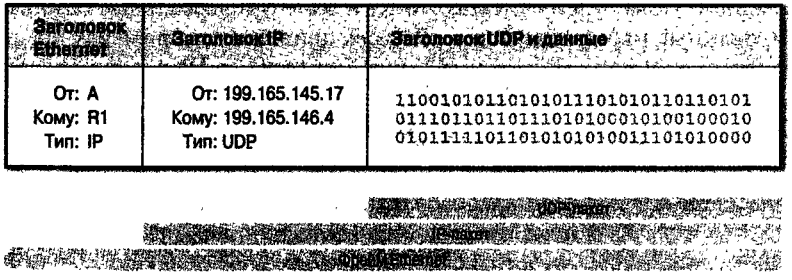



*Рис. 15.2. Ethernet-пакет*

Целевой аппаратный Ethernet-адрес соответствует маршрутизатору R1, но IP-пакет, скрытый в Ethernet-фрейме, не содержит никаких упоминаний о маршрутизаторе. Когда маршрутизатор просматривает поступивший пакет, он обнаруживает, что не является его получателем. Тогда он обращается к собственной таблице маршрутизации, чтобы узнать, как переслать пакет узлу B, не переписывая его IP-заголовок (необходимо, чтобы отправителем пакета оставался узел A).

Таблица маршрутизации узла R1 выглядит следующим образом.

```
R1% netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS  Window irtt  Iface
127.0.0.0       0.0.0.0         255.0.0.0       U       0  0          0  lo
199.165.145.0   0.0.0.0         255.255.255.0   U       0  0          0  eth0
199.165.146.0   0.0.0.0         255.255.255.0   U       0  0          0  eth1
0.0.0.0         199.165.146.3   0.0.0.0         UG      0  0          0  eth1
```

Она почти аналогична таблице узла A, за исключением того, что здесь присутствует два физических сетевых интерфейса. Стандартный маршрут в данном случае ведет к узлу R2, поскольку через него осуществляется выход в Интернет. Пакеты, адресованные любой из сетей 199.165, доставляются напрямую.

Подобно узлу A, узел B имеет один реальный сетевой интерфейс. Но для корректного функционирования ему необходим дополнительный маршрут, поскольку узел имеет прямое соединение сразу с двумя маршрутизаторами. Трафик сети 199.165.145 должен проходить через узел R1, а весь остальной трафик — направляться в Интернет через узел R2.

```
B% netstat -rn
Kernel IP routing table
Destination   Gateway        Genmask         Flags MSS Window irtt Iface
127.0.0.0     0.0.0.0        255.0.0.0       U       0 0          0 lo
199.165.145.0 199.165.146.1  255.255.255.0   U       0 0          0 eth0
199.165.146.0 0.0.0.0        255.255.255.0   U       0 0          0 eth0
0.0.0.0       199.165.146.3  0.0.0.0         UG      0 0          0 eth0
```

🕮 Описание протокола ICMP приведено в разделе 14.5.

Как и узел A, узел B можно сконфигурировать так, чтобы изначально он "знал" только об одном шлюзе, полагаясь на помощь директив переадресации протокола ICMP для устранения избыточных переходов. Ниже показан пример начальной конфигурации.

```
B% netstat -rn
Kernel IP routing table
Destination   Gateway        Genmask         Flags MSS Window irtt Iface
127.0.0.0     0.0.0.0        255.0.0.0       U       0 0          0 lo
199.165.146.0 0.0.0.0        255.255.255.0   U       0 0          0 eth0
0.0.0.0       199.165.146.3  0.0.0.0         UG      0 0          0 eth0
```

Теперь, если узел B посылает пакет узлу A (199.165.145.17), прямой маршрут найден не будет, и пакет отправится на узел R2. Поскольку узел R2 является маршрутизатором, он имеет полную информацию о состоянии сети и, следовательно, "знает" о роли узла R1, куда и будет послан пакет. Кроме того, маршрутизатор R2 обнаружит, что узлы R1 и B находятся в одной сети, поэтому он дополнительно направит узлу B ICMP-сообщение, в соответствии с которым узел B добавит в свою таблицу маршрутизации прямой маршрут к узлу A.

```
199.165.145.17 199.165.146.1 255.255.255.255 UGHD 0 0        0 eth0
```

Благодаря этому маршруту весь трафик, адресованный узлу A, начнет идти непосредственно через маршрутизатор R1. Однако это изменение не затрагивает трафик к другим узлам в сети 145. Для них нужно получить отдельные директивы от маршрутизатора R2.

Некоторые администраторы выбирают для своих систем директивы переадресации протокола ICMP в качестве основного "протокола" маршрутизации, думая, что подобный подход обеспечит большую динамичность. К сожалению, системы и маршрутизаторы осуществляют перенаправление по-разному. Некоторые хранят эти директивы постоянно. Другие удаляют их из таблицы маршрутизации через относительно короткое время (5–15 минут). Третьи вообще игнорируют их, что, вероятно, правильно с точки зрения безопасности.

## 15.2. Демоны и протоколы маршрутизации

В простейших сетях, подобно той, которая представлена на рис. 15.1, целесообразно настраивать маршрутизацию вручную. Однако с определенного момента сети становят-

ся слишком сложными для подобного администрирования. Вместо того чтобы явно сообщать каждому компьютеру, как находить другие компьютеры и сети, лучше заставить сами компьютеры искать эту информацию. Данная задача возлагается на протоколы маршрутизации и демоны, которые их реализуют.

Основное преимущество протоколов маршрутизации над системами статической маршрутизации заключается в том, что они позволяют быстро адаптироваться к изменениям в топологии сети. Когда пропадает канал связи, демоны быстро находят альтернативные маршруты, если они существуют, и сообщают о них в сети, связанные с этим каналом.

Демоны маршрутизации собирают информацию из трех источников: конфигурационных файлов, существующих таблиц маршрутизации и "родственных" демонов других систем. Собранные данные объединяются, и вычисляется оптимальный набор маршрутов, после чего новые маршруты записываются в системную таблицу (и при необходимости посылаются другим системам посредством протоколов маршрутизации). Состояние сети время от времени меняется, поэтому демоны должны периодически опрашивать друг друга, чтобы убедиться в актуальности имеющейся у них информации.

Конкретный алгоритм вычисления маршрутов зависит от протоколов. Последние бывают двух типов: дистанционно-векторные и топологические.

## Дистанционно-векторные протоколы

В основе дистанционно-векторных протоколов лежит следующая идея: если маршрутизатор X находится в пяти переходах от сети Y и является моим соседом, то я нахожусь в шести переходах от данной сети. Демон, работающий по такому протоколу, объявляет о том, как далеко, по его расчетам, расположены известные ему сети. Если соседние демоны не "знают" более коротких маршрутов к этим сетям, они помечают данный компьютер как оптимальный шлюз. В противном случае они просто игнорируют этот анонс. Предполагается, что со временем таблицы маршрутизации придут в стабильное состояние.

Это довольно красивая идея, и если бы все работало так, как задумано, маршрутизация существенно упростилась бы. К сожалению, описанный алгоритм не лучшим образом справляется с изменениями топологии.[2] Иногда стабилизация таблиц вообще не наступает вследствие возникновения бесконечных циклов (например, маршрутизатор X получает информацию от маршрутизатора Y и посылает ее маршрутизатору Z, который возвращает ее маршрутизатору Y). На практике приходится вводить сложные эвристические правила или задавать ограничения. К примеру, в протоколе RIP (Routing Information Protocol — протокол маршрутной информации) считается, что любая сеть, находящаяся на расстоянии более пятнадцати переходов, недоступна.

Даже в обычных ситуациях может потребоваться слишком много циклов обновлений, прежде чем все маршрутизаторы перейдут в стабильное состояние. Следовательно, чтобы не превысить допустимые пределы, необходимо сделать периоды обновлений короткими, а это, в свою очередь, ведет к тому, что дистанционно-векторные протоколы оказываются слишком "словоохотливыми". Например, протокол RIP требует, чтобы маршрутизаторы осуществляли широковещательную рассылку всей имеющейся у них информации каждые 30 секунд. В протоколе EIGRP обновления анонсируются каждые 90 секунд.

[2] Проблема заключается в том, что изменения топологии могут приводить к удлинению существующих маршрутов. Некоторые дистанционно-векторные протоколы, например EIGRP, хранят информацию о всех возможных маршрутах, чтобы на крайний случай был "план отступления". Впрочем, конкретные детали не так уж важны.

В противоположность этому в протоколе BGP (Border Gateway Protocol — протокол пограничного шлюза) вся таблица посылается один раз, после чего изменения распространяются по мере возникновения. Такая оптимизация позволяет существенно снизить "переговорный" трафик (большей частью, ненужный).

В табл. 15.1 перечислены дистанционно-векторные протоколы, широко используемые в настоящее время.

**Таблица 15.1. Распространенные дистанционно-векторные протоколы маршрутизации**

| Аббревиатура | Полное название | Область применения |
|---|---|---|
| RIP | Routing Information Protocol (протокол маршрутной информации) | Локальные сети |
| RIPng | Routing Information Protocol (протокол маршрутной информации), следующее поколение | Локальные сети с протоколом IPv6 |
| EIGRP [a] | Enhanced Interior Gateway Routing Protocol (улучшенный протокол маршрутизации внутреннего шлюза) | Глобальные сети, корпоративные локальные сети |
| BGP | Border Gateway Protocol (протокол пограничного шлюза) | Магистральные сети Интернета |

[a] Протокол EIGRP является собственностью компании Cisco.

## Топологические протоколы

В топологических протоколах, или *протоколах состояния канала*, информация рассылается в относительно необработанном виде. Записи выглядят примерно так: "Маршрутизатор X является смежным по отношению к маршрутизатору Y, и канал функционирует". Полный набор таких записей образует карту сетевых связей, на основании которой каждый маршрутизатор может сформировать собственную таблицу. Основное преимущество топологических протоколов, по сравнению с дистанционно-векторными, заключается в способности быстро стабилизировать таблицы маршрутизации в случае непредвиденного сбоя. К издержкам относится необходимость хранения полной карты соединений на каждом узле, для чего требуются дополнительные процессорные мощности и память.

Поскольку процесс общения между маршрутизаторами не является частью собственно алгоритма вычисления маршрутов, появляется возможность реализовать топологические протоколы так, чтобы не возникало циклов. Изменения в топологической базе данных распространяются по сети очень быстро, не перегружая ни канал, ни центральный процессор.

Топологические протоколы сложнее дистанционно-векторных, зато они позволяют реализовать такие технологии, как маршрутизация на основании запрашиваемого типа обслуживания (поле TOS IP-пакета) и поддержка нескольких маршрутов к одному адресату.

Распространение получили только два топологических протокола: OSPF и IS-IS. Несмотря на то что протокол IS-IS был реализован в широких масштабах, он не так часто используется, и мы не рекомендуем его для использования в новых сетях. Дополнительные комментарии, касающиеся протокола IS-IS, приведены в этом разделе далее.

## Метрики стоимости

Для того чтобы протокол маршрутизации мог определить, какой путь к заданной сети является кратчайшим, необходимо прежде всего выяснить, что значит "кратчайший". Это путь с наименьшим числом переходов? С наименьшей задержкой? С наименьшими финансовыми затратами?

Для целей маршрутизации качество канала связи определяется числом, называемым *метрикой стоимости*. Путем сложения метрик отдельных отрезков пути вычисляется общая стоимость маршрута. В простейших системах каждому каналу назначается стоимость 1, и в результате метрикой маршрута становится число переходов. Но любой из перечисленных выше критериев может являться метрикой стоимости.

Эксперты в области сетей долго и упорно трудились над тем, чтобы определение такого понятия, как метрика стоимости, было максимально гибким, а некоторые современные протоколы даже позволяют использовать разные метрики для разных видов сетевого трафика. Тем не менее в 99% случаев на все это можно не обращать внимания. Для большинства систем вполне подойдут стандартные метрики.

Бывают ситуации, когда кратчайший физический маршрут к адресату не должен быть выбран по умолчанию из административных соображений. В таких случаях можно искусственно завысить метрики критических каналов. Всю остальную работу предоставьте демонам.

## Внутренние и внешние протоколы

*Автономная система* — это группа сетей, находящихся под административным или политическим контролем одного юридического лица. Такое определение довольно расплывчато. Реальные автономные системы могут представлять собой как глобальные корпоративные сети, так и сети университетов или даже отдельных факультетов. Все зависит от того, как осуществляется маршрутизация. Сейчас наблюдается тенденция к укрупнению автономных систем. Это упрощает администрирование и повышает эффективность маршрутизации.

Маршрутизация внутри автономной системы отличается от маршрутизации между такими системами. Протоколы второго типа (внешние, или протоколы внешних шлюзов) должны управлять множеством маршрутов к различным сетям и учитывать тот факт, что соседние маршрутизаторы находятся под контролем других людей. Внешние протоколы не раскрывают топологию автономной системы, поэтому в определенном смысле их можно рассматривать как второй уровень маршрутизации, на котором соединяются группы сетей, а не отдельные компьютеры или кабели.

На практике небольшим и среднего размера организациям редко требуется внешний протокол, если только они не подключены к нескольким провайдерам одновременно. При наличии нескольких провайдеров традиционное разделение на локальный домен и домен Интернета нарушается, поскольку маршрутизаторам приходится определять, какой маршрут в Интернете лучше всего подходит для конкретного адреса. (Это не означает, что каждый маршрутизатор должен знать всю необходимую информацию. Большинство узлов может направлять свои пакеты внутреннему шлюзу, хранящему необходимые сведения.)

Поскольку внешние протоколы не сильно отличаются от своих внутренних аналогов, в этой главе мы сосредоточим внимание на внутренних протоколах и демонах, которые их поддерживают. Если ваш сайт поддерживает внешний протокол, обратитесь к литературным источниками, ссылки на которые приведены в конце главы.

# 15.3. Основные протоколы маршрутизации

В этом разделе мы познакомимся с основными внутренними протоколами маршрутизации, узнаем их преимущества и недостатки.

## Протоколы RIP и RIPng

RIP (Routing Information Protocol — протокол маршрутной информации) — это старый протокол компании Xerox, адаптированный для IP-сетей. Его IP-версия была описана примерно в 1988 году в документе RFC1058. Существует три версии этого протокола: RIPv1, RIPv2 и RIPng только для протокола IPv6 (ng (next generation) означает "следующее поколение").

Все версии этого протокола представляют собой простые дистанционно-векторные протоколы, метрикой стоимости в которых является количество переходов. Поскольку протокол RIP разрабатывался в те времена, когда отдельные компьютеры были дорогими, а сети маленькими, в версии RIPv1 предполагается, что все узлы, находящиеся на расстоянии пятнадцати и более переходов, недоступны. В более поздних версиях это ограничение не было снято, чтобы стимулировать администраторов сложных сетей переходить на более сложные протоколы маршрутизации.

🕮 Информация о бесклассовой адресации, известной под именем CIDR, приведена в разделе 14.4.

Протокол RIPv2 — это улучшенная версия протокола RIP, в которой вместе с адресом следующего перехода передается сетевая маска. Это упрощает управление сетями, где есть подсети и применяется протокол CIDR, по сравнению с протоколом RIPv1. В нем была также предпринята невнятная попытка усилить безопасность протокола RIP.

Протокол RIPv2 можно выполнять в режиме совместимости. Это позволяет сохранить большинство его новых функциональных возможностей, не отказываясь от получателей, использующих простой протокол RIP. Во многих аспектах протокол RIPv2 идентичен исходному протоколу и отдавать предпочтение следует именно ему.

🕮 Детали протокола IPv6 приведены в разделе 14.2.

Протокол RIPng представляет собой переформулирование протокола RIP в терминах протокола IPv6. Он может использоваться только в рамках протокола IPv6, в то время как протокол RIPv2 — только в рамках протокола IPv4. Если вы хотите использовать как протокол IPv4, так и протокол IPv6 вместе с протоколом RIP, то RIP и RIPng необходимо выполнять как отдельные протоколы.

Несмотря на то что протокол RIP известен своим расточительным использованием широковещательного режима, он весьма эффективен при частых изменениях сети, а также в тех случаях, когда топология удаленных сетей неизвестна. Однако после сбоя канала он может замедлить стабилизацию системы.

Сначала исследователи были уверены, что появление более сложных протоколов маршрутизации, таких как OSPF, сделает протокол RIP устаревшим. Тем не менее протокол RIP продолжает использоваться, потому что он простой, легкий в реализации и не требует сложного конфигурирования. Таким образом, слухи о смерти протокола RIP оказались слишком преувеличенными.

Протокол RIP широко используется на платформах, не использующих операционную систему UNIX. Многие устройства, включая сетевые принтеры и сетевые управляемые SNMP-компоненты, способны принимать RIP-сообщения, узнавая о возможных сетевых шлюзах. Кроме того, почти во всех версиях систем UNIX и Linux в той или иной форме существует клиент протокола RIP. Таким образом, протокол RIP считается "наименьшим общим знаменателем" протоколов маршрутизации. Как правило, он применяется для маршрутизации в пределах локальной сети, тогда как глобальную маршрутизацию осуществляют более мощные протоколы.

Некоторые сайты запускают пассивных демонов протокола RIP (обычно демон `routed` или `ripd` из пакета Quagga), которые ожидают сообщений об изменениях в сети, не осуществляют широковещательную рассылку собственной информации. Реальные вычисления маршрутов выполняются с помощью более производительных протоколов, таких как OSPF (см. следующий раздел). Протокол RIP используется только как механизм распространения.

## Протокол OSPF

OSPF (Shortest Path First Open — открытый протокол первоочередного обнаружения кратчайших маршрутов) является самым популярным топологическим протоколом. Термин "первоочередное обнаружение кратчайших маршрутов" (shortest path first) означает специальный математический алгоритм, по которому вычисляются маршруты; термин "открытый" (open) — синоним слова "непатентованный". Основная версия протокола OSPF (версия 2) определена в документе RFC2328, а расширенная версия протокола OSPF, поддерживающая протокол IPv6 (версия 3), — в документе RFC5340. Первая версия протокола OSPF устарела и сейчас не используется.

OSPF — протокол промышленного уровня, который эффективно функционирует в крупных сетях со сложной топологией. По сравнению с протоколом RIP, он имеет ряд преимуществ, включая возможность управления несколькими маршрутами, ведущими к одному адресату, и возможность разделения сети на сегменты ("области"), которые будут делиться друг с другом только высокоуровневыми данными маршрутизации. Сам протокол очень сложный, поэтому имеет смысл использовать его только в крупных системах, где важна эффективность маршрутизации. Для эффективного использования протокола OSPF необходимо, чтобы ваша схема адресации имела иерархический характер.

В спецификации протокола OSPF не навязывается конкретная метрика стоимости. По умолчанию в реализации этого протокола компанией Cisco в качестве метрики используется пропускная способность сети.

## Протокол EIGRP

EIGRP (Enhanced Interior Gateway Routing Protocol — протокол маршрутизации внутренних шлюзов) — это патентованный протокол маршрутизации, используемый только маршрутизаторами компании Cisco. Протокол IGRP был разработан для устранения некоторых недостатков протокола RIP еще в те времена, когда не было такого надежного стандарта, как протокол OSPF. Протокол IGRP был отклонен в пользу протокола EIGRP, который допускает произвольные сетевые маски CIDR. Протоколы IGRP и EIGPR конфигурируются одинаково, несмотря на различия в их организации.

Протокол EIGRP поддерживает протокол IPv6, но в нем, как и во всех других протоколах маршрутизации, адресные пространства IPv6 и IPv4 конфигурируются отдельно и существуют как параллельные домены маршрутизации.

Протокол EIGRP является дистанционно-векторным, но он спроектирован так, чтобы избежать проблем зацикливания и медленной стабилизации, свойственных другим протоколам данного класса. В этом смысле протокол EIGRP считается образцом. Для большинства применений протоколы EIGRP и OSPF обеспечивают равные функциональные возможности.

## IS-IS: протокол маршрутизации между промежуточными системами

Протокол IS-IS (Intra-domain Intermediate System to Intermediate System Routing Protocol) является ответом на протокол OSPF со стороны организации ISO. Первоначально он предназначался для маршрутизации в рамках сетевых протоколов OSI, но впоследствии был расширен для поддержки IP-маршрутизации.

Оба протокола — IS-IS и OSPF — создавались в начале 90-х годов, когда протоколы организации ISO преднамеренно хранились в тайне. Благодаря усилиям со стороны организации IETF, протокол IS-IS получил видимость законности, но со временем стал все сильнее уступать в популярности протоколу OSPF и сегодня используется редко. В настоящее время из-за множества ненужных функциональных особенностей, заложенных в него организацией ISO, лучше его избегать.

## Протоколы RDP и NDP

Протокол RDP (Router Doscovery Protocol — протокол обнаружения маршрутизаторов) использует ICMP-сообщения, посылаемые по групповому IP-адресу 224.0.0.1, для распространения информации о других маршрутизаторах в сети. К сожалению, не все маршрутизаторы в настоящее время рассылают такие сообщения, и не все компьютеры могут их принимать. Остается надеяться, что когда-нибудь этот протокол станет более популярным.

🕮 Информация о протоколе ARP приведена в разделе 14.6.

Протокол NDP (Neighbor Discovery Protocol — протокол обнаружения соседнего узла), основанный на протоколе IPv6, объединяет функциональные возможности протоколов RDP и ARP (Address Resolution Protocol — протокол разрешения адреса), используемых для отображения адресов IPv4 в адреса аппаратных устройств в локальных сетях. Поскольку этот протокол является основным компонентом протокола IPv6, он используется там, где используется протокол IPv6, и протоколы маршрутизации в рамках протокола IPv6 основаны именно на нем.

## Протокол BGP

Протокол BGP (Border Gateway Protocol — протокол пограничной маршрутизации) является протоколом внешней маршрутизации, т.е. он управляет трафиком между автономными системами, а не между отдельными сетями. Существовало несколько популярных протоколов внешней маршрутизации, но протокол BGP пережил их всех.

В настоящее время BGP является стандартным протоколом, используемым для магистральной маршрутизации в Интернете. В средине 2010 года таблица маршрутизации Интернета содержала около 320 тыс. префиксов. Совершенно очевидно, что это масштаб, при котором магистральная маршрутизация существенно отличается от локальной.

# 15.4. Выбор стратегии маршрутизации

Существует четыре уровня сложности, характеризующих процесс управления маршрутизацией в сети:

- отсутствие маршрутизации как таковой;
- только статическая маршрутизация;

- преимущественно статическая маршрутизация, но клиенты принимают RIP-обновления;
- динамическая маршрутизация.

Общая топология сети существенно влияет на маршрутизацию каждого конкретного сегмента. В различных сетях могут требоваться совершенно разные уровни поддержки маршрутизации. При выборе стратегии следует руководствоваться перечисленными ниже эмпирическими правилами.

- Автономная сеть не нуждается в маршрутизации.
- Если из сети есть только один выход во внешний мир, клиенты сети (нешлюзовые узлы) должны иметь стандартный статический маршрут к этому шлюзу. Никакой другой конфигурации не требуется, кроме как на самом шлюзе.
- Можно задать явный статический маршрут к шлюзу, ведущему в небольшую группу сетей, и стандартный маршрут к шлюзу, ведущему во внешний мир. Однако динамическая маршрутизация предпочтительнее, если до нужной сети можно добраться разными маршрутами.
- Если сети пересекают государственные или административные границы, следует использовать динамическую маршрутизацию, даже если сложность сетей этого не требует.
- Протокол RIP отлично работает и широко используется. Не отказывайтесь от него просто потому, что он имеет репутацию устаревшего протокола.
- Проблема протокола RIP заключается в том, что его невозможно масштабировать до бесконечности. Расширение сети рано или поздно приведет к отказу от него. Из-за этого протокол RIP считается промежуточным протоколом с узкой областью применения. Эта область ограничена с одной стороны сетями, которые являются слишком простыми, чтобы применять в них какой-либо протокол маршрутизации, а с другой стороны — сетями, которые являются слишком сложными для протокола RIP. Если вы планируете расширять свою сеть, то было бы целесообразно игнорировать протокол RIP вообще.
- Даже если протокол RIP не соответствует вашей стратегии глобальной маршрутизации, он остается хорошим способом распределения маршрутов к концевым узлам. Однако его не следует применять без особой надобности: системы в сети, имеющей только один шлюз, никогда не требуют динамического обновления.
- Протоколы EIGRP и OSPF имеют одинаковые функциональные возможности, но протокол EIGRP является собственностью компании Cisco. Компания Cisco делает прекрасные и оптимальные по соотношению "цена-качество" маршрутизаторы; тем не менее стандартизация протокола EIGRP ограничивает ваши возможности будущего расширения.

Маршрутизаторы, подключенные к магистрали Интернета, должны использовать протокол BGP. Обычно большинство маршрутизаторов имеет только один вход и, следовательно, для них достаточно задать простой статический стандартный маршрут.

Для организации среднего размера с относительно стабильной локальной структурой, где есть выход в другую сеть, подойдет сочетание статической и динамической маршрутизаций. Маршрутизаторы локальной сети, которые не являются шлюзами во внешние сети, могут использовать статическую маршрутизацию, направляя все неизвестные пакеты стандартной машине, способной общаться с внешним миром и выполнять динамическую маршрутизацию.

Сеть, управлять которой по такой схеме было бы слишком сложно, должна основываться на динамической маршрутизации. В концевых сетях (leaf nets) по-прежнему можно использовать стандартные статические маршруты, но компьютеры в сетях с более чем одним маршрутизатором должны запускать демон **routed** в пассивном режиме.

# 15.5. Демоны маршрутизации

Мы не рекомендуем использовать системы UNIX и Linux в качестве маршрутизаторов в производственных сетях. Специализированные маршрутизаторы проще, надежнее, безопаснее и более быстродействующие (даже если они скрытно запускают ядро системы Linux). Иначе говоря, очень хорошо иметь возможность организовать новую подсеть, используя всего лишь сетевую карту стоимостью 15 долл. и коммутатор за 40 долл. Это вполне разумный подход к организации разреженных, тестовых и вспомогательных сетей.

Системы, действующие как шлюзы в таких подсетях, не требуют дополнительных инструментов для управления их собственными таблицами маршрутизации. Статические маршруты вполне адекватны как для машин, используемых в качестве шлюзов, так и для машин, представляющих собой узлы самой подсети. Однако если вы хотите, чтобы ваша подсеть была доступной для других сетей в вашей организации, вам необходимо сообщить о ее существовании и идентифицировать маршрутизатор, к которому будут привязаны пакеты, посылаемые данной подсети. Обычно для этого на шлюзе запускается демон маршрутизации.

Системы UNIX и Linux могут участвовать в большинстве протоколов маршрутизации с помощью различных демонов маршрутизации. Важным исключением из этого правила является протокол EIGRP, который, насколько нам известно, не имеет широко доступной реализации в системах UNIX и Linux.

Поскольку демоны маршрутизации редко реализуются в производственных системах, мы не будем подробно описывать их использование и конфигурацию. Тем не менее в следующих разделах мы приведем краткое описание типичных программ и укажем, где искать детали конфигурации.

## Демон route: устаревшая реализация в протоколе RIP

Долгое время демон **routed** был стандартным демоном маршрутизации, и его до сих пор включают в дистрибутивы некоторых систем. Демон **routed** понимает только протокол RIP и при этом плохо: даже поддержка версии RIPv2 не поправила ситуацию. Демон **routed** не понимает протокол RIPng, реализация которого основана на современных демонах, таких как Quagga и **ramd** в системе HP-UX.

Демон **routed** целесообразно использовать в пассивном режиме (**-q**), в котором он принимает сообщения об обновлениях маршрутов, но не осуществляет широковещательную рассылку собственных сообщений. Кроме указания опций в командной строке, демон **routed** не требует конфигурирования. Он представляет собой дешевый и простой способ получения сообщений об обновлениях маршрутов без сложного конфигурирования.

Демон заносит обнаруженные маршруты в таблицу ядра. Все маршруты должны анонсироваться, как минимум, каждые четыре минуты, иначе они будут удалены. Правда, демон **route** помнит, какие маршруты он добавлял, и не удаляет статические маршруты, заданные с помощью команды **route**.

🕮 Команда **route** описана в разделе 14.9.

## Демон gated: первый многопротокольный демон маршрутизации

Демон **gated** — элегантная и легко доступная программа маршрутизации, позволяющая одновременно использовать несколько протоколов маршрутизации. Он предоставляет администратору полный контроль над анонсированными маршрутами, широковещательными адресами, правилами безопасности и метриками стоимости. Демон **gated** разрешает нескольким протоколам использовать одни и те же маршруты, позволяя тем самым создавать шлюзы между автономными областями, в которых приняты разные системы маршрутизации. Наконец, демон **gated** имеет один из самых удобных командных интерфейсов и форматов файлов конфигурации среди всего административного программного обеспечения.

Увы, демон **gated** мертв (или, по крайней мере, при смерти), хотя память о нем еще живет в выпусках систем HP-UX 3.5.9 и AIX 6.0, которые вообще медленно реагируют на происходящие изменения.

Демон **gated** представляет собой поучительный пример риска, с которым связаны попытки разрабатывать открытое программное обеспечение. Он задумывался как свободно распространяемая программа, но в 1992 году был приватизирован и переделан консорциумом по разработке программ; в результате его обновления стали доступными только членам этого консорциума. В конце концов консорциум был расформирован, и права на коммерческую версию демона **gated** несколько раз "меняли" владельцев. Тем временем открытые проекты Zebra и Quagga вытеснили демон **gated** и сами стали играть ведущие роли в области открытых пакетов маршрутизации. В настоящее время демон **gated** угас и как коммерческий продукт, и как открытый проект. Печальный конец полезного и хорошо спроектированного пакета.

## Пакет Quagga: основной демон маршрутизации

Quagga (`quagga.net`) — это опытно-конструкторское подразделение проекта Zebra, выполняемого в рамках проекта GNU. Проект Zebra был запущен Кунихиро Ишигуро (Kunihiro Ishiguro) и Йошинари Йошикава (Yoshinari Yoshikawa) для реализации многопротокольной маршрутизации с помощью коллекции независимых демонов, а не одного монолитного приложения. В реальной жизни квагга (quagga) — это исчезнувший подвид зебры, которая была последний раз сфотографирована в 1870 году. Но его цифровая реинкарнация Quagga выжила, а проект Zebra закрылся.

В настоящее время пакет Quagga реализует все протоколы RIP (все версии), OSPF (версии 2 и 3), BGP и IS-IS. Он выполняется в системах Linux, Solaris и разных вариантах системы BSD. В системе Solaris и версиях системы Linux, имеющихся в нашем распоряжении, пакет Quagga либо инсталлирован по умолчанию, либо доступен в качестве вспомогательного пакета, который можно загрузить из стандартного системного репозитория программного обеспечения.

В пакете Quagga демон **zebra** играет роль информационного центра для маршрутизации. Он управляет взаимодействием между таблицей маршрутизации ядра и демонами, соответствующими отдельным протоколам маршрутизации (**ripd**, **ripngd**, **ospfd**, **ospf6d**, **bgpd** и **isisd**). Кроме того, он управляет потоками информации о маршрутах, которыми обмениваются протоколы. Каждый демон имеет свой собственный конфигурационный файл в каталоге **/etc/quagga**.

Для того чтобы послать запрос или изменить конфигурацию, можно соединиться с любым из демонов Quagga с помощью интерфейса командной строки (**vtysh** в системе

Linux и **quaggaadm** в системе Solaris). Командный язык разработан так, чтобы он был понятен пользователям операционной системы IOS компании Cisco; дополнительные детали можно найти в разделе 15.6, посвященном маршрутизаторам Cisco. Как и в системе IOS, для того чтобы войти в режим "суперпользователя", необходимо выполнить команду **enable**, для того чтобы ввести команды конфигурирования, необходимо выполнить команду **config term**, а для того чтобы сохранить изменения конфигурации в файле конфигурации демона, необходимо выполнить команду **write**.

Официальная документация доступна на сайте `quagga.net` в виде файлов в форматах HTML и PDF. Несмотря на свою полноту, эта документация содержит, в основном, удобный каталог опций, а не общий обзор системы. Более практичную документацию можно найти на сайте `wiki.quagga.net`. Здесь находятся подробно прокомментированные примеры файлов конфигурации, ответы на часто задаваемые вопросы и советы.

Несмотря на то что файлы конфигурации имеют простой формат, необходимо знать протокол, конфигурацию которого вы настраиваете, а также понимать, что означают те или иные опции. Список рекомендованной литературы по этому вопросу приведен в конце главы.

Системы Solaris и Red Hat содержат в каталоге **/ets/quagga/** коллекцию полезных примеров файла конфигурации для разных демонов Quagga.

## Демон ramd: многопротокольная система маршрутизации для HP-UX

Система HP-UX содержит набор демонов маршрутизации, которые по своей архитектуре напоминают демонов Zebra и Quagga. Мы не знаем, можно ли объяснить эту схожесть подражанием, конвергентной эволюцией или, возможно, ранним ответвлением от основного кода проекта Zebra.

В любом случае эта схожесть носит лишь поверхностный характер. Важным различием является то, что система HP поддерживает только протокол IPv6 и протоколы внешней маршрутизации (RIPng, BGP и IS-IS), но вообще не поддерживает протокол OSPF. Кроме того, язык конфигурации отличается от базового, а управляющая утилита (**rdc**, в отличие от утилит **vtysh** или **quaggaadm** пакета Quagga) принимает аргументы только из командной строки; она не имеет независимой интерфейсной оболочки.

Эта подсистема системы HP называется Route Administration Manager Daemon, а ее демон **ramd** играет ту же роль, какую в пакете Quagga играет демон **zebra**. Как и в пакете Quagga, демоны, соответствующие разным протоколам, называются **ripngd**, **isisd** и **bgpd**.

## Маршрутизатор XORP

Проект XORP (eXtensible Open Router Platform — расширяемая платформа маршрутизации с открытым кодом) был открыт примерно в то же время, что и проект Zebra, но его цели были более универсальными. Вместо маршрутизации, проект XORP был нацелен на эмулирование всех функций заданного маршрутизатора, включая фильтрацию пакетов и управление трафиком. Информация об этом проекте хранится на сайте `xorp.org`.

Интересный аспект платформы XORP заключается в том, что она не только функционирует в разных операционных системах (Linux, разных версиях BSD, Mac OS X и Windows Server 2003), но и может запускаться непосредственно с "живого" компакт-диска (т.е. непосредственно с компакт-диска, без инсталляции на жесткий диск. — *Примеч.*

*ред.*) на персональном компьютере. Этот "живой" компакт-диск (live CD) скрытно основан на системе Linux, но он прошел долгую эволюцию и позволяет превращать типичный персональный компьютер в специализированное устройство для маршрутизации.

## Специфика поставщиков

Пакет Quagga — это лучшее программное обеспечение для маршрутизации в системе Linux. Все доступные нам дистрибутивы либо инсталлировали его по умолчанию, либо позволяли загружать его из репозитория. Пакет Quagga настолько распространен, что большинство дистрибутивов больше не содержит демон **routed**. Даже в тех дистрибутивах, в которых он есть, содержится его устаревшая версия, не поддерживающая протокол RIPv2.

Система Solaris содержит функционал **routed** (который на самом деле называется **in.routed**), понимающий протокол RIPv2. Она также содержит пакет Quagga, поэтому вы можете выбирать. Для любой маршрутизации в рамках протокола IPv6 необходимо использовать пакет Quagga.

Функционал **in.routed** представляет собой механизм маршрутизации, заданный по умолчанию. Он начинает работать автоматически в момент загрузки системы, если в каталоге **/etc/defaultrouter** пользователь не задал свой собственный шлюз сети. Система Solaris продолжает поддерживать демон поиска маршрутов (router discovery daemon) **in.rdisc**, что довольно странно, поскольку его функциональными возможностями теперь обладает демон **in.routed**.

Базовая подсистема маршрутизации **ramd** в системе HP-UX рассматривалась выше. Кроме того, система HP содержит также копию демона **gated** 3.5.9, которая является довольно старой и не поддерживает протокол IPv6. Если вы хотите управлять маршрутизацией одновременно в рамках протоколов IPv4 и IPv6 в системе HP-UX, то для протокола IPv4 следует использовать демон **gated**, а для протокола IPv6 — демон **ramd**. К сожалению, пакет Quagga в настоящее время в системе HP-UX не работает.

В системе AIX есть три демона маршрутизации: **gated** v6.0, **routed**, понимающий только протокол RIPv1, и **ndpd-router**, представляющий собой реализацию протоколов RIPng и NDP. Кроме того, демон **gated** в системе AIX понимает протокол RIPng. Однако если вы хотите использовать демон **gated** для маршрутизации в рамках протокола IPv6, необходимо запустить оба демона: **gated** и **ndpd-router**. Детали описаны в документации.

# 15.6. Маршрутизаторы Cisco

Маршрутизаторы, выпускаемые компанией Cisco Systems, Inc., сегодня являются стандартом де-факто на рынке интернет-маршрутизаторов. Захватив более 60% рынка, компания Cisco добилась широкой известности своих продуктов, к тому же есть много специалистов, умеющих работать с этими устройствами. Раньше в качестве маршрутизаторов часто приходилось использовать UNIX-системы с несколькими сетевыми интерфейсами. Сегодня специализированные маршрутизаторы размещены в коммутационных шкафах и над подвесными потолками, где сходятся сетевые кабели.

Большинство маршрутизаторов Cisco работает под управлением операционной системы Cisco IOS, которая является собственностью компании и не имеет отношения к си-

стеме UNIX. У нее достаточно большой набор команд: полная бумажная документация занимает на полке почти полтора метра. Мы никогда не смогли бы полностью описать здесь эту операционную систему, но все же постараемся дать о ней основные сведения.

По умолчанию в системе IOS определены два уровня доступа (пользовательский и привилегированный), каждый из которых включает механизм парольной защиты. По умолчанию к маршрутизатору Cisco можно получить доступ на пользовательском уровне, воспользовавшись утилитой **telnet**. Вы получите приглашение на ввод пароля.

```
$telnet acme-gw.acme.com[3]
Connected to acme-gw.acme.com.
Escape character is '^]'.

User Access Verification
Password:
```

После ввода правильного пароля появится приглашение интерпретатора EXEC.

```
acme-gw.acme.com>
```

С этого момента можно начинать вводить команды, например **show interfaces** для отображения списка сетевых интерфейсов маршрутизатора или **show ?** для получения справки по возможным параметрам.

Для того чтобы перейти в привилегированный режим, введите команду **enable**, а при последующем запросе — привилегированный пароль. Признаком привилегированного режима является наличие символа '#' в конце строки приглашения.

```
acme-gw.acme.com#
```

Будьте осторожны! В данном режиме можно делать все что угодно, включая удаление конфигурационной информации и даже самой операционной системы. Если сомневаетесь, обратитесь к справочным руководствам по системе Cisco или одной из исчерпывающих книг, выпускаемых издательством Cisco Press.

Команда **show running** позволяет узнать текущие динамические параметры маршрутизатора, а по команде **show config** отображаются текущие неизменяемые параметры. В большинстве случаев оба набора данных идентичны. Вот типичная конфигурация.

```
acme-gw.acme.com# show running
Current configuration:
version 12.1
hostname acme-gw
enable secret xxxxxxxx
ip subnet-zero

interface Ethernet0
 description Acme internal network
  ip address 192.108.21.254 255.255.255.0
  no ip directed-broadcast
interface Ethernet1
 description Acme backbone network
  ip address 192.225.33.254 255.255.255.0
  no ip directed-broadcast

ip classless
line con 0
```

[3] Современные версии системы IOS поддерживают разные методы доступа, включая SSH. Разумеется, утилита **telnet** очень опасна. Если ваша организация уже использует маршрутизаторы Cisco, свяжитесь с системным администратором и выясните, какие методы можно использовать.

```
transport input none

line aux 0
 transport input telnet
line vty 0 4
 password xxxxxxxx
 login

end
```

Модифицировать конфигурацию маршрутизатора можно разными способами. Компания Cisco предлагает графические утилиты, работающие в некоторых версиях UNIX/Linux и Windows, но опытные сетевые администраторы никогда ими не пользуются. Их самый мощный "инструмент" — командная строка. Кроме того, с помощью команды **scp** можно загрузить с маршрутизатора его конфигурационный файл и отредактировать в любимом текстовом редакторе, после чего снова записать файл в память маршрутизатора.

Для того чтобы отредактировать конфигурационный файл в режиме командной строки, введите команду **config term**.

```
acme-gw.acme.com# config term
Enter configuration commands, one per line. End with CNTL/Z.
acme-gw(config)#
```

Теперь можно вводить конфигурационные команды в том виде, в котором их будет отображать команда **show running**. Например, если требуется изменить IP-адрес интерфейса Ethernet0, введите следующее.

```
interface Ethernet0
ip address 192.225.40.253 255.255.255.0
```

По окончании ввода конфигурационных команд нажмите <Ctrl+Z>, чтобы вернуться к обычной командной строке. Если все прошло успешно, сохраните новую конфигурацию в постоянной памяти посредством команды **write mem**.

Приведем несколько советов, касающихся эффективной работы с маршрутизаторами Cisco.

- Присвойте маршрутизатору имя с помощью команды **hostname**. Это позволит избежать несчастных случаев, связанных с изменением конфигурации не того маршрутизатора. Заданное имя всегда будет отображаться в командной строке.
- Всегда храните резервную копию конфигурационного файла маршрутизатора. С помощью команд **scp** или **tftp** можно каждую ночь пересылать текущую конфигурацию в другую систему на хранение.
- Зачастую существует возможность хранить копию конфигурации в энергонезависимой памяти NVRAM или на переносном флеш-накопителе. Не пренебрегайте этим!
- Сконфигурировав маршрутизатор для SSH-доступа, отключите протокол Telnet совсем.
- Контролируйте доступ к командной строке маршрутизатора, создавая списки доступа для каждого порта VTY (аналогичен порту PTY в UNIX-системе). Это не позволит посторонним пользователям "взломать" маршрутизатор.
- Управляйте трафиком между сетями (по возможности, и во внешний мир тоже), создавая списки доступа для каждого интерфейса. Более подробная информация приведена в разделе 22.11.

- Физически ограничивайте доступ к маршрутизаторам. Ведь если у злоумышленника будет физический доступ к устройству, он легко сможет изменить пароль привилегированного режима.

Если у вас несколько маршрутизаторов и несколько сотрудников, отвечающих за их работу, воспользуйтесь утилитой RANCID, которую можно загрузить с сайта `shruberry.net`. Название RANCID (игра слов: *rancid — негодный. — Примеч. ред.*) говорит само за себя, но у этой программы есть одно преимущество: она регистрируется на ваших маршрутизаторах каждую ночь и получает их файлы конфигурации. Затем она сравнивает эти файлы и сообщает вам об их изменениях. Кроме того, она автоматически передает файлы конфигурации системе управления версиями (см. раздел 12.9).

## 15.7. Рекомендуемая литература

- Perlman Radia. *Interconnections: Bridges, Routers, Switches, and Interworking Protocols (2nd Edition).* Reading, MA: Addison-Wiley, 2000.

  Это выдающаяся книга в данной области. Если вы решили купить только одну книгу по основе работы в сетях, покупайте ее. Кроме того, не упускайте шанса "зависнуть" с Радией — она очень веселая и обладает удивительно большими знаниями.
- Dooley Kevin and Ian J. Brown. *Cisco IOS Cookbook (2nd Edition).* Sebastopol, CA: O'Reilly Media, 2007.
- Doyle Jeff and Jennifer Carroll. *Routing TC/IP, Volume I (2nd Edition).* Indianapolis, IN: Cisco Press, 2005.
- Doyle Jeff and Jennifer Carroll. *Routing TC/IP, Volume II.* Indianapolis, IN: Cisco Press, 2001.

  Это двухтомное издание представляет собой глубоко продуманное введение в протоколы маршрутизации и не ориентировано на какую-то конкретную реализацию. Том I посвящен внутренним протоколам, а том II — внешним протокола, системе NAT и многоадресной маршрутизации.
- Halabi Sam and Danny McPherson. *Internet Routing Architectures, Second Edition.* Cisco Press. 2000.
- В этой книге основной акцент делается на BGP.
- Huitema Christian. *Routing in the Internet, Second Edition.* Prentice Hall, 2000.

  Эта книга представляет собой отличное введение в маршрутизацию для начинающих. В ней описано большинство используемых сегодня протоколов маршрутизации, а также рассматривается ряд сложных тем, например групповое вещание.

Есть много документов RFC, посвященных маршрутизации. Основные из них перечислены в табл. 15.2.

**Таблица 15.2. Документы RFC, посвященные маршрутизации**

| RFC | Название | Авторы |
|---|---|---|
| 1075 | Distance Vector Multicast Routing Protocol | Waitzman et al. |
| 1256 | ICMP Router Discovery Messages | Deering |
| 1724 | RIP Version 2 MIB Extension | Malkin, Baker |
| 2080 | RIng for IPv6 | Malkin, Minnear |

*Окончание табл. 15.2*

| RFC | Название | Авторы |
|---|---|---|
| 2328 | OSPF Version 2 | Moy |
| 2453 | RIP Version 2 | Malkin |
| 4271 | A Border Gateway Protocol 4 (BGP-4) | Rekhter, Li, et al. |
| 4552 | Authentication/Confidentiality for OSPFv3 | Gupta, Melam |
| 4822 | RIPv2 Cryprographic Authentication | Arkinson, Fanto |
| 4861 | Neighbor Discovery for IPv6 | Narten et al. |
| 5175 | IPv6 Router Advertisement Flags Option | Haberman, Hinden |
| 5308 | Routing IPv6 with IS-IS | Hopps |
| 5340 | OSPF for IPv6 | Coltun et al. |
| 5643 | Management Information Base for OSFv3 | Joyal, NManral, et al. |

# 15.8. Упражнения

15.1. Изучите возможности команды **route** в Linux и составьте краткое описание ее работы. Как с помощью этой команды сделать следующее:

а) добавить стандартный маршрут к шлюзу 128.138.129.1 через интерфейс `eth1`?

б) удалить маршрут 128.138.129.1?

в) определить, каким образом был добавлен маршрут: с помощью демона, например **gated**, или директивы переадресации протокола ICMP (ту же информацию можно получить, анализируя вывод команды **netstat -rn**)?

15.2. Сравните статический и динамический способы маршрутизации, укажите преимущества и недостатки каждого способа. Опишите ситуации, для которых подходит тот или иной вид маршрутизации, и объясните почему.

15.3. ★ Команда **netstat -rn** сообщает показанные ниже результаты. Опишите каждый маршрут и нарисуйте общую схему сети. Какая сеть — 10.0.0.0 или 10.1.1.0 — "ближе" к Интернету? Как появился каждый из маршрутов?

```
Destination     Gateway      Genmask          Flags   MSS Window  irtt Iface
10.0.0.0        0.0.0.0      255.255.255.0    U       40 0        0 eth1
10.1.1.0        0.0.0.0      255.255.255.0    U       40 0        0 eth0
0.0.0.0         10.0.0.1     0.0.0.0          UG      40 0        0 eth1
```

15.4. ★★ Определите стратегию маршрутизации в вашей системе. Какие протоколы используются? Какие компьютеры подключены к Интернету? Воспользуйтесь утилитой **tcpdump** для просмотра маршрутных обновлений, курсирующих в локальной сети, и командой **traceroute** — для выхода за пределы локальной сети. (Необходим корневой доступ.)

15.5. ★★ Если бы вы были интернет-провайдером среднего размера, предоставляющим услуги коммутируемого доступа и виртуального хостинга, какую конфигурацию системы маршрутизации вы бы использовали? Убедитесь в том, что учитываются не только шлюзы между интернет-магистралью и вашей сетью, но и любые внутренние маршрутизаторы. Нарисуйте схему маршрутизации в сети.

# Глава 16

# Сетевые аппаратные средства

Пользуетесь ли вы поисковой машиной Google с помощью своего мобильного телефона[1], выполняете ли интерактивные банковские операции или получаете видеозвонки с помощью программы Skype от своих кузин из Бельгии, практически все в мире в настоящее время обрабатывается в цифровой форме. Перемещение данных из одного места в другое на уме почти у всех. В основе всего этого сумасшествия лежит фантастическое сетевое оборудование и, как вы уже догадались, множество разнообразных программ, созданных в недрах системы UNIX. Крупномасштабная передача пакетов данных на основе технологии UNIX вошла в жизнь очень многих людей.

Многие сетевые технологии продвигались в течение долгих лет, но в результате появился очевидный победитель: Ethernet. Сегодня технологию Ethernet можно встретить всюду: от игровых приставок до холодильников. Глубокое понимание принципов работы этой системы чрезвычайно важно для успешной работы системного администратора.

Совершенно очевидно, что быстродействие и надежность сетей непосредственно влияют на результаты деятельности компаний. Однако в настоящее время сетевые технологии настолько вездесущи, что состояние сети может повлиять на возможность взаимодействия между людьми, например возможность делать телефонные звонки. Плохая организация сети — это личная и профессиональная неудача, которая может иметь катастрофические социальные последствия. Кроме того, устранение этих недостатков порой обходится очень дорого.

Успешное создание сети зависит от, по крайней мере, четырех важнейших факторов, а именно:

- разработки разумной структуры сети;
- выбора высококачественного оборудования;

[1] Знаете ли вы, что на телефонах iPhone установлена встроенная система UNIX?

- правильной инсталляции и документирования;
- компетентной эксплуатации и сопровождения.

В этой главе рассматриваются принципы, инсталляция и функционирование сетей Ethernet. Мы также кратко опишем такие устаревшие технологии, как DSL (Digital Subscriber Line), которые обычно предстают перед конечными пользователями в облике — сюрприз! — технологии Ethernet.

# 16.1. Технология Ethernet: сетевая панацея

Захватив более 95% мирового рынка локальных сетей (LAN — Local Area Network), технология Ethernet в самых разных формах проявляется почти всюду. Разработку стандарта Ethernet начал Боб Меткалф (Bob Metcalfe) из Массачусетсского технологического института в рамках своей кандидатской диссертации, но в настоящее время она описана во многих стандартах IEEE.

В первоначальной спецификации Ethernet была определена скорость передачи данных 3 Мбит/с (мега*бит* в секунду), но почти сразу же она выросла до 10 Мбит/с. Как только в 1994 году была закончена работа над стандартом, предусматривавшим скорость 100 Мбит/с, стало ясно, что технология Ethernet будет лишь эволюционировать, а не вытесняться новой технологией. Это вызвало гонку технологий, в ходе которой производители старались создать все более быстродействующую версию Ethernet, и это соревнование еще не закончено. Основные этапы эволюции различных стандартов Ethernet приведены в табл. 16.1[2].

**Таблица 16.1. Эволюция Ethernet**

| Год | Скорость | Название стандарта | Номер IEEE | Расстояние | Средство передачи |
|---|---|---|---|---|---|
| 1973 | 3 Мбит/с | Xerox Ethernet | – | ? | Коаксиальный кабель |
| 1980 | 10 Мбит/с | Ethernet 1 | – | 500 м | Коаксиальный кабель RG-11 |
| 1982 | 10 Мбит/с | DIX Ethernet (Ethernet II) | – | 500 м | Коаксиальный кабель RG-11 |
| 1985 | 10 Мбит/с | 10Base5 ("Thicknet") | 802.3 | 500 м | Коаксиальный кабель RG-11 |
| 1985 | 10 Мбит/с | 10Base2 ("Thinnet") | 802.3 | 180 м | Коаксиальный кабель RG-58 |
| 1989 | 10 Мбит/с | 10BaseT | 802.3 | 100 м | Медный кабель НВП[a] категории 3 |
| 1993 | 10 Мбит/с | 10BaseF | 802.3 | 2 км | ММ[б] оптоволокно |
| | | | | 25 км | ОМ оптоволокно |
| 1994 | 100 Мбит/с | 100BaseTX ("Fast Ethernet") | 802.3u | 100 м | Медный кабель НВП категории 5 |
| 1994 | 100 Мбит/с | 100BaseFX | 802.3u | 2 км | ММ оптоволокно |
| | | | | 20 км | ОМ оптоволокно |
| 1998 | 1 Гбит/с | 1000BaseSX | 802.3z | 260 м | ММ оптоволокно (62,5 мкм) |
| | | | | 550 м | ММ оптоволокно (50 мкм) |
| 1998 | 1 Гбит/с | 1000BaseLX | 802.3z | 440 м | ММ оптоволокно (62,5 мкм) |
| | | | | 550 м | ММ оптоволокно (50 мкм) |
| | | | | 3 км | ОМ оптоволокно |

[2] Мы не упомянули несколько стандартов, которым не удалось завоевать популярность, в частности 100BaseT4 и 100BaseVG-AnyLAN.

*Окончание табл. 16.1*

| Год | Скорость | Название стандарта | Номер IEEE | Расстояние | Средство передачи |
|---|---|---|---|---|---|
| 1998 | 1 Гбит/с | 1000BaseCX | 802.3z | 25 м | Двухпроводный экранированный кабель |
| 1999 | 1 Гбит/с | 1000BaseT ("Gigabit Ethernet") | 802.3ab | 100 м | Медный кабель НВП категорий 5E и 6 |
| 2002 | 10 Гбит/с | 10Gbase-SR | 802.3ae | 300 м | MM оптоволокно |
| | | 10Gbase-LR | | 10 км | OM оптоволокно |
| | | 10Gbase-ER | 802.3aq | 40 км | OM оптоволокно |
| | | 10Gbase-ZR | | 80 км | OM оптоволокно |
| 2006 | 10 Гбит/с | 10Gbase-T ("10 Gig") | 802.3an | 100 м | ВП категории 6a, 7, НВП категории 7a |
| 2009 | 40 Гбит/с | 40Gbase-CR4 | P802.3ba | 10 м | Медный кабель НВП |
| | | 40Gbase-SR4 | | 100 м | MM оптоволокно |
| 2012[в] | 1 Тбит/с | TBD | TBD | TBD | CWDM оптоволокно |
| 2015[в] | 10 Тбит/с | TBD | TBD | TBD | DWDM оптоволокно |

[а] НВП — неэкранированная витая пара, ВП — витая пара.

[б] ММ — многомодовое, ОМ — одномодовое.

[в] Промышленный проект.

## Как работает Ethernet

Технологию Ethernet можно представить в виде великосветского раута, на котором гости (компьютеры) не перебивают друг друга, а ждут паузы в разговоре (отсутствия трафика в сетевом кабеле), чтобы заговорить. Если два гостя начинают говорить одновременно (т.е. возникает конфликт), оба они останавливаются, извиняются друг перед другом, ждут немного, а затем один из них начинает говорить снова.

В технической терминологии такая схема называется CSMA/CD (Carrier Sense Multiple Access with Collision Detection — множественный доступ с контролем несущей и обнаружением конфликтов). Смысл этого названия заключается в следующем:

- контроль несущей — можно определить, занят ли канал;
- множественный доступ — кто угодно может передавать сообщения;
- обнаружение конфликтов — передающая система "знает", когда она "перебивает" кого-нибудь.

Фактическая задержка при обнаружении конфликтов является случайной. Это позволяет избежать такого развития событий, при котором два компьютера одновременно передают сообщения в сеть, обнаруживают коллизию, ждут некоторое время, а затем синхронно возобновляют передачу, переполняя, таким образом, сеть конфликтами.

В настоящее время важность соглашений CSMA/CD осознали даже приверженцы коммутаторов, которые обычно ограничивают количество узлов в домене, в котором происходят коллизии, до двух. (Если продолжить аналогию с великосветским раутом, можно описать этот вариант как ситуацию, в которой два собеседника, как в старом кино, чопорно сидят на противоположных концах длинного обеденного стола.)

## Топология Ethernet

С точки зрения топологии, сеть Ethernet представляет собой разветвляющуюся шину, но без петель. У пакета есть только один путь следования между любыми двумя узлами, расположенными в одной сети. В сети Ethernet могут передаваться пакеты трех типов: однонаправленные, групповые и широковещательные. Пакеты первого типа адресованы одному узлу, второго — группе узлов, третьего — всем узлам сегмента.

Широковещательный домен — это совокупность узлов, которые принимают пакеты, направляемые по аппаратному широковещательному адресу. В каждом логическом сегменте сети Ethernet существует только один широковещательный домен. В ранних стандартах Ethernet и средствах передачи (например, 10Base5) понятия физического и логического сегментов были тождественными, поскольку все пакеты передавались по одному большому кабелю, в который втыкались сетевые интерфейсы компьютеров[3].

С появлением современных коммутаторов логические сегменты стали включать в себя множество (десятки и даже сотни) физических сегментов, к которым подключено всего два устройства: порт коммутатора и компьютер. Коммутаторы отвечают за доставку групповых и однонаправленных пакетов в физический сегмент, где расположен нужный адресат (адресаты); широковещательные пакеты направляются во все сетевые порты логического сегмента.

Логический сегмент может состоять из физических сегментов, имеющих разную скорость передачи данных (10 Мбит/с, 100 Мбит/с, 1 Гбит/с или 10 Гбит/с). Следовательно, коммутаторы должны иметь средства буферизации и синхронизации для предотвращения возможных конфликтов.

## Неэкранированная витая пара

Неэкранированная витая пара (НВП) — самая популярная среда передачи данных в сетях Ethernet. Она основана на звездообразной топологии и обладает рядом преимуществ по сравнению с другими средами:

- применяется недорогой, недефицитный медный кабель (иногда можно использовать готовую телефонную проводку);
- соединение на основе НВП гораздо проще смонтировать и наладить, чем соединение на основе коаксиального кабеля или оптического волокна, к тому же легко подобрать нужную длину кабеля;
- используются дешевые, надежные и удобные в эксплуатации разъемы RJ-45;
- все соединения функционируют независимо друг от друга, поэтому дефект кабельной системы в одном соединении не повлияет на другие компьютеры сети.

Общая схема сети на основе НВП изображена на рис. 16.1.

Провода, используемые в современных локальных вычислительных сетях на основе неэкранированной витой пары, обычно подразделяют на восемь категорий. Эта система оценки параметров была впервые введена компанией Anixter, крупным поставщиком кабельной продукции, и впоследствии стандартизирована организацией TIA (Telecommunications Industry Association — ассоциация телекоммуникационной промышленности). Сегодня выделяются категории 1–7 и промежуточные категории 5e и 6a.

[3] Мы не шутим! Подключение нового компьютера к сети предполагало прокалывание отверстия в изоляции кабеля с помощью специального соединителя, называемого "зуб вампира", который позволял добраться до центрального проводника. Этот соединитель затем зажимался винтами.

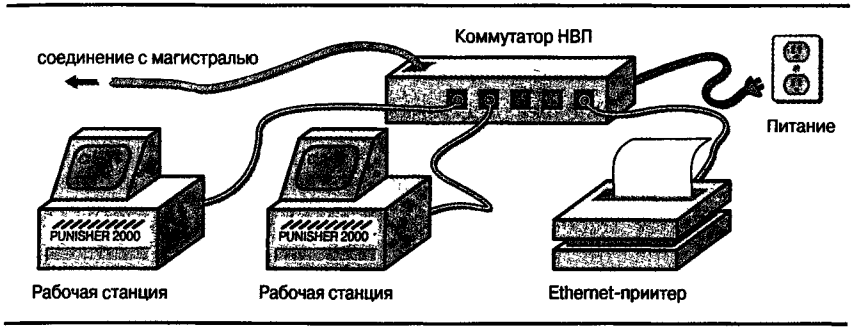



*Рис. 16.1. Схема сети на основе НВП*

Организация ISO (International Organization for Standartization — Международная организация по стандартизации) тоже подключилась к процессу стандартизации кабелей и предложила собственную классификацию, которая почти в точности повторяет классификацию TIA. Например, кабель категории 5 в системе TIA эквивалентен кабелю класса D в системе ISO. Для наглядности в табл. 16.2 подытожены ключевые различия между основными современными стандартами кабелей. Эта таблица поможет вам произвести впечатление на своих друзей во время вечеринки.

**Таблица 16.2. Характеристики кабелей НВП**

| Параметр[a] | Единица измерения | 5 D[б] | 5e | 6 E | 6a EA | 7 | 7a |
|---|---|---|---|---|---|---|---|
| Ширина полосы | МГц | 100 | 100 | 250 | 500 | 600 | 1000 |
| Затухание | дБ | 24 | 24 | 21,7 | 18,4 | 20,8 | 60 |
| NEXT | дБ | 27,1 | 30,1 | 39,9 | 59 | 62,1 | 60,4 |
| ELFEXT | дБ | 17 | 17,4 | 23,2 | 43,1 | 46,0 | 35,1 |
| Затухание отраженного сигнала (обратная потеря) | дБ | 8 | 10 | 12 | 32 | 14,1 | 61,93 |
| Задержка распространения сигнала | нс | 548 | 548 | 548 | 548 | 504 | 534 |

[a] NEXT (Near-end crosstalk) — ослабление перекрестной помехи на ближнем конце. ELFEXT (Equal level far-end crosstalk) — ослабление равноуровневой перекрестной помехи на дальнем конце.

[б] Включая дополнительные спецификации TIA TSB 95 и ISO FDAM 2.

На практике кабели категорий 1 и 2 годятся только для передачи звуковых сигналов. Кабель категории 3 является стандартом для самых медленных локальных сетей: 10BaseT со скоростью передачи 10 Мбит/с. Кабель категории 4 ни для чего конкретного не предназначен.

Кабель категории 5 поддерживает работу на скорости 100 Мбит/с. Кабели категорий 5E, 6 и 6a поддерживают скорость передачи 1 Гбит/с и в настоящее время используются в качестве стандарта. Кабель категории 6a лучше всего подходит для организации новых сетей, поскольку он особенно устойчив к помехам, возникающим из-за использования старых стандартов передачи сигналов (например, 10BaseT). При прокладке кабелей категорий 5 и 5e были зафиксированы определенные проблемы. Кабели категорий 7 и 7a предназначены для передачи данных со скоростью 10 Гбит/с.

Для соединений 10BaseT требуются две пары проводов категории 3, причем длина каждой линии передачи ограничена 100 метрами. В соединениях 100BaseTX предельная

длина та же, но используются две пары проводов категории 5. Соединение 1000BaseTX требует четырех пар проводов категорий 5e или 6/6a. Аналогично соединение 10GBase-TX требует четырех пар проводов категорий 6a, 7 или 7a.

🕮 Подробнее о прокладке кабелей рассказывается в разделе 16.5.

Стандарты 1000BaseTX и 10GBase-TX предназначены для передачи данных по многим парам проводов. Они используют многожильные кабели для передачи данных по линии связи, которые выполняют ее быстрее, чем это может осуществить любая отдельная пара проводов.

Существуют провода с поливинилхлоридной и тефлоновой изоляцией. Выбор изоляции диктуется средой, в которой будут проложены кабели. В замкнутых помещениях, связанных с вентиляционной системой здания, обычно требуется тефлоновая изоляция[4]. Поливинилхлоридная изоляция дешевле и проще в эксплуатации.

🕮 Информация о стандарте RS-232 приведена в разделе 31.1.

Подключая четырехпарный НВП-кабель к коммутационным панелям и настенным розеткам RJ-45, придерживайтесь стандарта разводки TIA/EIA-568A. Этот стандарт, совместимый с другими вариантами RJ-45 (например, RS-232), позволяет избежать ошибок при разводке концов кабеля, независимо от того, есть ли свободный доступ к парам. Требования стандарта отражены в табл. 16.3.

**Таблица 16.3. Стандарт TIA/EIA-568A для подключения четырехпарного НВП-кабеля к розетке RJ-45**

| Пара | Цвета | Контакты разъема |
|---|---|---|
| 1 | Белый/Синий | 5/4 |
| 2 | Белый/Оранжевый | 3/6 |
| 3 | Белый/Зеленый | 1/2 |
| 4 | Белый/Коричневый | 7/8 |

Имеющаяся в здании проводка может не подходить для прокладки сетей, в зависимости от того, как и когда она прокладывалась. В 1980-х годах многие старые здания были переоснащены новыми кабелями. К сожалению, эти кабели обычно не поддерживают скорость передачи данных выше 100 Мбит/с.

## Оптическое волокно

Оптическое волокно используется в тех ситуациях, когда использование медного кабеля по тем или иным причинам неприемлемо. Оптическое волокно передает сигнал быстрее, чем медный провод. Кроме того, оно является более устойчивым к электрическим помехам, что в некоторых приложениях очень важно. Там, где оптическое волокно не является абсолютно необходимым, обычно выбирают медный кабель, поскольку он дешевле и с ним легче работать.

Оптическое волокно бывает "многомодовым" и "одномодовым". Многомодовое оптическое волокно обычно используется в зданиях или комплексах зданий. Оно толще, чем одномодовое, и может проводить несколько лучей света; это свойство позволяет использовать менее дорогую электронику (например, в качестве источника света можно использовать светодиоды).

[4] Конкретную информацию можно получить у пожарного инспектора или ответственного за пожарную безопасность.

Одномодовое оптическое волокно часто используется в магистральных приложениях, например, для прокладки линий связи между городами и регионами. Оно может проводить только один световой луч и требует дорогой прецизионной электроники в конечных точках.

Стандарт TIA-598C рекомендует цветовую кодировку оптического волокна, представленную в табл. 16.4. Следует помнить основное правило: все элементы должны соответствовать друг другу. Оптическое волокно, соединяющее конечные точки, оптические кабели перекрестной коммутации и электронные приборы, установленные в конечных точках, должны иметь один и тот же тип и размер. Обратите внимание на то, что кабели OM1 и OM2 не являются взаимозаменяемыми, хотя и окрашены в один и тот же оранжевый цвет — проверьте размеры, указанные на кабелях, чтобы убедиться, что они соответствуют друг другу. Если вы нарушите это правило, то вам будет сложно обеспечить изоляцию в конечных точках.

**Таблица 16.4. Атрибуты стандартных оптических волокон**

| Количество мод | Название ISO | Диаметр сердечника, мкм | Диаметр оптической оболочки, мкм | Цвет |
|---|---|---|---|---|
| Много | OM1 | 62,5 | 125 | Оранжевый |
| Много | OM2 | 50 | 125 | Оранжевый |
| Много | OM3 | 50 | 125 | Голубой |
| Одна | OS1 | 8–10 | 125 | Желтый |

На концах оптических волокон используются разъемы более чем 30 типов, и нет ни четких правил, ни принципов, регламентирующих их выбор. В каждой конкретной ситуации на выбор того или иного типа разъема влияют поставщики оборудования или параметры оптического волокна, уже проложенного внутри здания.

## Соединение и расширение сетей Ethernet

Сети Ethernet можно соединять с помощью устройств нескольких типов. На выбор устройств, описанных ниже, влияет их стоимость, причем более дешевые устройства описаны в первую очередь. Чем сложнее логические правила, по которым устройства перемещают биты из одной сети в другую, тем больше аппаратного и встроенного программного обеспечения необходимо и тем более дорогой становится сеть.

### *Концентраторы*

Концентраторы (hub) иногда еще называют повторителями (repeators). Это активные устройства, используемые для соединения сегментов сетей Ethernet на физическом уровне. Им требуется внешний источник питания.

Выступая в качестве повторителя, концентратор ретранслирует Ethernet-фреймы, но никак не интерпретирует их. Он "не имеет представления" ни о том, куда направляются пакеты, ни о том, какой протокол они используют. За исключением экзотических ситуаций, *концентраторы больше не должны использоваться в промышленных сетях*, и мы не советуем их использовать даже в домашних сетях. (Почему? Потому что коммутаторы (switches) значительно эффективнее используют полосу пропускания частот в сети и в настоящее время стоят недорого.)

🕮 Сетевые уровни рассматриваются в разделе 14.2.

### Коммутаторы

Коммутатор соединяет сети Ethernet на канальном уровне. Его назначение — объединить две физические сети так, чтобы они выглядели как одна большая физическая сеть. В настоящее время коммутаторы являются промышленным стандартом для соединения устройств Ethernet.

Коммутаторы принимают, регенерируют и ретранслируют пакеты на аппаратном уровне. Они используют алгоритм динамического обучения. Они запоминают, какие исходные адреса поступают с одного порта, а какие — с другого. Пакет переходит из одного порта в другой только при необходимости. Первоначально пересылаются все пакеты, но через несколько секунд, когда коммутатор изучит расположение большинства узлов сети, запускается механизм фильтрации.

Поскольку между сетями пересылаются не все пакеты, каждый сегмент кабеля менее загружен, чем в случае, когда все компьютеры подключены к одному кабелю. А если учесть, что основной трафик имеет тенденцию к локализации, то увеличение реальной пропускной способности может оказаться заметным. Кроме того, коммутатор не влияет на логическую модель сети, поэтому его установка требует лишь незначительного вмешательства со стороны администратора.

Если сеть имеет петли, коммутатор может безнадежно запутаться, потому что пакеты, посылаемые одним компьютером, окажутся сразу на двух (или более) портах коммутатора. В одной сети Ethernet петлей не бывает, но после объединения нескольких таких сетей с помощью маршрутизаторов и коммутаторов топология изменится, вследствие чего может образоваться несколько путей к одному узлу. Некоторые коммутаторы решают эту проблему путем резервирования альтернативных маршрутов на тот случай, если основной маршрут станет недоступным. Они упрощают топологию видимой ими сети, отсекая дублирующиеся пути до тех пор, пока в оставшихся сегментах не окажется только по одному маршруту к каждому узлу сети. Другие коммутаторы создают между сетями двойные каналы и переключают трафик по циклическому принципу.

Коммутаторы должны просматривать каждый пакет, определяя, нужно ли его пересылать в другой сегмент. Производительность этих устройств обычно измеряют как скоростью просмотра пакетов, так и скоростью их пересылки. Многие поставщики не указывают в диаграммах производительности коммутаторов размеры протестированных пакетов, поэтому реальная производительность может быть ниже объявленной.

Несмотря на то что быстродействие коммутаторов Ethernet все время растет, эффективно использовать их можно при объединении в один логический сегмент не более сотни компьютеров. В крупных коммутируемых сетях часто возникают проблемы наподобие "широковещательных штормов", поскольку широковещательный трафик должен проходить через все порты. Для решения этой проблемы нужно изолировать широковещательный трафик между коммутируемыми сегментами посредством маршрутизатора (создавая тем самым более одного логического Ethernet-сегмента).

Выбор коммутатора может представлять определенную трудность. В этом сегменте рынка очень высокая конкуренция, следствием которой являются многочисленные рекламные заявления, не всегда подтверждаемые на практике. Поэтому не стоит особо доверять данным, которые приводятся поставщиками; лучше прислушаться к советам независимых экспертов (просмотрите тесты, приводимые в журналах). В последние годы нередко случалось так, что чей-то продукт оказывался "лучшим" в течение нескольких месяцев, а затем, после попыток внесения улучшений, его производительность или надежность падала ниже критической отметки.

В любом случае убедитесь, что скорость объединительной панели коммутатора является достаточной. У хорошо спроектированного коммутатора эта скорость должна превышать сумму скоростей всех его портов.

### Коммутаторы, позволяющие создавать виртуальные локальные сети

В крупных организациях можно использовать коммутаторы, позволяющие разбивать их порты (программным путем) на группы, называемые виртуальными локальными сетями (VLAN — Virtual Local Area Network). Виртуальная локальная сеть — это группа портов, принадлежащая к одному логическому сегменту, как если бы порты были соединены со своим собственным выделенным коммутатором. Подобное секционирование позволяет повысить степень изоляции трафика, что полезно с точки зрения как безопасности, так и производительности.

Трафиком между виртуальными локальными сетями управляет маршрутизатор или, в некоторых случаях, модуль маршрутизации или уровень программной маршрутизации самого коммутатора. Расширение этой системы, называемое "транкингом виртуальной локальной сети" (один из примеров реализации — протокол IEEE 802.1Q), позволяет разным коммутаторам обслуживать порты одной логической виртуальной локальной сети.

### Маршрутизаторы

Маршрутизаторы (известные также как "коммутаторы третьего уровня") направляют трафик на третьем сетевом уровне модели OSI. Маршрутизаторы доставляют пакеты адресатам на основании информации, хранящейся в TCP/IP-заголовках. Помимо простого перемещения пакетов, маршрутизаторы могут также выполнять ряд особых функций, например фильтрацию пакетов (в соответствии с правилами безопасности), разделение трафика по приоритетам (в соответствии с заданным качеством обслуживания) и обнаружение общей сетевой топологии. Алгоритмы маршрутизации рассматриваются в главе 15.

Конфигурация маршрутизаторов бывает фиксированной или модульной. Устройства первого типа содержат сетевые интерфейсы, установленные в заводских условиях. Они обычно подходят для специализированных применений. Например, маршрутизатор с интерфейсами T1 и Ethernet может оказаться удобным, когда нужно подключить небольшую компанию к Интернету.

Модульные маршрутизаторы имеют слотовую или шинную архитектуру, а интерфейсы к ним добавляются пользователями. Как правило, это более дорогие устройства, но зато они гибче в эксплуатации.

В зависимости от необходимой надежности и ожидаемого трафика, специализированный маршрутизатор может оказаться как дороже, так и дешевле системы UNIX или Linux, сконфигурированной в качестве маршрутизатора. Однако специализированное устройство, как правило, демонстрирует более высокую производительность и надежность. Это та область сетевого проектирования, где лучше заранее вложить чуть больше денег, чем потом иметь головную боль.

### Автосогласование

С появлением разных стандартов Ethernet возникла необходимость, чтобы устройства могли идентифицировать конфигурацию своих соседей и согласовывать с ними свои настройки. Например, сеть не будет работать, если на одной стороне соединения она работает со скоростью 1 Гбит/с, а на другой — со скоростью 10 Гбит/с. Для выявления и решения этой проблемы организацией IEEE был разработан стандарт автосогласования Ethernet. В одних случаях он работает, а в других применяется неправильно и лишь усугубляет проблему.

Следует запомнить два золотых правила автосогласования.

- Вы *обязаны* использовать автосогласование всех интерфейсов, работающих на скорости 1 Гбит/с и выше. Этого требует стандарт.
- Если интерфейсы ограничены скоростями 100 Мбит/с и ниже, необходимо либо конфигурировать *оба конца* соединения, либо вручную настроить скорость и дуплекс (половинный или полный) *обеих* сторон. Если в режиме автосогласования настроить только одну сторону соединения, то в большинстве случаев она не сможет выяснить, какую конфигурацию имеет другая сторона. В результате конфигурация станет несогласованной и производительность упадет.

Для того чтобы выяснить, как задать стратегию автосогласования интерфейсов, прочитайте специальные разделы главы 14, начиная с раздела 14.11.

### Передача электропитания по сетям Ethernet

Технология передачи питания по сетям Ethernet (PoE — Power on Ethernet) основана на передаче электропитания по той же неэкранированной витой паре (UTP Ethernet), по которой передается сигнал Ethernet. Эта технология регламентируется стандартом IEEE 802.3af. Это особенно удобно для систем связи, обеспечивающих передачу речевого сигнала по сети Интернет (VoIP — Voice over IP), или пунктов доступа к системе беспроводной связи (мы указали только два примера, но список можно продолжить), в которых требуется как мощность тока, так и сетевое соединение.

По мощности тока, системы PoE разделяются на четыре класса в диапазоне от 3,84 до 12,95 ватт. Промышленность, которая никогда не останавливается на достигнутом, уже работает над новым стандартом (802.3at), предусматривающим более высокую мощность тока (более 60 ватт). Будет ли этого достаточно, чтобы подключить духовку Easy-Bake к сетевому порту в конференц-зале?[5]

Технология PoE порождает два обстоятельства, о которых должен знать системный администратор.

- Вы должны знать о существовании устройств PoE в вашей инфраструктуре, чтобы правильно спланировать доступ к портам коммутаторов, поддерживающих технологию PoE. Эти порты дороже, чем порты, не поддерживающие технологию PoE.
- Вычисляя расход электроэнергии на обслуживание коммуникационных шкафов, содержащих коммутаторы PoE, следует учитывать мощность устройств PoE. Обратите внимание на то, что вы не должны планировать тот же самый расход электроэнергии на дополнительное охлаждение коммуникационных шкафов, поскольку большая часть тепла, выделяемого из-за потребления мощности PoE, рассеивается за пределами шкафа (обычно по офису).

### Гигантские пакеты

Технология Ethernet стандартизована для типичного пакета размером 1 500 байт (вместе с фреймом — 1 518 байт). Это значение было выбрано давно, когда сети были медленными и память для буферов была дефицитной. В настоящее время пакеты размером 1 500 байт выглядят крохотными в контексте гигабитных сетей Ethernet. Поскольку с каж-

[5] К сожалению, выяснилось, что в духовках Easy-Bake используется электрическая лампа мощностью 100 ватт (правда, в некоторых моделях для освещения используются нагревательные элементы). Это вызывает сомнения по поводу совместимости этих моделей со стандартом IEEE 802.3at. Для интересующихся этим вопросом: да, существует возможность загрузить небольшую систему Linux через порт сети PoE. Найти специальное оборудование для этого мы предоставляем читателям.

дым пакетом связаны накладные расходы и определенное время задержки, производительность сети можно повысить, если допустить более крупные размеры пакетов.

К сожалению, стандарты IEEE для разных типов сетей Ethernet запрещают использование крупных пакетов по соображениям совместимости сетей. Однако поскольку скорость магистрального трафика часто во много раз превышает установленный предел, нестандартные большие пакеты Ethernet в современных сетях перестали быть редкостью. Подстрекаемые нетерпеливыми потребителями, производители сетевого оборудования негласно бойкотируют стандарт IEEE и обеспечивают поддержку крупных фреймов в своей гигабитной продукции.

Для использования так называемых гигантских пакетов (jumbo frames) необходимо лишь повысить максимально возможный размер пакета (MTU — maximal transmission unit) в интерфейсах сети. Повышение производительности зависит от вида трафика, но наибольший выигрыш достигается для крупномасштабных перемещений по протоколу TCP (например, в файловых службах NFSv4 или CIFS). Ожидается, что умеренное, но заметное повышение производительности должно составить примерно 10%.

Тем не менее следует отметить следующее.

- Поддерживать и использовать гигантские пакеты должно все сетевое оборудование в подсетях, включая коммутаторы и маршрутизаторы. Их нельзя смешивать и подгонять.
- Поскольку гигантские пакеты являются нестандартными, обычно их необходимо разрешать явным образом. Устройства могут принимать гигантские пакеты по умолчанию, но, вероятнее всего, они не будут их генерировать.
- Поскольку гигантские пакеты представляют собой незаконное явление, не существует консенсуса, насколько большими они могут или должны быть. Типичной величиной является 9 000 бит или 9 018 вместе с фреймом. Необходимо проверить, какой максимальный размер пакета может принять ваше устройство. Пакеты размером больше 9 Кбайт иногда называют сверхгигантскими, но это экзотическое название вас пугать не должно. Чем больше размер, тем лучше, по крайней мере в диапазоне до 64 Кбайт.
- Гигантские пакеты могут существовать только во внутренних сетях. Интернет не передает такие пакеты.

Мы одобряем использование гигантских пакетов в гигабитных сетях Ethernet, но только там, где это легко и безопасно (например, в сложных промышленных средах их использовать вряд ли стоит). Будьте готовы к дополнительной отладке, если что-то пойдет не так, как надо. Лучше всего развернуть новую сеть, задав максимально возможный размер пакета по умолчанию, а позднее, когда надежность сети будет проверена, изменить эти настройки и разрешить гигантские пакеты.

## 16.2. Беспроводной стандарт: локальная сеть для кочевников

Беспроводные сети состоят из беспроводных точек доступа (WAP — Wireless Access Points) и клиентов беспроводной сети. Точки WAP могут соединяться традиционными проводными сетями (обычная конфигурация) или с другими точками WAP без использования проводов (конфигурация известна под названием "беспроводная сеть").

Точки WAP обычно оснащаются специальным оборудованием, состоящим из одного или нескольких радиоприемников и встроенной операционной системы, которая часто

представляет собой усеченный вариант системы Linux. Одна точка WAP может обеспечить доступ для многих клиентов, но их количество ограничено. Обычно одна точка промышленной сети одновременно обслуживает не более восьми клиентов. Любое устройство, действующее с помощью беспроводного стандарта, поддерживается точкой WAP как клиент.

Распространенными стандартами беспроводных сетей в настоящее время являются IEEE 802.11g и 802.11n. Стандарт IEEE 802.11g работает на частоте 2,4 ГГц и обеспечивает доступ к локальной сети со скоростью, достигающей 54 Мбит/с. Радиус действия одной точки доступа колеблется от 100 м до 40 км, в зависимости от оборудования и физических особенностей местности.

Стандарт 802.11n обеспечивает скорость до 600 Мбит/с[6] и может использовать диапазоны частот как 5 ГГц, так и 2,4 ГГц (при этом рекомендуется использовать диапазон 5 ГГц). Радиус действия точки доступа в стандарте IEEE 802.11n в два раза больше, чем в стандарте IEEE 802.11g.

В настоящее время стандарт IEEE 802.11g и его предшественник IEEE 802.11b используются повсеместно. Трансиверы недороги и встроены в большинство ноутбуков. Кроме того, платы расширения также стоят недорого и легко устанавливаются в любой персональный компьютер.

Для того чтобы сконфигурировать устройство с системой Linux в качестве точки доступа по стандарту 802.11a/b/g, необходимы соответствующее оборудование и драйвер. Поскольку большинство плат беспроводной связи по-прежнему предназначается для системы Microsoft Windows, на заводе могут не установить драйверы для системы Linux.

Прекрасным выбором для организации отдельной станции беспроводной связи в домашних условиях или в офисе, работающей по стандарту 802.11b/g, является точка доступа Airport Express компании Apple. Она напоминает блок питания, который стоит недорого (около 99 долл.) и имеет много функций.[7] Другим вариантом является приобретение коммерческого устройства беспроводного доступа, работающего под управлением усеченной версии системы Linux (например, Open WRT). Более подробную информацию можно найти на сайте `openwrt.com`.

Точки беспроводного доступа выпускают десятки производителей. Их можно купить в интернет-магазинах и даже в бакалейных лавках. Но, как говорится, "скупой платит дважды". Дешевая точка доступа (в пределах 50 долл.), скорее всего, будет плохо работать при передаче больших файлов или при одновременном обслуживании нескольких клиентов.

Отладка беспроводных сетей напоминает шаманство. Решая проблемы, вы должны учитывать множество факторов. Если вы разворачиваете беспроводную сеть промышленного масштаба, то, вероятно, стоит приобрести средства для анализа беспроводных сетей. Мы настоятельно рекомендуем использовать инструменты анализа, выпускаемые компанией AirMagnet.

### Безопасность беспроводных сетей

Традиционно безопасность беспроводных сетей очень низкая. Существует протокол WEP (Wired Equivalent Privacy), применяемый в сетях 802.11b и для шифрования пакетов, передаваемых с помощью радиоволн. К сожалению, в современной версии стандар-

[6] Скорость 600 Мбит/с в стандарте 802.11n является, скорее, теоретической. На практике полоса пропускания в окрестности точки WAP при оптимальной конфигурации может обеспечить скорость передачи данных не более 400 Мбит/с. Это объясняется различием между теоретическими и практическими возможностями оборудования и среды. В беспроводных сетях всякое бывает!

[7] Фактически она также обеспечивает беспроводное соединение для воспроизведения музыки с вашего персонального компьютера или ноутбука.

та была обнаружена фатальная проектная недоработка, которая делает его практически бесполезным. Посторонний человек, находящийся за пределами здания, может получить прямой доступ к сети и остаться незамеченным.

Тем не менее недавно появившиеся стандарты Wi-Fi Protected Access (WPA) возродили доверие к безопасности беспроводных сетей. В настоящее время во всех новых инсталляциях должны использоваться стандарты WPA (в частности, стандарт WPA2), а не WEP. Без применения стандарта WPA2 беспроводные сети должны считаться полностью незащищенными и не должны использоваться за пределами предприятия. Даже дома не используйте стандарт WEP!

Для того чтобы запомнить, что стандарт WEP является незащищенным, а стандарт WPA — безопасным, просто расшифруйте аббревиатуру WAP (Wired Equivalent Privacy — облегченный протокол беспроводных точек доступа). Это название точно отражает суть дела; протокол WEP обеспечивает такую защиту, как проводная сеть, допускающая непосредственное подключение посторонних лиц. (Иначе говоря, никакой защиты — по крайне мере, на уровне IP.)

## Беспроводные коммутаторы и облегченные точки беспроводного доступа

Аналогично тому, как концентраторы Ethernet доросли до уровня коммутаторов Ethernet, беспроводная продукция эволюционирует и постепенно достигает уровня крупных предприятий. Большое количество поставщиков (таких, как компания Cisco) в настоящее время производят "беспроводные коммутаторы", работающие в сочетании с комплектами точек доступа, развернутых в зданиях. Теоретически можно развернуть кучу недорогих точек доступа, а затем централизованно управлять ими с помощью "умного" беспроводного коммутатора. Такой коммутатор хранит информацию о конфигурации стандарта WAP и обеспечивает удобную аутентификацию и роуминг. Эту функциональную возможность обеспечивает протокол LWAPP (Leightweight Wireless Access Point Protocol — облегченный протокол беспроводных точек доступа).

Если вам нужен повсеместный беспроводной доступ в среднем или крупном предприятии, то определенно стоит потратить время на изучение продукции из этой категории. Эта продукция не только сокращает время, затрачиваемое на управление, но и позволяет контролировать качество беспроводной связи, предоставляемой пользователям.

Существует один изящный прием: можно развернуть сеть по стандарту 802.11g/n в своем здании, а затем использовать ее для голосовой связи между сотрудниками через Интернет. Это напоминает бесплатную сотовую сеть!

# 16.3. DSL и кабельные модемы: "последняя миля"[8]

Для крупных компаний не составляет особого труда перемещать большие объемы данных. Такие технологии, как T1, T3, SONET, MPLS и Frame Relay, реализуют достаточно простые каналы обмена данными. Но они не подходят для подключения к сети в домашних условиях. Их стоимость слишком высока, да и не всегда доступно соответствующее оборудование.

В технологии DSL (Digital Subscriber Line — цифровая абонентская линия) используется обычный медный телефонный провод, по которому передаются данные со скоро-

[8] "Последняя миля" (last mile) — это кабельная линия связи между абонентом и телефонной компанией. — *Примеч. ред.*

стью до 24 Мбит/с (правда, для типичного DSL-соединения этот показатель находится в диапазоне от 256 Кбит/с до 5 Мбит/с). В большинстве домов есть телефонная проводка, что делает данную технологию очень удобной для телефонных компаний. Со стороны пользователя DSL-линия заканчивается в устройстве, работающем подобно маршрутизатору TCP/IP. К нему подключаются локальные Ethernet-устройства.

В отличие от обычных телефонных линий и ISDN-соединений, требующих "дозваниваться" до абонента, DSL — это выделенная линия, в которой постоянно есть связь. Это делает технологию DSL еще более привлекательной, поскольку отсутствуют задержки, связанные с начальным конфигурированием и дозвоном.

Существует несколько разновидностей технологии DSL, поэтому название технологии часто приводят в виде *x*DSL, где *x* — префикс разновидности, например: A (асимметричная), S (симметричная), H (высокоскоростная), RA (с адаптируемой скоростью) и I (DSL-на-ISDN). Последний вариант особенно полезен для удаленных от офиса рабочих мест, если скорость передачи данных должна быть выше, чем в обычной технологии DSL. Конкретный вариант реализации и доступная скорость передачи зависят от оборудования, установленного в центральном офисе или у интернет-провайдера.

Подключение к сети сотен миллионов домашних пользователей — заветная мечта многих компаний. Здесь замешаны большие деньги. Технология DSL основана на существующей инфраструктуре телефонных линий, инвестиции в которую позволили операторам местной связи (ILEC — Incumbent Local Exchanges Carriers) извлекать сказочные прибыли, пока мимо них проносилась сетевая революция 80-х–90-х гг.

Компании кабельного телевидения, развивающие собственную оптоволоконную инфраструктуру, предлагают высокоскоростные (хоть и асимметричные) решения для пользователей. В индустрии кабельных модемов не так давно начался процесс стандартизации протоколов передачи данных, и сегодня рекламируется стандарт DOCSIS (Data Over Cable Service Interface Specification — спецификация интерфейса передачи данных по кабельным линиям). Он определяет технические характеристики как кабельных модемов, так и оборудования компании, предоставляющей соответствующие услуги, и позволяет взаимодействовать устройствам различных производителей.

Если сравнивать DSL и кабельную технологию, то выигрыш в конкурентной борьбе достанется тому, что обеспечит более высокую частоту передачи данных в конкретную точку по более низкой цене. Хорошие новости для потребителей в том, что это вынуждает компании инвестировать средства в инфраструктуру для обслуживания жилых районов.

## 16.4. Тестирование и отладка сетей

Основной причиной широкомасштабного перехода к стандарту Ethernet (и к другим технологиям, основанным на использовании неэкранированной витой пары) явилась простота отладки сети. Поскольку эти сети можно проверять посегментно, аппаратные проблемы часто решаются в считанные секунды.

Ключ к отладке сети — ее разбивка на сегменты и тестирование каждого из них до тех пор, пока не будет обнаружена неисправность. Загадочные лампочки на коммутаторах и концентраторах (обозначающие, к примеру, состояние канала и наличие трафика пакетов) помогают быстро выявить источник проблемы. Для того чтобы эти индикаторы работали так, как вы хотите, следует руководствоваться первоклассной документацией.

Как всегда, важно иметь под рукой нужные инструменты, чтобы выполнить работу правильно и без проволочек. На рынке предлагаются средства сетевой отладки двух типов (правда, наблюдается тенденция к их объединению).

Устройство первого типа — ручной кабельный тестер. Он измеряет электрические характеристики кабеля, включая его длину (для этого применяется особая технология, называемая рефлектометрией во временной области). Такие устройства способны выявлять простейшие проблемы, например разрыв или неправильную разводку кабеля. Нашим любимым инструментом тестирования локальных сетей является устройство Fluke LanMeter. Это универсальный анализатор, способный даже посылать эхо-пакеты протокола ICMP. Профессиональные варианты этого оборудования описаны на специальном сайте. Для телекоммуникационных сетей WAN лучше всего подходит тестер T-BERD, выпускаемый компанией JDSU (`jdsu.com`).

Средства отладки второго типа — это анализаторы сетевых пакетов. Они просматривают сетевые пакеты на предмет наличия ошибок протоколов, неправильной конфигурации и прочего беспорядка. Эти анализаторы работают на уровне каналов, а не на электрическом уровне, поэтому они не могут распознавать проблемы, связанные с физическими повреждениями кабелей или электропитанием.

Существуют профессиональные анализаторы сетевых пакетов, но мы нашли свободно распространяемую программу Wireshark (`wireshark.org`), которая может выполняться на полнофункциональном ноутбуке. Именно ее можно считать наилучшим выбором.[9] Более подробная информация об анализаторах сетевых пакетов приведена в разделе 21.7.

# 16.5. Прокладка кабелей

Если вы занялись прокладкой кабелей в здании, то самый ценный совет, который мы можем вам дать, звучит так: "Делайте все правильно с первого раза". Это не та область, в которой можно скупиться или халтурить. Покупая качественные материалы, выбирая компетентного подрядчика для прокладки кабелей и инсталлируя дополнительные разъемы (отводы), вы избегаете многолетних мучений.

## Неэкранированная витая пара

Кабель категории 6a имеет наилучшее соотношение цены и производительности на современном рынке. Его стандартный вариант — четыре пары проводов под одной оболочкой, что подходит для большинства соединений, включая RS-232 и гигабитные линии.

Спецификации кабеля категории 6a требуют, чтобы скрутка провода заканчивалась в точке контакта. Для того чтобы обеспечить это требование, необходимы специальное обучение и оконечное оборудование. При этом необходимо использовать настенные розетки и коммутационные панели категории 6a. Самые хорошие отзывы заслужила продукция компании Siemon (`www.siemon.com`).

## Офисные точки подключения

Одной точки подключения на офис явно недостаточно. Сколько же нужно — две или четыре? Мы рекомендуем четыре, обосновывая это следующими причинами.

- Их можно использовать просто для подключения телефонов.
- Их можно применять для подключения портативных или демонстрационных компьютеров.

[9] Как и многие популярные программы, программа Wireshark часто подвергается атакам хакеров. Убедитесь, что вы используете самую последнюю версию.

- Стоимость материалов составляет, как правило, всего 5–10% от общей стоимости прокладки сети.
- Как правило, все предполагаемые оценки следует умножать на два.
- Гораздо дешевле проложить весь кабель сразу, чем делать это поэтапно.
- Если порты работают медленно, то люди обычно добавляют коммутаторы с четырьмя или восемью портами, купленные в ближайшем специализированном магазине, а потом жалуются на форумах на низкую скорость соединения.

При прокладке кабеля в здании можно установить дополнительные розетки в коридорах, конференц-залах, столовых, туалетных комнатах и на потолках (для точек беспроводного доступа). Однако не забывайте о безопасности и размещайте открыто предоставляемые порты на "гостевой" виртуальной локальной сети, не допуская посторонних к своим внутренним сетевым ресурсам.

## Стандарты кабельных систем

Необходимость обеспечения всех видов деятельности внутри современных зданий обусловливает потребность в крупной и сложной кабельной инфраструктуре. Заглянув в обычный коммутационный шкаф, вы будете потрясены, увидев его стенки, сплошь покрытые непомеченными проводами одного цвета.

С целью улучшения оперативного контроля и стандартизации кабельных систем зданий, в феврале 1993 года организация TIA опубликовала административный стандарт на телекоммуникационную инфраструктуру коммерческих зданий (TIA/EIA-606). Этот стандарт устанавливает требования и принципы идентификации и документирования телекоммуникационной инфраструктуры. Он касается следующих аспектов:

- оконечной аппаратуры;
- кабелей;
- прокладки кабелей;
- расстояний между элементами оборудования;
- цветовой маркировки;
- символических обозначений стандартных компонентов.

В частности, определены стандартные цвета маркировки проводов (табл. 16.5).

**Таблица 16.5. Таблица цветовой маркировки по стандарту TIA/EIA-606**

| Тип оконечного устройства | Цвет | Код [а] | Комментарии |
|---|---|---|---|
| Граничное | Оранжевый | 150C | Центральная телефонная станция |
| Сетевые соединения | Зеленый | 353C | Также применяется для вспомогательных электросетей |
| Общее оборудование [б] | Фиолетовый | 264C | Основное оборудование коммутации и передачи данных |
| Магистраль первого уровня | Белый | — | Кабели |
| Магистраль второго уровня | Серый | 422C | Кабели |
| Станция | Синий | 291C | Горизонтальные кабели |
| Магистраль между зданиями | Коричневый | 465C | Кампусные кабели |
| Разное | Желтый | 101C | Служебные и сигнальные линии |

*Окончание табл. 16.5*

| Тип оконечного устройства | Цвет | Код* | Комментарии |
|---|---|---|---|
| Ключевые телефонные системы | Красный | 184C | — |

[а] В соответствии с цветовой моделью Pantone.

[б] Офисные АТС, компьютеры, локальные сети, мультиплексоры и т.д.

# 16.6. Проектирование сетей

В этом разделе рассматриваются вопросы, связанные с логическим и физическим проектированием сетей среднего размера. Представленные здесь идеи подходят для нескольких сотен узлов, но неприменимы ни для трех, ни для нескольких тысяч компьютеров, включенных в одну сеть. Также предполагается, что работа будет начата с нуля.

Основной объем работ по проектированию сети состоит из определения:

- типов сред передачи;
- топологии и способов прокладки кабелей;
- системы концентраторов, коммутаторов и маршрутизаторов.

Еще один ключевой вопрос проектирования сети связан с управлением перегрузкой. Например, файловые протоколы NFS и CIFS очень сильно загружают сеть, поэтому такие файловые системы нежелательно подключать по магистральному кабелю.

Ниже анализируются аспекты, которые необходимо учитывать при проектировании сети.

## Структура сети и архитектура здания

Структуру сети проще изменить, чем архитектуру здания, но обе они должны нормально сосуществовать. Если вам крупно повезло, т.е. представилась возможность проектировать сеть до постройки здания, будьте щедрым. К сожалению, в большинстве случаев здание и отдел технического обслуживания компании на момент проектирования сети уже существуют и налагают жесткие ограничения на структуру сети.

В готовых зданиях сеть должна адаптироваться к архитектуре, а не противостоять ей. В современных зданиях, помимо высоковольтной электропроводки, водо- и газопроводов, иногда имеются каналы для прокладки кабелей. Часто монтируются подвесные потолки — настоящий подарок для тех, кто прокладывает сеть. Во многих университетских городках существуют туннели, которые облегчают создание сетей.

Необходимо следить за целостностью брандмауэров[10]. При прокладке кабеля через брандмауэр отверстие должно соответствовать диаметру кабеля и заполняться негорючим веществом. Выбирая кабель, учитывайте наличие приточной вентиляции. Если узнают, что вы нарушили правила пожарной безопасности, вас могут оштрафовать и заставить устранить недостатки, даже если для этого придется проложить заново всю сеть.

Логическая структура сети должна соответствовать физическим ограничениям зданий, в которых она будет функционировать. Приступая к проектированию, помните, что можно найти логически красивое решение, а затем вдруг обнаружить, что реализовать его физически сложно или вообще невозможно.

[10] Речь идет о брандмауэрах в виде бетонных, кирпичных или огнеупорных стен, которые препятствуют распространению огня по всему зданию. Значительно отличаясь от сетевых брандмауэров, они не менее важны.

## Расширение сетей

Прогнозировать потребности на десять лет вперед очень сложно, особенно в области вычислительной техники и сетей. Поэтому, проектируя сеть, всегда следует учитывать перспективы ее расширения и увеличения пропускной способности. Прокладывая кабель, особенно в труднодоступных местах, протягивайте в три-четыре раза больше пар, чем нужно. Помните: основная часть стоимости прокладки сети приходится на оплату труда, а не на материалы.

Даже если волоконно-оптические линии не планируется использовать немедленно, разумно будет все же проложить немного оптического волокна, особенно если известно, что впоследствии протянуть его будет гораздо труднее. Прокладывайте и многомодовый, и одномодовый кабели. Как правило, нужным оказывается как раз тот кабель, который не проложен.

## Перегрузка

Сеть — как цепь: ее качество определяется самым слабым или самым медленным звеном. Производительность Ethernet, как и многих других сетевых технологий, при увеличении нагрузки падает.

Активно эксплуатируемые коммутаторы, нестыкующиеся интерфейсы, низкоскоростные каналы связи — все это может привести к перегрузке. Эффективный способ борьбы с ней заключается в локализации трафика путем создания подсетей и установки маршрутизаторов. Подсети можно использовать и для изоляции компьютеров, задействованных в отдельных экспериментах. Трудно проводить эксперимент на нескольких компьютерах, если нет надежного способа изолировать их физически и логически от остальной части сети.

## Обслуживание и документирование

Опыт показывает, что удобство обслуживания сети напрямую зависит от качества документации на нее. Точная, полная, своевременно корректируемая документация абсолютно необходима.

Кабели следует маркировать во всех точках подключения. Рекомендуем вкладывать копии местных монтажных схем в коммутационные шкафы, чтобы при всех изменениях эти экземпляры можно было откорректировать на месте. Каждые несколько недель необходимо переносить все корректировки в электронную базу данных.

Стыки между крупными системами в виде коммутаторов или маршрутизаторов могут упростить отладку, поскольку позволяют изолировать части сети и отлаживать их по отдельности. Полезно также разграничивать административные области.

# 16.7. Управление сетью

Если необходимо обеспечить нормальную работу сети, одни функции управления следует централизовать, другие — распределить, а третьи — оставить на локальном уровне. Требуется сформулировать и согласовать обоснованные "правила поведения добропорядочных граждан".

Типичная крупномасштабная среда включает в себя:

- магистральную сеть, соединяющую здания;

- сети подразделений, подключенные к магистрали;
- подсети рабочих групп в рамках подразделения;
- соединения с внешним миром (например, с Интернетом или периферийными филиалами).

При проектировании и реализации сетей следует предусматривать централизованные контроль, ответственность, сопровождение и финансирование. Поскольку подразделения, как правило, стремятся свести к минимуму собственные расходы, быстро растет число сетей с централизованной оплатой каждого соединения. Вот основные объекты централизованного управления:

- структура сети, в том числе принципы использования подсетей, маршрутизаторов, коммутаторов и т.д.;
- магистральный кабель, в том числе подключения к нему;
- IP-адреса и имена компьютеров, доменные имена;
- используемые протоколы (требуется обеспечить их взаимодействие);
- правила доступа в Интернет.

Имена доменов, IP-адреса и сетевые имена компьютеров в определенном смысле уже находятся под централизованным контролем таких организаций, как ARIN (American Registry for Internet Numbers) и ICANN, но координация использования этих элементов на локальном уровне также необходима.

Центральный орган управления имеет общее представление о сети, ее структуре, производительности и перспективах роста. Он может позволить себе иметь собственное контрольное оборудование (и обслуживающий его персонал) и следить за нормальной работой магистральной сети. Центральный орган может настоять на правильном выборе структуры сети, даже если для этого придется заставить подразделение купить маршрутизатор и создать подсеть для подключения к магистрали. Такое решение иногда необходимо для того, чтобы новое соединение не навредило работе существующей сети.

Если в сети работают разнородные компьютеры, операционные системы и протоколы, обязательно нужно иметь "высокоинтеллектуальный" маршрутизатор (например, компании Cisco), который будет служить шлюзом между сетями.

# 16.8. Рекомендуемые поставщики

Занимаясь более 20 лет инсталляцией сетей по всему миру, мы не раз обжигались на продуктах, которые не соответствовали спецификациям, имели завышенную цену, неправильно указанные характеристики или как-то иначе не оправдывали ожидания. Ниже приведен список поставщиков, которым мы доверяем и услугами которых рекомендуем пользоваться.

## Кабели и разъемные соединения

AMP (подразделение Tyco)
(800) 522-6752
amp.com

Belden Cable
(800) 235-3361
(765) 983-5200
belden.com

Anixter
(800) 264-9837
anixter.com

Newark Electronics
(800) 463-9275
newark.com

Black Box Corporation
blackbox.com

The Siemon Company
(860) 945-4395
siemon.com

### Тестовые приборы

| | | |
|---|---|---|
| Fluke | JDSU | Siemon |
| (800) 443-5853 | (866) 228-3762 | (800) 945-4395 |
| fluke.com | jdsu.com | siemon.com |

### Маршрутизаторы/коммутаторы

| | |
|---|---|
| Cisco Systems | Juniper Network |
| (415) 326-1941 | (408) 745-2000 |
| www.cisco.com | juniper.com |

## 16.9. Рекомендуемая литература


- Barnett David, Groth David and Jim McBee. *Cabling: The Complete Guide to Network Wiring (3rd edition)*. San Francisco: Sybex, 2004.
- Seifert Rich. *Gigabit Ethernet: Technology and Applications for High Speed LANs*. Reading, MA: Addison-Wesley, 1998.
- ANSI/TIA/EIA-568-A. *Commercial Building Telecommunications Cabling Standard*, и ANSI/TIA/EIA-606, *Administration Standard for the Telecommunications Infrastructure of Commercial Buildings*.

  Это стандарты телекоммуникационной промышленности для построения кабельных систем зданий. К сожалению, они доступны не бесплатно. Посетите веб-сайт www.tiaonline.org.
- Spurgeon Charles. *Guide to Ethernet*; ethermanage.com/ethernet.


## 16.10. Упражнения

16.1. Сегодня для прокладки компьютерных сетей в офисных зданиях чаще всего используется неэкранированная витая пара. Для поддержки таких сетей требуется определенное сочетание концентраторов и коммутаторов. Во многих случаях устройства обоих типов оказываются взаимозаменяемыми. Перечислите преимущества и недостатки этих устройств.

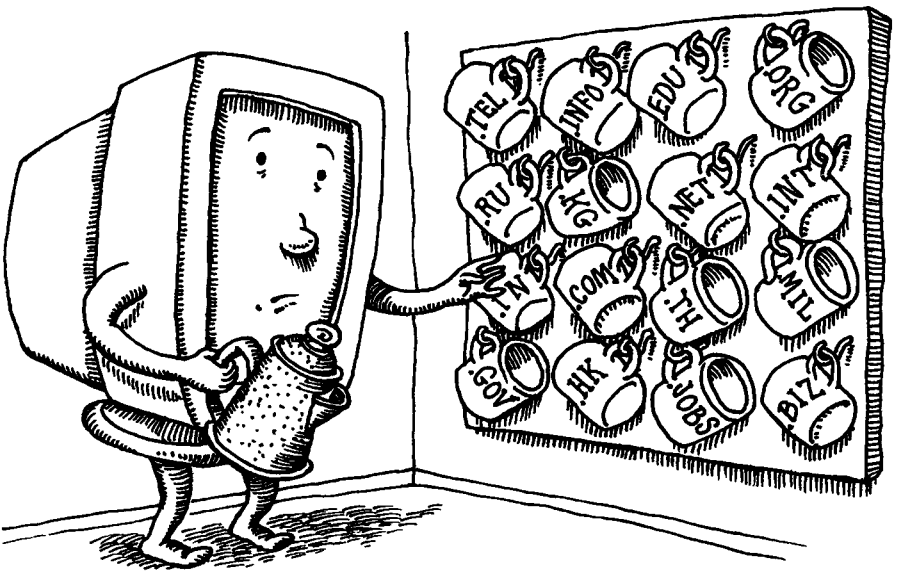
16.2. ★ Нарисуйте схему воображаемой сети, соединяющей компьютер вашей лаборатории с узлом Amazon.com. Покажите компоненты локальных, общегородских и глобальных сетей и перечислите технологии, применяемые на каждом уровне. Укажите, где могут использоваться концентраторы, коммутаторы и маршрутизаторы.

16.3. ★ Исследуйте протокол Temporal Key Integrity Protocol из семейства WPA2. Объясните его преимущества перед протоколом WEP и назовите атаки, которые он может предотвратить.

16.4. ★★ TTCP — средство измерения производительности протоколов TCP и UDP. Инсталлируйте пакет на двух компьютерах и замерьте производительность соединения между ними. Что происходит с пропускной способностью сети при изменении размеров сетевых буферов? Как полученные результаты согласуются с теоретической пропускной способностью физической среды?

# Глава 17

# Система доменных имен




К Интернету подключено огромное множество компьютеров. Как управлять ими, если они расположены в разных странах и принадлежат разным сетям и административным группам? Этой цели служат два элемента глобальной сетевой инфраструктуры: система доменных имен (Domain Name System, DNS), которая отслеживает информацию об именах и адресах компьютеров, и система маршрутизации в Интернете, контролирующая соединения между компьютерами.

Система доменных имен предназначена для нескольких целей, но основная из них — преобразование имен компьютеров в IP-адреса и наоборот. Пользователям и программам пользовательского уровня удобнее ссылаться на компьютеры по именам, но низкоуровневое сетевое программное обеспечение понимает только числовые адреса. Система DNS играет роль промежуточного связующего звена. Она также важна для маршрутизации электронной почты и организации доступа к веб-серверам.

Система DNS представляет собой распределенную базу данных. Термин "распределенная" означает, что сведения о компьютерах хранятся на серверах, которые автоматически вступают в контакт друг с другом, запрашивая данные и обмениваясь информацией.

🕮 Информация о документах RFC изложена в разделе 14.1.

Система DNS описана в ряде документов RFC, последний из которых имеет номер 108. Существует несколько реализаций этой системы, которые отличаются друг от друга по функциональным возможностям, целям и степени соответствия документам RFC. Доли рынка, занимаемые этими реализациями, приведены в табл. 17.1.

В главе содержится общая информация о системе DNS и рутинных действиях системного администратора, связанных с реализациями серверов имен BIND, NSD и Unbound. Приведенные примеры относятся к серверам BIND 9.7, NSD 3.2.4 и Unbbound 1.4.1.

**Таблица 17.1. Некоторые распространенные реализации системы DNS**

| Название | Автор | Источник | Доля[а] | Примечания |
|---|---|---|---|---|
| BIND | ISC | `isc.org` | 80,3% | Авторизация или кеширование |
| Microsoft DNS | Microsoft | `microsoft.com` | 15,4% | Мириады уведомлений о статических IP-адресах |
| djbdns[б] | Dan Bernstein | `tinydns.org` | 2,6% | Не соответствует некоторым документам RFC |
| PowerDNS | PowerDNS BV | `powerdns.com` | 0,7% | Только авторизация |
| NSD[в] | NLnet Labs | `nlnetlabs.nl` | <0,1% | Только авторизация, очень быстрая |
| Unbound | NLnet Labs | `unbound.net` | – | Только кеширование, быстрое |

[а] Доля рынка в соответствии с июльским обзором Internet Domain Survey 2009 года, размещенным на сайте `ics.org`.

[б] Известна также под названием tinydns в качестве компонента пакета djbdns.

[в] Задумывалась для корневых серверов и серверов домена верхнего уровня; в настоящее время используется повсеместно.

Может возникнуть вопрос, зачем вообще описывать здесь серверы NSD и Unbound, если их рыночная доля такая маленькая. На это есть три причины.

- Для того чтобы развернуть действительно надежную среду DNS, необходимо, чтобы не все серверы работали на одном и том же программном обеспечении. Успешная атака на систему DNS вашего сайта отключит вас от Интернета. Разнообразие программного и аппаратного обеспечения, а также средств, обеспечивающих связность сети, является ключевым фактором выживания в Интернете. Добавьте к этому географическое расположение и опыт системного администратора, и вы будете в прекрасной форме.
- Серверы NSD и Undound работают намного быстрее, чем BIND.
- И наконец, среди всех реализаций серверов имен только BIND и NSD/Unbound реализуют систему DNSSEC — криптографическое расширение системы DNS.

В настоящее время многие сайты (возможно, большинство) не используют ни сервер BIND, ни сервер NSD/Unbound, предпочитая службу Active Directory компании Microsoft. Эта служба кратко описана в разделе 30.9.

# 17.1. Основные задачи системы DNS

Система DNS определяет следующее:

- иерархическое пространство имен компьютеров и IP-адресов;
- таблицу имен и адресов компьютеров, реализованную в виде распределенной базы данных;
- "распознаватель" — клиентскую библиотеку функций, осуществляющих запросы к базе данных DNS;
- усовершенствованные средства маршрутизации и аутентификацию отправителей сообщений электронной почты;
- механизм поиска служб в сети;
- протокол обмена информацией об именах.

DNS — это клиент-серверная система. Серверы (называемые серверами имен) загружают данные из клиентских DNS-файлов в память и пользуются ими при ответах на запросы как от внутренних клиентов, так и от внешних компьютеров. Все компьютеры, подключенные к сети, должны быть клиентами системы DNS, но лишь некоторые из них обязаны быть серверами имен.

## Управление системой DNS

В небольшой организации (несколько узлов в одной сети) можно запустить сервер на одном из компьютеров или попросить интернет-провайдера предоставить услуги DNS. Организация среднего размера с несколькими подсетями должна иметь несколько серверов DNS, чтобы сократить время выполнения запросов и повысить общую надежность сети. Очень большая организация может разделить свой домен на поддомены и запустить в каждом из них несколько серверов.

Прямое преобразование DNS связывает имя компьютера с IP-адресом. Обратное преобразование ставит в соответствие этому адресу имя компьютера. Прямые и обратные преобразования по возможности должны осуществляться в одном месте. Некоторые провайдеры с удовольствием отдают контроль над прямым преобразованием, но отказываются делать то же самое в отношении обратного преобразования. Подобное разделение обязанностей ведет к проблемам синхронизации. В разделе 17.8 описан элегантный прием, позволяющий управлять делегированием даже крошечных фрагментов адресного пространства.

Домены DNS должны обслуживаться как минимум двумя серверами, хотя рекомендуется использовать три сервера, расположенных в разных местах. Чаще всего один из этих серверов назначается главным (первичным), на котором хранится копия данных домена. Остальные серверы являются резервными (вторичными). Они загружают информацию из главного сервера организации.

Некоторые организации управляют только главным сервером, а серверы провайдера являются резервными. После того как система сконфигурирована, серверы провайдера автоматически загружают информацию из главного сервера организации. Изменения в конфигурации DNS отражаются на резервных серверах без вмешательства администраторов любой из сторон.

Кроме того, часто все службы DNS реализуют за пределами организации, полагаясь на разнообразие компаний, их надежность и географическое распределение.

Не размещайте все серверы DNS в одной сети. Если по какой-то причине она окажется недоступной, пользователи других сетей не смогут работать. Распределите серверы DNS так, чтобы функционирование системы не зависело от единственного звена. При правильном конфигурировании DNS становится высоконадежной системой.

# 17.2. Как работает система DNS

Каждый компьютер, пользующийся услугами службы DNS, является либо клиентом этой системы, либо одновременно и клиентом, и сервером. Читатели, которые не планируют запускать серверы DNS, следующие несколько разделов могут пропустить, перейдя сразу к разделу 17.3. Отметим, однако, что приведенная информация позволит глубже понять принципы работы системы доменных имен.

## Записи о ресурсах

Каждая организация поддерживает один или несколько фрагментов распределенной базы данных, лежащей в основе всемирной системы DNS. Ваш фрагмент данных состоит из текстовых файлов, содержащих записи о каждом компьютере вашей сети; эти записи называются "записями о ресурсах". Каждая запись представляет собой отдельную строку, состоящую из имени (обычно имени компьютера), типа записи и некоторых значений. Поле имени можно не указывать, если его значение совпадает с именем в предыдущей строке.

Например, строки

```
nubark      IN   A    63.173.189.1
            IN   MX   10 mailserver.atrust.com.
```

в файле "прямого преобразования" (с именем **atrust.com**) и строка

```
1           IN   PTR nubark.atrust.com.
```

в файле "обратного преобразования" (с именем **63.173.189.rev**) связывают сайт `nubark.atrust.com` с IP-адресом 63.173.189.1. Запись MX перенаправляет сообщение электронной почты, адресованное на эту машину, на компьютер `mailserver.atrust.com`.

Записи о ресурсах — это универсальный язык системы DNS. Они не зависят от файлов конфигурации, управляющих операциями, которые выполняются на любой реализации данного сервера DNS. Все они являются фрагментами данных, циркулирующих внутри системы DNS, и кешируются в разных местах.

## Делегирование

Все серверы имен считывают имена корневых серверов из локальных файлов конфигурации или содержат их в своем коде. Корневые серверы "знают" о доменах `com`, `org`, `edu`, `fi`, `de` и других доменах верхнего уровня. Эта цепочка продолжается дальше: сервер домена `edu` "знает" о домене `colorado.edu`, `berkely.edu`, сервер домена `com` "знает" о домене `admin.com` и т.д. Каждая зона может делегировать полномочия по управлению своими поддоменами другим серверам.

Рассмотрим реальный пример. Предположим, требуется узнать адрес узла `vangogh.cs.berkeley.edu`, находясь на узле `lair.cs.colorado.edu`. Компьютер `lair` просит локальный сервер имен, `ns.cs.colorado.edu`, найти ответ на этот вопрос. Последующие события представлены на рис. 17.1.

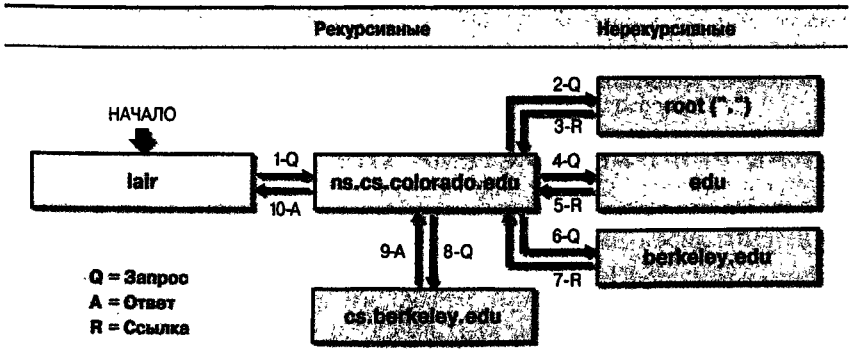



*Рис. 17.1. Обработка запроса в DNS*

Числа возле стрелок определяют порядок событий, а буквы — тип транзакции (запрос, ответ или ссылка). Предполагается, что никакие из требуемых данных предварительно не кешировались, за исключением имен и IP-адресов серверов корневого домена.

Локальный сервер имен не "знает" адреса компьютера `vangogh`. Более того, ему ничего не известно о доменах `cs.berkeley.edu`, `berkeley.edu` и даже `edu`. Он "знает" лишь некоторые серверы корневого домена, запрашивает корневой домен об узле `vangogh.cs.berkeley.edu` и получает ссылку на серверы в домене `edu`.

Локальный сервер имен является рекурсивным. Если ответ на запрос содержит ссылку на другой сервер, то локальный сервер повторно направляет запрос новому серверу. Он продолжает этот процесс, пока не найдет требуемый сервер.

В нашем примере локальный сервер имен посылает запрос серверу домена `edu` (как всегда, запрашивая компьютер `vangogh.cs.berkeley.edu`) и получает ссылку на независимые серверы домена `berkeley.edu`. Затем локальный сервер повторяет запрос, направляя его в домен `berkeley.edu`. Если сервер университета Беркли не содержит ответ в кеше, он вернет ссылку на домен `cs.berkeley.edu`. Сервер этого домена компетентен в отношении запрашиваемой информации и возвращает адрес компьютера `vangogh`.

По окончании процедуры в кеше сервера `ns.cs.colorado.edu` окажется адрес компьютера `vangogh`. В кеше будут также находиться списки серверов доменов `edu`, `berkeley.edu` и `cs.berkeley.edu`.

Детали процедуры запроса можно выяснить с помощью утилит **`dig +trace`** или **`drill -T`**.[1]

## Кеширование и эффективность

Кеширование повышает эффективность поиска: кешированный ответ выдается почти мгновенно и обычно точен, так как адресно-именные соответствия меняются редко. Ответ хранится в течение периода времени, называемого TTL (time to live), продолжительность которого задается владельцем искомой записи. Большинство запросов касается локальных компьютеров и обслуживается быстро. Повышению эффективности невольно содействуют и сами пользователи, так как многие запросы повторяются.

Обычно записи о ресурсах вашей организации должны использовать период TTL, продолжительность которого, как правило, лежит в диапазоне от одного часа до одного дня. Чем дольше период TTL, тем меньше трафик сети, потребляемый интернет-клиентами, получающими свежие копии записи.

Если у вас есть специальная служба, загрузка которой сбалансирована между логическими подсетями (этот процесс называется "балансированием загрузки глобального сервера"), вы можете потребовать, чтобы поставщик услуг, выполняющий балансирование загрузки, выбрал более короткую продолжительность TTL, например десять секунд или одну минуту. (Короткий период TTL позволяет балансировщику загрузки быстро реагировать на бездействие серверов и атаки на основе отказа в обслуживании.) Система с короткими периодами TTL продолжает работать корректно, но ваши серверы имен должны работать в более интенсивном режиме. В примере, описанном выше, продолжительность периодов TTL была установлена так: на корневых серверах — 42 дня, в домене `edu` — два дня, в домене `berkeley.edu` — два дня и один день — для сайта `vangough.cs.berkeley.edu`. Это разумные величины. Если вы планируете крупную перенумерацию, то можете сделать периоды TTL короче, чем раньше.

[1] Утилиты ***dig*** и ***drill*** — это инструменты системы DNS; утилита ***dig*** входит в дистрибутивный набор сервера BIND, а утилита ***drill*** разработана группой NLnet Labs.

Серверы DNS также реализуют негативное кеширование. Иначе говоря, они помнят, в каких ситуациях запрос остался без ответа, и не повторяют его, пока не истечет период TTL для негативного кеширования. Ответы при негативном кешировании сохраняются в следующих ситуациях.

- Нет узла или домена, соответствующего запрашиваемому имени.
- Данные запрашиваемого типа для данного узла не существуют.
- Запрашиваемый сервер не отвечает.
- Сервер недоступен из-за проблем в сети.

Сервер BIND кеширует данные в двух первых ситуациях, а сервер Unbound — во всех четырех. Количество повторений негативного кеширования можно задавать в любой реализации.

## Неоднозначные ответы

Сервер имен в ответ на запрос часто получает несколько записей. Например, при попытке узнать адрес сервера имен корневого домена можно получить список, содержащий все 13 корневых серверов. Большинство серверов имен возвращает ответы в случайном порядке, выполняя примитивный вид балансирования загрузки.

Можно достичь балансирования загрузки своих серверов, закрепив одно имя за несколькими IP-адресами (которые в реальности соответствуют разным компьютерам).

```
www             IN   A    192.168.0.1
                IN   A    192.168.0.2
                IN   A    192.168.0.3
```

Интенсивно эксплуатируемые веб-серверы, такие как Yahoo! и Google, в действительности не являются одним компьютером. Они просто известны под одним доменным именем в DNS.

# 17.3. DNS для нетерпеливых: подключение нового компьютера

Прежде чем погружаться в детали функционирования DNS, ответим на наиболее часто задаваемые вопросы.

- Как подключить новый компьютер к сети, где уже используется сервер имен?
- Как указать в конфигурации, что новый компьютер является клиентом системы DNS?

Ниже приводится алгоритм, в котором не определяются и не объясняются никакие термины и который, возможно, не точно соответствует правилам и процедурам системного администрирования, принятым в вашей организации. Отнеситесь к нему с осторожностью и изучите документ RFC1912 Common DNS Operational and Configuration Errors.

## Добавление новой машины в систему DNS

Если ваша сеть настроена на использование протокола DHCP (Dynamic Host Configuration Protocol), то вы можете вообще не выполнять никаких действий для конфигурирования системы DNS. При подключении нового компьютера сервер DHCP информирует его о серверах DNS, которым он должен посылать запросы. Отображение

"имя-адрес" для использования компьютера во внешнем мире обычно задается при настройке конфигурации сервера DHCP, и оно автоматически вводится с помощью средств динамического обновления системы DNS.

В сетях, не использующих протокол DHCP, для обновления конфигурации системы DNS необходимо выполнить следующий алгоритм, в ходе которого происходит копирование и редактирование записей об аналогичном компьютере.

**Этап 1.** Выберите для нового компьютера свободные сетевое имя и IP-адрес, согласовав их с системным администратором или интернет-провайдером.

**Этап 2.** Найдите похожий компьютер в той же подсети. Мы воспользуемся записями этого компьютера в качестве модели для новых записей. В нашем примере мы используем в качестве модели компьютер с именем `templatehost.example.com` в подсети 208.77.188.0/24.

**Этап 3.** Зарегистрируйтесь на компьютере, который является главным сервером имен. Если вы не знаете, какая машина является главным сервером, то для его идентификации вы можете использовать команду **dig** (**dig SOA** *имя_домена*). (Если утилита **dig** не инсталлирована, можно использовать утилиту **drill**.)

**Этап 4а** (для организаций, использующих серверы BIND).

- Найдите файл конфигурации; как правило, это файл **/etc/named.conf**.
- Найдите в инструкции `options` файла **named.conf** строку `directory`, где сообщается о местонахождении зонных файлов в системе (см. раздел 17.9). В этих файлах хранятся реальные имена и IP-адреса компьютеров.
- Найдите в инструкциях `zone` имена файлов зон прямого и обратного преобразований для сети, к которой принадлежит новый IP-адрес (см. раздел 17.9).
- Проверьте по инструкциям `zone`, действительно ли данный сервер является главным сервером домена (имеет тип `master`, а не `slave` или какой-то другой тип). Если это не так, вы зашли в другую систему! Инструкция в зоне прямого преобразования в файле **/etc/named.conf** должна выглядеть примерно следующим образом.

```
zone "example.com" {
  type master;
  file "имя_файла";
...
```

Инструкция в зоне обратного преобразования должна выглядеть примерно так.

```
zone "188.77.208.in-addr.arpa" {
  type master;
  file "имя_файла";
...
zone "example.com" {
  type master;
  file "имя_файла";
...
```

**Этап 4б** (для организаций, использующих серверы NSD).

- Найдите файл конфигурации сервера NSD **/etc/nsd/nsd.conf**.
- Найдите инструкцию `zone`, соответствующую вашему домену в файле **nsd.conf** (каждая зона идентифицируется ключевым словом `name`).
- Проверьте, действительно ли данный сервер является главным сервером домена! Инструкция в зоне, соответствующей вашему домену, будет содержать предложение

provide-xfr. Если она содержит предложение request-xfr, то это вторичный сервер данной зоны и вы находитесь на неправильном компьютере. Инструкция в зоне прямого преобразования должна выглядеть примерно следующим образом.

```
zone:
  name: example.com
  zonefile: /var/nsd/prinary/example.com;
  provide-xfr: ip-адрес tsig.key.name
  notify: ip-адрес NOKEY
```

Инструкция в зоне обратного преобразования должна выглядеть примерно так.

```
zone:
  name: 188.77.208.in-addr.arpa
  zonefile: /var/nsd/prinary/188.77.208.in-addr.arpa
  provide-xfr: ip-адрес tsig.key.name
  notify: ip-адрес NOKEY
```

- Запишите имена файлов, указанных как аргументы после ключевого слова zonefile в зонах прямого и обратного преобразований.

**Этап 5.** Перейдите в каталог файлов зон и отредактируйте файл зоны прямого преобразования. Найдите записи, соответствующие компьютеру-прототипу. Они будут выглядеть примерно так.

```
templatehost    IN   A    128.138.243.100
                IN   MX   10 mail-hub
                IN   MX   20 templatehost
```

Ваша версия может не содержать строк MX, которые используются для маршрутизации почты. Кроме того, ваши зонные файлы могут не содержать спецификатор IN (по умолчанию) или использовать прописные буквы.

**Этап 6.** Скопируйте эти записи и модифицируйте их должным образом для нового компьютера. Записи зонного файла могут быть отсортированы по именам компьютеров, поэтому следуйте принятым соглашениям.

**Этап 7.** Измените порядковый номер записи SOA, расположенной в начале файла (это первый из пяти числовых параметров записи). Порядковый номер может только возрастать. Прибавьте к нему единицу, если порядковые номера выбираются произвольно, или запишите в это поле текущую дату, если в системе используется такое соглашение[2].

**Этап 8.** Отредактируйте запись, соответствующую компьютеру-прототипу, чтобы она приняла следующий вид.

```
100     IN   PTR  templatehost.example.com.
```

Скопируйте измененную запись. Обратите внимание на последнюю точку после имени компьютера; не забудьте о ней.

Если в файле зоны обратного преобразования приводится не только последний байт IP-адреса узла, то числа нужно вводить в обратном порядке. Например, запись

```
100.188     IN   PTR  templatehost.my.domain.
```

соответствует IP-адресу 208.77.188.100 (здесь зона обратного преобразования относится к домену 77.208.in-addr.arpa, а не 188.77.208.in-addr.arpa).

**Этап 9.** Обновите порядковый номер в записи SOA в зоне обратного преобразования, как описано на этапе 7.

[2] Соглашение о датах предусматривает двузначное количество изменений, например в течение дня вы можете сделать до 99 изменений.

**Этап 10а.** Если вы работаете с сервером BIND и ленитесь, то выполните команду **ndc reload**. Если сервер занят, вы можете перезагрузить только домены (или представления), которые вы изменили.

```
$ sudo rndc reload зона_прямого_преобразования
$ sudo rndc reload зона_обратного_преобразования
```

**Этап 10б.** Если вы используете сервер NSD, выполните команду **nsdc reload**, а затем **nsdc restart**.

**Этап 11.** Проверьте конфигурацию с помощью команды **dig** (описана в разделе 17.15). Попробуйте также выполнить команды **ping** или **traceroute**, задав имя нового компьютера, даже если он еще не сконфигурирован. Сообщение "host unknown" (неизвестный узел) свидетельствует о наличии ошибки. Сообщение "host not responding" (узел не отвечает) означает, что все в порядке.

Чаще всего системные администраторы совершают ошибки, забыв о следующем:

- обновить порядковый номер (этапы 7 и 9);
- перезапустить сервер имен (этап 10);
- добавить точку в конец доменного имени в записи PTR в зоне обратного преобразования (этап 8).

## Настройка конфигурации клиента DNS

У каждого компьютера, подключенного к сети, должен быть клиент, представляющий собой сервер имен. Конфигурация клиентской части системы DNS задается в файле **/etc/resolv.conf**, в котором перечисляются DNS-серверы, которым можно посылать запросы, когда пользователь пытается преобразовать имя компьютера в IP-адрес (например, запрашивает веб-страницу, посылает сообщение по электронной почте или использует Интернет).[3]

Если ваш компьютер получает свой IP-адрес и сетевые параметры от DHCP-сервера, то файл **/etc/resolv.conf** должен заполняться автоматически. В противном случае его нужно редактировать вручную. Формат записей файла имеет следующий вид.

```
search имя_домена ...
option имя_опции
nameserver IP-адрес
```

Может быть указано от одного до трех серверов имен. Рассмотрим пример.

```
search atrust.com booklab.atrust.com
nameserver 63.173.189.1      ; ns1
nameserver 174.129.219.225   ; ns2
```

Для файла **resolv.conf** никогда не вводилось понятие комментариев. Они поддерживаются лишь в том смысле, что любой нераспознанный элемент файла игнорируется. Комментарии вполне безопасны в конце директивы `nameserver`, поскольку анализатор файла ищет лишь IP-адрес, игнорируя остальную часть строки. Тем не менее, в директиве `search` их лучше не использовать, так как она может содержать несколько аргументов.

В строке `search` приведен список доменов, которые следует опрашивать, если имя компьютера определено не полностью. Например, когда пользователь вводит команду **ssh coraline**, то распознаватель дополняет имя первым доменом в списке (в рассмо-

[3] В системе Windows клиентская часть системы DNS настраивается с помощью панели конфигурации протокола TCP/IP для каждой сетевой платы. Точное содержание этой процедуры зависит от версии системы Windows.

тренном выше примере — `atrust.com`), после чего ищет узел `coraline.atrust.com`. Если это имя не найдено, делается попытка найти узел `coraline.booklab.atrust.com`. Количество доменов, которые можно задать в директиве `search`, зависит от специфики распознавателя; большинство из них допускает от шести до восьми доменов, при этом предельное количество символов равно 256.

Серверы имен, указанные в файле **`resolv.conf`**, должны разрешать вашему компьютеру посылать запросы и обязаны давать на них полные ответы (т.е. быть рекурсивными), а не отсылать вас на другие серверы имен. Серверы имен опрашиваются по порядку. Если первый сервер отвечает на запрос, другие серверы игнорируются. Если возникает проблема, то по истечении времени, отведенного для ответа на запрос, он переадресуется следующему серверу. Все серверы опрашиваются по очереди, до четырех раз каждый. С каждой неудачей тайм-аут увеличивается. По умолчанию интервал тайм-аута задается равным пяти секундам, что для нетерпеливых пользователей кажется вечностью.

Инструкция `options` может изменять интервал тайм-аута, количество попыток и поведение по умолчанию при выборе перечисленных серверов имен. Доступные опции этой инструкции зависят от реализации библиотеки распознавателя. Ниже описан пакет **`libbind`** для системы ISC. Эта информация относится ко всем системам, кроме HP-UX. Чаще всего имя ближайшего сервера указывается первым в строке `nameservers`, но если вы хотите сбалансировать загрузку между одинаково компетентными серверами, следует использовать опцию `rotation`. Например, инструкция

```
options rotate timeout:2 attepmts:2
```

обращается к перечисленным серверам имен по очереди, ожидает ответа две секунды и запрашивает каждый сервер не более двух раз.

Система HP-UX не полностью поддерживает инструкцию `options`, описанную выше. В этой системе продолжительность тайм-аута и количество попыток запроса задаются непосредственно в файле **`resolv.conf`**.

```
retrans продолжительность_тайм-аута_в_миллисекундах
retry количество_попыток
```

Большинство распознавателей позволяют включать в список не более трех серверов имен. Если указать больше, то "лишние" серверы будут по умолчанию проигнорированы. Правила, установленные по умолчанию, приведены в табл. 17.2.

**Таблица 17.2. Правила, установленные по умолчанию для файла /etc/resolv.conf**

| Система | Максимальное кол-во серверов имен | Максимальная длина поисковой строки | Тайм-аут, с. | Попытки |
|---|---|---|---|---|
| Linuxs | 3 | 6 доменов, 256 символов | 5 | 2 |
| Solaris | 3 | 6 доменов, 256 символов | 5 | 2 |
| HP-UX | 3 | 6 доменов, 256 символов | 5 | 4 |
| AIX | 3 | 6 доменов, 1024 символа | 5 | 2 |

Если компьютер сам является сервером имен, то он должен быть указан первым в собственном файле **`resolv.conf`**. Если в файле не указан ни один сервер имен, то компьютер считается рабочей станцией.

Файл **`resolv.conf`** также распознает директиву `domain`, представляющую собой альтернативу директиве `search`; она задает единственный домен, который должен добавляться к не полностью квалифицированным именам. Это устаревшая форма; мы

рекомендуем заменить директиву `domain` директивой `search` везде, где она встречается. Эти директивы взаимно исключают друг друга, поэтому использовать можно только одну из них. Если вы невольно использовали обе эти директивы одновременно, то выполняться будет только последняя.


Если файл **/etc/resolv.conf** отсутствует, то для того, чтобы решить, как распознать имя и извлечь из имени компьютера, которое должно быть полностью квалифицированным, имя домена, заданное по умолчанию, система AIX использует файл **/etc/netsvc.conf**. Система AIX хранит образец файла **resolv.conf** в виде файла **/usr/lpp/tcpip/samples/resolv.conf**, который можно скопировать. Поскольку добавлять множество строк в текстовый файл довольно неудобно, в системе AIX предусмотрена команда **namerslv**, обеспечивающая интерфейс для добавления, удаления и изменения серверов имен в файле **resolv.conf**. Но это еще не все! Вы также можете использовать команды **mknamsv**, **rmnamsv**, **chnamsv** и **lsnamsv** в качестве интерфейсов высокого уровня для команды **namerslv**. (Разумеется, системный администратор не должен разбираться во всех сложных интерфейсах, поэтому вам, вероятно, понадобится графический пользовательский интерфейс; попробуйте выполнить команду **smitty resolv.conf**.)

После того как файл **/etc/resolv.conf** будет сконфигурирован, система станет использовать в качестве сервера имен сервер DNS, поскольку этот сервер не был отключен в файле, назначающем приоритеты источникам административных данных (**/etc/nsswitch.conf** или **/etc/netsvc.conf** в системе AIX; см. раздел 19.5).

После конфигурирования файла **/etc/resolv.conf** вы должны иметь возможность ссылаться на другие машины по именам, а не по IP-адресам. Попробуйте выполнить команду **ping** *имя_компьютера*. Если вы пытаетесь найти другой локальный компьютер и команда просто зависает, попытайтесь сослаться на его IP-адрес. Если этот прием сработал, значит, в вашей конфигурации DNS есть проблема. Проверьте, правильно ли записаны IP-адреса серверов имен в файле **/etc/resolv** и позволяют ли серверы, на которые ссылаетесь, посылать запросы из вашей сети (см. раздел 17.9). Ответить на эти вопросы позволит утилита **dig**, выполненная на работающем компьютере.

## 17.4. Серверы имен

Сервер имен выполняет несколько рутинных операций.

- Отвечает на запросы об именах и IP-адресах компьютеров.
- Посылает запросы локальным и удаленным компьютерам от имени своих пользователей.
- Кеширует ответы на запросы, для того чтобы ускорить ответы на них в следующий раз.
- Пересылает данные между серверами имен, чтобы обеспечить их синхронизацию.

Серверы имен работают с *зонами*, которые представляют собой домен без поддоменов. Часто термин "домен" употребляется как синоним термина "зона", даже в этой книге.

Серверы NSD/Unbound разделяют функции, обеспечивающие ответы на запросы о ваших узлах (компетенция сервера NSD), и функции, связанные с посылкой запросов о других доменах от имени ваших пользователей (компетенция сервера Unbound). Это разумно.

Серверы имен работают в нескольких разных режимах, поэтому дать их точное определение нелегко. Ситуация усложняется еще и тем, что один и тот же сервер может выполнять неодинаковые функции в разных зонах. В табл. 17.3 перечислены прилагательные, употребляемые при описании серверов имен.

**Таблица 17.3. Классификация серверов имен**

| Тип сервера | Описание |
|---|---|
| авторитетный (authoritative) | Официальный представитель зоны |
| главный (master) | Основное хранилище данных зоны; берет информацию из дискового файла |
| подчиненный (slave) | Копирует данные с главного сервера |
| усеченный (stub) | Напоминает подчиненный сервер, но копирует данные только о серверах имен (записи NS) |
| внутренний (distribution) | Сервер, доступный только в пределах домена (также называется невидимым сервером) |
| неавторитетный [a] (nonauthoritative) | Отвечает на запросы, пользуясь данными из кеша; не "знает", являются ли эти данные корректными |
| кеширующий (caching) | Кеширует данные, полученные от предыдущих запросов; обычно не имеет локальных зон |
| переадресующий (forwarder) | Выполняет запросы от имени многих клиентов; формирует большой кеш |
| рекурсивный (recursive) | Осуществляет запросы от имени клиента до тех пор, пока не будет найден ответ |
| нерекурсивный (nonrecursive) | Отсылает клиента к другому серверу, если не может получить ответ на запрос |

[a] Строго говоря, атрибут "неавторитетный" относится к ответу на DNS-запрос, а не к самому серверу.

Серверы имен систематизируются на основании источника данных (авторитетный, кеширующий, главный, подчиненный), типа хранимых данных (усеченный), пути распространения запроса (переадресующий), типа выдаваемого ответа (рекурсивный, нерекурсивный) и, наконец, доступности сервера (внутренний). В следующих подразделах конкретизируются наиболее важные из этих различий; об остальных различиях пойдет речь в других разделах главы.

## Авторитетные и кеширующие серверы

Главный, подчиненный и кеширующий серверы различаются только двумя характеристиками: откуда поступают данные и авторитетен ли сервер для домена. Сервер BIND может относиться ко всем трем типам, сервер NCD — только к главному или подчиненному, а сервер Unbound — только к кеширующему.

В каждой зоне есть один главный сервер имен.[4] На нем хранится официальная копия данных зоны (в файле на диске). Системный администратор модифицирует информацию, касающуюся зоны, редактируя файлы главного сервера.

Подчиненный сервер копирует свои данные с главного сервера посредством операции, называемой *передачей зоны* (zone transfer). В зоне должен быть как минимум один подчиненный сервер. Усеченный сервер — особый вид подчиненного сервера, загружающий с главного сервера только записи NS (описания серверов имен). О том, зачем соз-

[4] Некоторые организации используют несколько главных серверов или вообще обходятся без них; мы описываем вариант, в котором существует один главный сервер.

дается такой сервер, пойдет речь в разделе 17.9. Один и тот же компьютер может быть главным сервером для одних зон и подчиненным — для других.

🕮 О передаче зоны рассказывается в разделе 17.12.

Кеширующий сервер имен загружает адреса серверов корневого домена из конфигурационного файла и накапливает остальные данные, кешируя ответы на выдаваемые им запросы. Собственных данных у кеширующего сервера нет, и он не является авторитетным ни для одной зоны (за исключением, возможно, зоны локального компьютера).

Гарантируется[5], что авторитетный ответ сервера имен является точным; неавторитетный ответ может быть устаревшим. Тем не менее очень многие неавторитетные ответы оказываются совершенно корректными. Главные и подчиненные серверы авторитетны для своих зон, но не для кешированной информации о других доменах. По правде говоря, даже авторитетные ответы могут быть неточными, если системный администратор изменил данные главного сервера (например, обновил порядковый номер зоны) и забыл передать их подчиненным серверам.

Серверы имен должны находиться на надежных и безопасных компьютерах, имеющих небольшое количество пользователей и подключенных к источнику бесперебойного питания. Требуется наличие хотя бы одного подчиненного сервера. В идеале подчиненных серверов имен должно быть минимум два, причем один из них — вне организации. Подчиненные серверы на местах должны быть включены в разные сети и в разные цепи электроснабжения. Если служба имен вдруг перестанет функционировать, пользователи не смогут нормально работать в сети.

Кеширующие серверы неавторитетны, зато позволяют сократить объем DNS-трафика во внутренней сети и уменьшить время, затрачиваемое на выполнение DNS-запросов. В большинстве организаций DNS-запросы от настольных компьютеров обычно проходят через кеширующий сервер. Более крупные организации должны использовать несколько кеширующих серверов.

С точки зрения безопасности и общей надежности системы DNS желательно разделить функции, связанные с обслуживанием ваших авторитетных данных.

## Рекурсивные и нерекурсивные серверы

Серверы имен бывают рекурсивными и нерекурсивными. Если у нерекурсивного сервера есть адрес, оставшийся в кеше от одного из предыдущих запросов, или он является авторитетным для домена, к которому относится запрашиваемое имя, то он даст соответствующий ответ. В противном случае он вернет отсылку на авторитетные серверы другого домена, которые с большей вероятностью ответят на запрос. Клиенты нерекурсивного сервера должны быть готовы принимать отсылки и обрабатывать их.

У нерекурсивных серверов обычно есть веские причины не выполнять дополнительную работу. Например, все корневые серверы и серверы доменов верхнего уровня являются нерекурсивными, поскольку им приходится обрабатывать десятки тысяч запросов в секунду.

Рекурсивный сервер возвращает только реальные ответы или сообщения об ошибках. Он сам отслеживает отсылки, освобождая от этой обязанности клиента. Базовая процедура анализа запроса, по сути, остается неизменной.

Из соображений безопасности, серверы имен организации, доступные извне, всегда должны быть нерекурсивными. Рекурсивные серверы имен, которые видны извне, могут стать объектом для атаки.

[5] Здесь слово "гарантируется" означает, что ответ придет от базы данных, находящейся в памяти авторитетного сервера, а не от кеша случайного неавторитетного сервера.

Библиотечные функции распознавания имен *не понимают* отсылок. Любой локальный сервер имен, указанный в файле **resolv.conf** клиента, должен быть рекурсивным.

При отслеживании отсылок возникает побочный эффект: в кеше сервера имен накапливается информация о промежуточных доменах. При работе в локальной сети от этого обычно только польза, поскольку в последующих операциях поиска, инициируемых компьютерами этой сети, можно будет пользоваться результатами предыдущих запросов. С другой стороны, сервер домена верхнего уровня (такого, как com или edu) не должен хранить информацию, запрашиваемую компьютером, который находится на несколько уровней ниже.

Серверы имен генерируют отсылки на иерархической основе. Если сервер, к примеру, не сможет узнать адрес узла lair.cs.colorado.edu, он выдаст отсылки к серверам доменов cs.colorado.edu, colorado.edu, edu или корневого домена. Отсылка должна включать адреса серверов того домена, на который она указывает, поэтому выбор домена не случаен: сервер должен ссылаться на домен, серверы которого ему известны.

Как правило, возвращается отсылка на наиболее полный из известных доменов. В нашем примере в первую очередь были бы выданы адреса серверов домена cs.colorado.edu, при условии что они известны. Если этот домен неизвестен, но известен домен colorado.edu, были бы выданы адреса его серверов имен, и т.д.

Сервер имен предварительно наполняет свой кеш содержимым файла "подсказок", в котором перечислены серверы корневого домена. Благодаря этому всегда можно сделать отсылку, даже если она гласит: "Спроси у корневого сервера".

## 17.5. Пространство имен DNS

Это пространство имен представляет собой дерево с двумя основными ветвями, соответствующими прямым и обратным преобразованиям. Прямые преобразования ставят в соответствие именам компьютеров их IP-адреса, а обратные преобразования отображают IP-адреса в имена компьютеров. Каждое полностью определенное имя компьютера (например, nubark.atrust.com) — это узел дерева на ветви прямых преобразований, а каждый IP-адрес — это узел дерева на ветви обратных преобразований. Уровни дерева разделяются точками; корнем дерева является точка.

Полностью определенное имя компьютера можно рассматривать как некое обозначение, в котором "самая важная часть" находится справа. Например, имя nubark.atrust.com означает, что компьютер nubark находится в поддереве atrust, а поддерево atrust — в поддереве com. С другой стороны, в IP-адресе "самая важная часть" находится слева. Например, адрес 128.138.243.100 означает, что узел 100 находится в подсети 243, которая является частью сети 128.138.

Для того чтобы система DNS могла работать с данными обоих видов, ветвь IP-адресов в пространстве имен инвертируется, т.е. октеты IP-адресов записываются в обратном порядке. Например, если узел "nubark.atrust.com." имеет адрес 63.173.189.1, то соответствующий узел на ветви прямых преобразований в дереве имеет имя "nubark.atrust.com.", а узел на ветви обратных преобразований имеет имя "1.189.173.63.in-addr.arpa."[6]. Эти имена завершаются точкой, аналогично тому как полные имена файлов начинаются с косой черты.

Полностью определенное доменное имя (FQDN — Fully Qualified Domain Name) — это полный путь к объекту в системе DNS, включая последнюю точку. Например, полно-

[6] Часть имени in-addr.arpa представляет собой фиксированный суффикс.

стью определенное имя узла nubark в домене atrust.com записывается как "nubark.atrust.com.".

Домен (domain) — это поддерево дерева имен DNS. Например, *домен* atrust.com содержит узел atrust.com и все поддомены и дочерние узлы узла atrust.com. В противоположность домену, зона — это домен без поддоменов, которые были делегированы другим серверам имен.

Если домен atrust.com далее разделить на поддомены engineering, marketing и booklab, то он будет содержать четыре зоны: исходную зону atrust.com, а также зоны engineering.atrust.com, marketing.atrust.com и booklab.atrust.com. Зона atrust.com содержит все узлы, находящиеся в домене atrust.com, за исключением узлов, относящихся к зонам engineering.atrust.com, marketing.atrust.com и booklab.atrust.com.

Имена серверов ассоциируются с зонами, а не доменами. Проверив систему DNS, пользователь может выяснить, что заданное имя (например, booklab.atrust.com) идентифицирует поддомен, а не узел. Поддомены имеют записи серверов имен (NS), связанные с ними.

Изначально имена доменов состояли из букв, чисел и косых черт, причем каждый компонент (метка) не мог содержать более 63 символов, а полностью определенное доменное имя (FQDN) — более 256 символов. Имена FQDN не чувствительны к регистру, но обычно записываются с помощью строчных букв. Ограничения на доменные имена были ослаблены в документе RFC2181.

Продолжающаяся интернационализация доменных имен оказывает влияние на эти правила. В результате ограничения на длину полностью определенных доменных имен были ослаблены. Кроме того, допускается использование символов, не принадлежащих латинскому алфавиту и представленных с помощью метода кодирования Punycode, похожего на стандарт Unicode, но все же отличающегося от него деталями реализации.

Существует два типа доменов верхнего уровня: национальные домены верхнего уровня (ccTLD — country code top-level domain) и общие домены верхнего уровня (gTLD — generic top level domain). Организация ICANN (International Corporation for Assigned Names and Numbers) управляет проектом регистрации имен в доменах gTLD, таких как com, net и org, с помощью аккредитованных агентств. На момент написания книги у вас есть выбор из 1 000 регистраторов и 21 домен gTLD, в которых можно зарегистрироваться. Точную информацию можно найти на сайте icann.org. В настоящее время корпорация ICANN работает над созданием большого количества новых общих доменов верхнего уровня.

Для регистрации имени в домене ccTLD, зайдите на веб-страницу организации IANA (Internet Assigned Numbers Authority) iana.org/cctld и найдите регистр соответствующей страны.

## Регистрация домена второго уровня

Для того чтобы зарегистрировать домен второго уровня, необходимо подать заявку в администрацию соответствующего домена верхнего уровня. Для того чтобы заполнить бланки регистрации, необходимо назначить ответственного технического специалиста и ответственного администратора, а также выбрать хотя бы два компьютера, которые будут серверами домена. Кроме того, необходимо выбрать доменное имя, никем не занятое. Стоимость регистрации зависит от агентства, но в настоящее время она относительно небольшая.

## Создание собственных поддоменов

Процедура создания поддомена аналогична той, что используется при регистрации домена второго уровня, только центральная администрация теперь находится в пределах самой организации. Этот процесс предусматривает следующие этапы.

- Выбор имени, уникального в пределах организации.
- Назначение двух или более компьютеров серверами нового домена.[7]
- Согласование действий с администратором родительского домена.

Прежде чем передавать полномочия, администратор родительского домена должен убедиться в том, что серверы имен дочернего домена сконфигурированы и работают правильно. В противном случае произойдет то, что называется *некорректное делегирование*, и вы получите по электронной почте неприятное сообщение с требованием исправить ошибку (подробнее об этом рассказывается в разделе 17.15).

# 17.6. Разработка собственной среды DNS

На разработку надежной и эффективной системы DNS для своего окружения влияет множество факторов: каков, например, размер вашей организации, используете ли вы частные IP-адреса в своей локальной сети в соответствии с документом RFC1918, протокол DHCP и службу Active Directory, применяете ли вы маршрутизацию или коммутацию своей внутренней сети, где расположен брандмауэр по отношению к вашим DNS-серверам. Полезно разделить эту проблему на три части.

- Управление иерархией пространства имен: поддоменами, многократными уровнями и т.д.
- Поставка авторитетных данных о вашей организации во внешний мир.
- Преобразование имен для пользователей.

## Управление пространством имен

Если ваша организация небольшая и независимая, использование поддоменов не является ни необходимым, ни желательным, если ваше руководство не требует этого по каким-то причинам, не имеющим технической природы. С другой стороны, в организациях среднего размера с несколькими независимыми группами системного администрирования поддомены могут уменьшить потребность во взаимодействии на уровне организации. (Чаще всего поддомены выделяются по географическому или организационному принципу.) В крупной организации трудно обеспечивать уникальность имен и поэтому необходимы поддомены, возможно даже многоуровневые.

Недавно в системе DNS были определены зонные записи (SPF и DKIM/ADSP), с помощью которых можно запрещать другим организациям штамповать почтовые сообщения, якобы исходящие из вашего домена. Для оптимального использования этой функциональной возможности, возможно, потребуется определить поддомен, ориентируясь на конфиденциальность информации, отправляемой по электронной почте. Более подробно эта процедура описана в разделе 17.8.

Создание поддоменов требует взаимодействия между системными администраторами, отвечающими за родительский и подчиненный домены. При организации и настройке

[7] С технической точки зрения, поскольку вы сами устанавливаете правила для своего поддомена, может быть один сервер или несколько.

поддомена следует запомнить, с кем вы должны вступить в контакт, если захотите добавить, изменить или удалить серверы. Убедитесь, что ваш брандмауэр не блокирует доступ к серверам поддомена, если вы хотите, чтобы поддомен был доступен для внешнего мира.

Если вы используете поддомены для управления вашим пространством имен, раз в неделю запускайте утилиту **doc** (domain obscenity control — контроль корректности домена) с помощью программы **cron**, чтобы убедиться, что делегирование остается синхронным и вы случайно не выполнили неправильного делегирования. Описание утилиты **doc** и некоторых других инструментов, поддерживающих работоспособность системы DNS, изложено в разделе 17.13.

## Авторитетные серверы

Спецификации системы DNS требуют, чтобы в каждом домене было, по крайней мере, два авторитетных сервера. Главный и подчиненный серверы являются авторитетными, а кеширующие серверы и заглушки — нет. В идеале организация имеет несколько авторитетных серверов, по одному на каждую отдельную сеть и цепь электропитания. Многие организации поддерживают работу внешних авторитетных серверов, которые часто располагаются у их интернет-провайдеров. Если ваш провайдер не предоставляет таких услуг, вы можете воспользоваться ими у провайдера DNS-службы или договориться с местной фирмой (желательно не с конкурентом) или университетом.

Несколько лет назад компания Microsoft попалась на нарушении правила, касающегося разделения сетей. Все три их авторитетных сервера находились в одной и той же подсети, и когда маршрутизатор, соединяющий эту подсеть с Интернетом, дал сбой, серверы стали недоступны. Через два часа, по истечении срока действия записей в кеше, сайт `microsoft.com` и все его другие домены отключились от Интернета. Количество запросов к именам, связанным с компанией Microsoft, на корневых серверах достигло 25% общей нагрузки (10 тыс. запросов в секунду), хотя эта величина обычно равнялась 0,000001%. Решение проблемы заняло несколько дней. Когда страсти улеглись, компания Microsoft исправила маршрутизатор и передала свою службу DNS внешней организации.

Авторитетные серверы поддерживают синхронизацию данных с помощью зонных передач (zone transfers). Для аутентификации и управления переносом зон от главного сервера к подчиненным следует использовать ключи протокола TSIG (transaction signature — подпись транзакции). Описание конфигурации протокола TSIG приводится в разделе 17.13.

Иногда ответы, выдаваемые вашими авторитетными серверами по запросу, зависят от того, кто посылал запрос. Запрос, поступивший из внешнего мира, может иметь один ответ, а тот же самый запрос, поступивший изнутри организации, может иметь другой (более полный) ответ. Эта конфигурация называется "разделением DNS" ("split DNS") и осуществляется на зонном уровне, а не на уровне сервера имен.

Каждая версия зоны называется "представлением", по аналогии с инструкцией `view`, с помощью которой задается ее конфигурация в конфигурационном файле сервера BIND. Для внешних пользователей данные имеют одно представление, а для внутренних — другое. Эта функциональная возможность широко используется для маскировки внутренних машин от шпионов и для гарантии того, что машины, использующие частные IP-адреса в соответствии с документом RFC1918, не раскрывают их в Интернете. Отладка представления представляет собой довольно сложную процедуру, но широкие возможности регистрации, существующие в сервере BIND, в сочетании с правильным использованием команды **dig** могут помочь. Некоторые подсказки приведены в разделе 17.15.

Сервер NSD не поддерживает представления и разделение DNS. Однако вы можете имитировать эту функциональную возможность, запустив два экземпляра сервера NSD с разными конфигурациями. (Разумеется, вы можете это сделать и с сервером BIND.)

## Кеширующие серверы

Рекурсивные кеширующие серверы отвечают на запросы локальных пользователей относительно сайтов, расположенных в Интернете. Каждый компьютер на вашем сайте должен иметь доступ к локальному кеширующему серверу.

Некоторые организации используют иерархию, в которой одна или несколько машин играют роль механизма продвижения данных, через который локальные кеширующие серверы подсети передают свои запросы. В связи с этим такие машины создают наполненный кеш, доступный для всего сайта. В зависимости от размера сайта, эти машины могут быть независимыми или образовывать иерархию. Конфигурация механизмов продвижения данных в реализации BIND описана в разделе 17.9, а в реализации Unbound — в разделе 17.11.

Если кеширующий сервер "падает", то работа всех сетевых пользователей, являющихся основными клиентами этого сервера, практически останавливается.[8] (И ваш телефон начинает разрываться от звонков.) Загрузите ваши кеширующие серверы имен по сценарию, в котором через несколько секунд после сбоя они перезагружаются. Рассмотрим пример сценария для машины, применяющей программу **named** к нескольким доменам верхнего уровня (TLD).

```
#!/bin/sh

PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH
export PATH

trap "" 1
while :; do
     named -f -c /var/named/named.conf >> /var/log/named 2>&1
         < /dev/null
     logger "named restart"
     sleep 15
done
exit
```

Когда выполнение программы **named** завершается крахом, этот сценарий выводит на экран запись в системном журнале с командой **logger**, ждет 15 секунд (эта величина задается произвольно), а затем повторно запускает программу **named**. В инсталляционном пакете сервера BIND этот сценарий записан в каталоге **contrib**, хотя это не обязательно.


В системе Solaris существует служба защиты SMF(см. раздел 3.6).

## Требования к аппаратному обеспечению

Серверы имен должны быть хорошо оснащенными по трем направлениям: центральный процессор, память и пропускная способность сети. Среди них центральный процессор, вероятно, в настоящее время является наименее важным фактором, но в будущем

[8] Если в клиентском файле ***/etc/resolv/conf*** перечислено несколько серверов имен, то механизм разрешения должен переключиться на один из резервных серверов. Однако слишком часто в этом файле указывается только один сервер имен.

после полного развертывания набора спецификаций DNSSEC потребуется поддерживать возможность подписи зон и проверки их подлинности. По возможности используйте специализированные машины в качестве загруженных серверов имен и отделяйте авторитетные серверы от рекурсивных.

Загруженные серверы имен получают тысячи запросов в секунду, и, следовательно, им требуются несколько сетевых интерфейсов и сетевые соединения с высокой пропускной способностью. Трафик обычно состоит из огромного количества маленьких пакетов UDP.

Рекурсивным серверам требуется достаточный объем памяти для кеширования всех ответов, которых ожидают пользователи. Для того чтобы определить, имеет ли сервер имен достаточный объем памяти, лучше всего запустить его и проследить за размером процесса сервера имен. В течение одной-двух недель эта величина сводится к стабильному размеру, при котором старые записи кеша удаляются с той же скоростью, с которой вставляются новые. В стабильном состоянии система не должна выполнять свопинг, а скорость подкачки страниц должна быть разумной.

Если сервер имен функционирует на специализированном компьютере, то объем памяти должен быть вдвое больше, чем потребляет демон сервера имен за неделю. Объем использованной памяти демонстрируют команды **top** и **vmstat**; более подробная информация приведена в разделе 29.4.

Авторитетным серверам требуется довольно большой объем памяти для хранения всех данных, относительно которых они являются авторитетными. Большинство сайтов может управлять этим процессом, но серверы для доменов верхнего уровня и сайтов DNS-хостинга могут потребовать огромный размер памяти или специальное программное обеспечение, позволяющее хранить данные на диске.

Вы можете управлять объемами ресурсов, используемых сервером имен, с помощью опций конфигурации. Список настроек для сервера BIND приведен в разделе 17.9, а для сервера NSD — в разделе 17.11.

## Безопасность

Безопасности системы DNS посвящен раздел 17.13. Мы не собираемся повторять одно и то же и лишь напомним, что при использовании брандмауэра следует убедиться, что система DNS не эмитирует запросы, ответы на которые блокируются брандмауэром. Проверьте также, что ваши администраторы системы DNS взаимодействуют с администраторами по вопросам безопасности и администрацией сети.

По умолчанию система DNS использует для запросов протокол UDP со случайными непривилегированными портами источников (>1023); ответами являются пакеты UDP, адресованные тем же портам. При использовании набора спецификаций DNSSEC и интернационализированных имен доменов, размеры ответов системы DNS могут превышать емкость пути, и поэтому они фрагментируются. Таким образом, ваш брандмауэр не должен блокировать фрагментированные пакеты UDP. Если запрос UDP завершился неудачей из-за фрагментации, то часто он посылается заново как запрос TCP, поэтому брандмауэр должен также пропускать ответы системы DNS по протоколу TCP.

## Итоги

На рис. 17.2 показана архитектура, описанная в предыдущих абзацах.

На этом рисунке продемонстрировано четкое разделение кеширующих серверов (слева) для пользователей и авторитетных серверов (справа) для данных. Обратите также внимание на подчиненный сервер, находящийся за пределами сайта, что считается очень желательным.

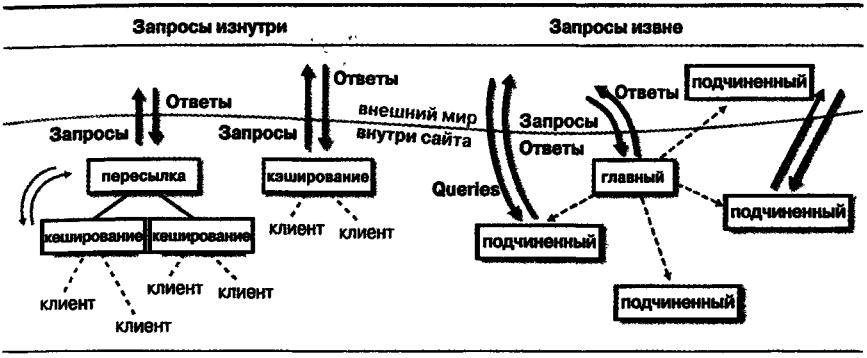



*Рис. 17.2. Архитектура сервера DNS*

🕮 Информация об альтернативной адресации описана в разделе 14.4.

Калифорнийский университет в Беркли (`berkeley.edu`) использует альтернативную IP-адресацию для репликации своих кеширующих серверов. Все клиенты контактируют с одним и тем же набором серверов, а система маршрутизации (в данном случае — OSPF) направляет их на ближайший кеширующий сервер. Эта конфигурация порождает простую и логичную конфигурацию клиента и надежную среду DNS.

# 17.7. Что нового в системе DNS

Одним из самых изящных нововведений в системе DNS является использование записей DNS для аутентификации и проверки целостности сообщений электронной почты. Эта система, получившая название DomainKeys Identified Mail (DKIM), позволяет выявлять фишинг (например, письма, которые якобы приходят из вашего банка и в которых вас просят "подтвердить" информацию о вашем банковском счете). Система DKIM позволяет также распознавать спамеров, подделывающих адреса отправителей.

В системе DKIM сервер, являющийся источником исходящих сообщений электронной почты, подписывает их с помощью закрытого криптографического ключа. Соответствующий открытый ключ публикуется в виде TXT-записи DNS. Получатель сообщения электронной почты может проверить его целостность и происхождение, просмотрев (открытый) ключ DKIM отправителя и сравнив его с подписями самого сообщения.

Система DKIM не требует изменения программного обеспечения системы DNS, но она предполагает взаимодействие сервера, являющегося источником почтовых сообщений (для их подписи), и сервера, получающего эти сообщения (для проверки подписей). С точки зрения системы DNS для поддержки нового поддомена с именем `_domainkey` требуется лишь изменить файлы конфигурации и данных.

Кроме того, правила Author Domain Signing Practice (ADSP) позволяют сайту сообщать, подписывает ли он все, часть или вообще не подписывает исходящие сообщения электронной почты для каждой зоны DNS. Сайты-получатели могут использовать эту политику, чтобы решить, как поступать с неподписанными сообщениями, а также с сообщениями, подпись которых невозможно верифицировать.

Например, банк, генерирующий несколько категорий сообщений электронной почты (например, маркетинговые письма, письма о состоянии счетов и инструкции по электронным платежам), может создавать поддомены для каждой функции и устанавливать для них разные политики. Получатели могут игнорировать отсутствие или несоот-

ветствие подписи оригиналу на рекламных письмах, но отклонять сообщения, которые должны иметь высокий уровень безопасности.

Этот механизм напоминает систему Sender Policy Framework (SPF), определяющую способ, с помощью которого организации публикуют имена своих правильных почтовых серверов в системе DNS, чтобы распознавать спамеров, желающих подделать адрес отправителя, и отклонять их письма.

В списке новшеств мы указываем также сервер BIND 10, представляющий собой следующее поколение программного обеспечения BIND, разрабатываемого консорциумом ISC (Internet System Consortium), поддерживающим сервер BIND, начиная с четвертой версии. Разработка сервера BIND 10 была оплачена спонсорами со всего мира, в основном регистраторами доменов.

Сервер BIND 10 по-прежнему представляет собой открытую реализацию системы DNS. Частично он основывается на версии BIND 9 и основное внимание уделяет повышению модульности, гибкости, интеграции, устойчивости и управляемости во время выполнения.

Сервер BIND 9 и более ранние версии хранили базу данных DNS в памяти; сервер BIND 10 поддерживает работу нескольких систем хранения данных. Еще одна запланированная функциональная возможность сервера — это удобный пользовательский графический интерфейс API. Он облегчит заполнение зон и управление программным обеспечением. Подробная информация об этом приведена на сайте `ics.org/bind10`.

Некоторые изменения, которые были упомянуты в предыдущих изданиях книги, по-прежнему проходят стандартизацию, но еще не приняты. К ним относятся спецификация DNSSEC-bis (безопасность), система IDN (интернациональные имена доменов) и протокол IPv6. Эти инициативы развиваются, но медленно. Мы включили их в последние строки табл. 17.4.

**Таблица 17.4. Новые возможности в DNS и BIND**

| Раздел | RFC | Описание |
|---|---|---|
| 17.9 | 5001 | NSID, идентификация имен серверов для серверов альтернативной адресации |
| 17.8 | 5518, 5016, | Требования DKIM, подписи, подписи третьих лиц |
| 17.8 | 4871, 4686 | Практика подписания сообщений отправителями ADSP |
| 17.8 | 4470 | Система идентификации почтовых серверов SPF |
| 17.8 | 4255 | Слепки открытых ключей SSHFP, SSH |
| 17.8 | 5198, 4952, 4690, 4290, 4185, 3492 | Интернациональные доменные имена (в кодировке Punycode, в доменах верхнего уровня, в формате обмена) |
| 14.10 | 4472, 4339, 4159, 3901 | Протокол IPv6, оперативные вопросы, конфигурация узлов, использование доменной зоны ip6.arpa, а не ip6.int для обратного преобразования, текущие наилучшие практики |
| 17.13 | 5155, 5011, 4641, 4509, 4470, 4033-5 | Спецификация DNSSEC, аутентификация, записи о делегировании домена во время подписи сообщения (DS), практики эксплуатации, "точки доверия" (trust anchors), отказ в существовании (NXDOMAIN) |

Некоторые из этих предложений представляют собой огромные проекты, стандартизацию которых организация IETF еще не закончила. Рабочие группы, которые пишут стандарты, состоят из хороших специалистов, но недостаток "бдительных бойцов" приводит к тому, что некоторые спецификации трудно или даже невозможно реализовать. Текущие выпуски серверов BIND, NSD и Unbound содержат большинство из этих новшеств.

🕮 Протокол IPv6 более подробно описан в главе 14.

Комментариев заслуживают две новые функциональные возможности: поддержка протокола IPv6 и спецификация DNSSEC. Протокол IPv6 увеличивает длину IP-адресов с 32 до 128 бит. Если он когда-нибудь будет полностью реализован, то окажет огромное влияние на Интернет. Серверы BIND, NSD и Unbound частично поддерживают протокол IPv6, который уже стандартизован, но вряд ли будет широко развернут в ближайшее время. По этой причине мы лишь кратко описываем протокол IPv6. В этой главе достаточно дать общее представление об этом протоколе, но этого недостаточно для того, чтобы вы могли перевести ваш сайт на протокол IPv6 и настроить систему DNS на работу с ним.

Стандарт DNSSEC добавляет в базу данных DNS и ее серверов данные об аутентификации. Он использует открытый криптографический ключ для верификации источника и целостности данных DNS, а также заставляет систему DNS распространять ключи наряду с данными об узле.

Организации, желающие развернуть зоны, подписанные в соответствии со спецификациями DNSSEC, столкнутся с проблемой самонастройки, пока корневой домен и домены верхнего уровня не будут подписаны, потому что модель доверия DNSSEC требует, чтобы подписи образовывали цепочку, начинающуюся в корневом домене. Однако новая временная схема DLV (domain lookaside validation — опережающая проверка домена) позволяет находить и соединять друг с другом "островки доверия", пока корневой домен и домены gTLD не реализуют спецификации DNSSEC полностью. Подробности описаны в разделе 17.13.

Описание интернациональных имен доменов, позволяющих использовать неанглийские символы, сводится к способу, которым символы кодировки Unicode отображаются в символы кодировки ASCII. Это однозначное отображение, имеющее обратное отображение, выполняет система Punycode, использующая алгоритм Bootstring. Детали описаны в документе RFC3492. Интернациональные имена доменов существенно сокращают максимальную длину (как покомпонентную, так и общую) имен DNS. Представление имен в кодировке Pubycode начинается со строки xf--, поэтому если вы увидите странный запрос, начинающийся с этих четырех символов, то будете знать, что они означают.

Каждая из этих возможностей (IPv6, DNSSEC и интернационализация) значительно повышает размер записей о данных в системе DNS. В результате система DNS может превысить размеры пакетов UDP и потребовать применения протокола EDNS0 (Extended DNS, версия 0), чтобы увеличить размер пакета с 512 байт (по умолчанию) до более крупной величины, например 4096 байт. В 2009 году статистические данные, накопленные на корневом сервере K, свидетельствовали о том, что примерно 35% запросов не использовали протокол EDNS0 и получали усеченные или фрагментированные ответы от сайтов, использовавших более крупные пакеты.[9]

## 17.8. База данных DNS

Зонные базы данных DNS — это множество текстовых файлов, поддерживаемых системным администратором на главном сервере имен зоны. Эти текстовые файлы часто называются файлами зон. Они содержат записи двух типов: команды синтаксического анализатора (например, `$ORIGIN` и `$TTL`) и записи о ресурсах. Базе данных принадлежат

[9] Текущие данные можно найти на странице `k.root-servers.org/statistics/GLOBAL/monthly`.

только записи о ресурсах, а команды синтаксического анализатора предназначены для того, чтобы упростить ввод записей.

🕮 Команды файла зоны стандартизованы в документах RFC 1035 и 2308.

## Команды в файлах зон

Команды могут быть встроены в файлы зон, чтобы сделать их более читабельными и облегчить их сопровождение. Эти команды либо влияют на способ, с помощью которого синтаксический анализатор интерпретирует последующие записи, либо сами являются сокращением нескольких записей DNS. После того как файл зоны будет прочитан и интерпретирован, ни одна из этих команд не становится частью данных о зоне (по крайней мере, в их исходном виде).

Три команды являются стандартными в системе DNS, а четвертая, `$GENERATE`, относится только к серверу BIND. Пример использования команды `$GENERATE` приведен далее. Стандартные команды выглядят следующим образом.

```
$ORIGIN имя_домена
$INCLUDE имя_файла [источник]
$TTL стандартное_время_существования
```

Директивы должны начинаться в первой колонке и занимать отдельную строку.

Файлы зон читаются и анализируются сверху вниз за один проход. Когда сервер имен читает файл зоны, он добавляет стандартное имя домена ("источник") к любому имени, которое не полностью определено. По умолчанию источником служит домен, указанный в файле конфигурации сервера имен. Однако посредством директивы `$ORIGIN` можно задать или изменить источник в файле зоны.

Использование относительных имен вместо полностью определенных позволяет сэкономить много времени на вводе данных и делает зонные файлы гораздо более удобными для восприятия.

Во многих организациях в зонные файлы включаются директивы `$INCLUDE`, позволяющие разделять зонные базы данных на логические блоки или хранить ключи шифрования в отдельном файле с ограниченными правами доступа. Синтаксис директивы `$INCLUDE` таков.

```
$INCLUDE имя_файла [источник]
```

Указанный файл включается в базу данных в том месте, где стоит эта директива. Если имя файла не является полностью определенным, оно интерпретируется относительно каталога, из которого был запущен сервер имен.

Если задано значение параметра *источник*, то синтаксический анализатор действует так, будто чтению файла предшествовала директива `$ORIGIN`. Обратите внимание на то, что значение *источник* не возвращается к своему предыдущему значению после выполнения директивы `$INCLUDE`. Возможно, вы захотите вернуться к предыдущему значению либо в конце включенного файла, либо в строке, следующей за директивой `$INCLUDE`.

Директива `$TTL` задает стандартное время существования последующих записей. Она должна стоять в первой строке файла зоны. По умолчанию время жизни измеряется в секундах, но можно задать и другие единицы измерения: часы (`h`), минуты (`m`), дни (`d`) или недели (`w`). Например, все перечисленные ниже директивы

```
$TTL 86400
$TTL 24h
$TTL 1d
```

устанавливают значение `$TTL` равным одному дню.

## Записи о ресурсах

С каждой зоной в иерархии DNS связан набор записей о ресурсах. Базовый формат этой записи имеет следующий вид.

```
[имя] [ttl] [класс] тип данные
```

Поля разделяются знаками табуляции или пробелами и могут содержать специальные символы (табл. 17.5).

**Таблица 17.5. Специальные символы, используемые в записях о ресурсах**

| Символ | Назначение |
|---|---|
| ; | Начало комментария |
| @ | Имя текущей зоны |
| () | Разбивка данных на несколько строк |
| * | Метасимвол [a] (только в поле *имя*) |

[a] Предупреждения, связанные с этим символом, см. ниже.

Поле *имя* идентифицирует объект (обычно узел или домен), к которому относится запись. Если несколько последовательно расположенных записей ссылаются на один и тот же объект, то после первой записи поле *имя* можно опустить. Поле должно начинаться в первой колонке, если она присутствует.

Имя может быть относительным либо абсолютным. Абсолютные имена заканчиваются точкой и полностью определены. На внутреннем уровне программное обеспечение работает с полными именами, добавляя имя текущего домена и точку ко всем именам, которые не заканчиваются точкой. Это позволяет укорачивать имена, но часто сопряжено с ошибками.

Например, в домене `cs.colorado.edu` имя **anchor** интерпретируется как "`anchor.cs.colorado.edu.`". Если же ввести имя `anchor.cs.colorado.edu`, то отсутствие точки в конце также будет подразумевать сокращенное имя, к которому следует добавить стандартный домен, что в итоге даст имя "`anchor.cs.colorado.edu.cs.colorado.edu.`". Это очень распространенная ошибка.

В поле `ttl` (time-to-live — время существования) задается время (в секундах), в течение которого элемент данных может оставаться в кеше и при этом считаться достоверным. Это поле часто опускают, но оно обязательно на корневом сервере в файле подсказок. Стандартное значение поля задается директивой `$TTL`, указываемой в первой строке файла зоны.

🕮 Более подробная информация о службе NIS приведена в главе 19.

Если время существования записей задать равным примерно неделе, то это приведет к значительному снижению сетевого трафика и нагрузки на DNS. Однако следует помнить о том, что после сохранения записи в кеше за пределами локальной сети ее уже нельзя принудительно сделать недостоверной. Поэтому, планируя серьезную реструктуризацию сети, сделайте значение `$TTL` достаточно низким (например, один час), чтобы записи, находящиеся во внешних кешах, быстро устарели, а новые параметры в течение часа вступили в действие. После завершения работы восстановите исходное значение этого параметра.

Некоторые организации устанавливают небольшое время существования своих записей на серверах, имеющих выход в Интернет, для того чтобы в случае возникновения проблем на этих серверах (сбоя сети, отказа аппаратного обеспечения или атаки на

основе отказа в обслуживании) администраторы могли отреагировать, изменив параметры отображения "имя-адрес". Поскольку исходное время существования записей было незначительным, новые значения будут быстро распространены по сети. Например, имя `google.com` имеет параметр TTL, равный пяти минутам, а параметр TTL на серверах имен компании Google равен четырем дням (345 600 с).

```
google.com          300    IN    A   209.85.171.100
google.com       345600    IN   NS   ns1.google.com
ns1.google.com   345600    IN    A   216.239.32.10
```

Для выяснения этих данных мы использовали команду **dig**; для простоты вывод был сокращен.

В поле *класс* задается тип сети. Распознаются следующие три значения.

- IN — Интернет (по умолчанию).
- HS — Hesiod (служба каталогов, локально используемая в некоторых организациях).
- CH — ChaosNet (служжба, используемая на некоторых серверах имен для самоидентификации).

По умолчанию задается класс IN. Он часто явно указывается в файлах данных о зоне, несмотря на то что по умолчанию его можно не задавать. Информационная служба Hesiod, разработанная Массачусетсским технологическим институтом, является надстройкой пакета BIND.

Параметр CH означает ChaosNet — почти исчезнувший сетевой протокол, ранее применявшийся на Lisp-машинах компании Symbolics. В настоящее время в пределах класса CH скрыто только два информационных элемента: номер версии программного обеспечения сервера имен и имя узла, на котором запущен этот сервер. Эти данные можно извлечь с помощью команд **dig** или **drill** (см. раздел 17.9).

Администраторы и хакеры используют номера версий программного обеспечения серверов имен для обновления информации, а администраторы используют имя узла для отладки серверов имен, реплицированных в ходе альтернативной маршрутизации (anycast routing). Раскрытие этой информации посредством класса CH сначала было представлено в пакете BIND, а потом эта возможность была осуществлена во всех реализациях системы DNS.

Существуют различные типы DNS-записей, из которых широко используются менее десяти. В стандарте IPv6 было добавлено еще несколько типов. Записи о ресурсах разбиваются на четыре группы.

- Зонные записи определяют домены и их серверы имен.
- Базовые записи связывают имена с адресами и обеспечивают маршрутизацию электронной почты.[10]
- Аутентификационные записи предоставляют информацию, касающуюся аутентификации и сигнатур.
- Вспомогательные записи содержат дополнительную информацию о компьютерах и доменах.

Содержимое поля *данные* зависит от типа записи. Запрос DNS о конкретном домене и типе записи получает в ответ все соответствующие ему записи о ресурсах, извлеченные из файла зоны (табл. 17.6).

[10] Записи о маршрутизации почты MX относятся как к категории зонных записей, так и к категории базовых записей, поскольку они могут относиться как ко всей зоне, так и к отдельным узлам.

**Таблица 17.6. Записи базы данных DNS**

| | Тип | Имя | Назначение |
|---|---|---|---|
| Зонные | SOA | Start Of Authority | Определение DNS-зоны |
| | NS | Name Server | Определение серверов имен зоны, делегирование полномочий поддоменам |
| Базовые | A | IPv4 Address | Преобразование имени в адрес IPv4 |
| | AAAA | IPv6 Address | Преобразование имени в адрес IPv6; ранее считалась устаревшей, но в настоящее время возрождается |
| | PTR | Pointer | Преобразование адреса в имя |
| | MX | Mail Exchanger | Управляет маршрутизацией почты |
| Безопасность и DNSSEC | DS | Delegation Signer | Хеширует подписанный ключ дочерней зоны |
| | DNSKEY | Public Key | Открытый ключ для имени DNS |
| | NSEC | Next Secure | Используется вместе со спецификацией DNSSEC для генерации отказов |
| | NSEC3 [a] | Next Secure v3 | Используется вместе со спецификацией DNSSEC для генерации отказов |
| | RRSIG | Signature | Множество подписанных аутентифицированных записей о ресурсах |
| | DLV | Lookaside | Некорневая точка доверия для спецификации DNSSEC |
| | SSHFP | SSH Fingerprint | SSH-ключ узла, позволяющий верификацию с помощью DNS |
| | SPF | Sender Policy | Идентифицирует почтовые серверы, запрещает подделки |
| | DKIM | Domain Keys | Проверяет отправителя сообщения электронной почты и целостность сообщения |
| Вспомогательные | CNAME | Canonical Name | Дополнительные имена (псевдонимы) узла |
| | SRV | Services | Местонахождение известных служб в пределах домена |
| | TXT | Text | Комментарии или нестандартная информация [б] |

[a] Оригинальная система NSEC позволяет хакерам использовать команду **dig** для перебора всех зонных записей. В системе NSEC3 этот недостаток устранен за счет усложнения вычислений; в настоящее время используются обе системы.

[б] Записи TXT все шире используются для того, чтобы не ожидать одобрения организацией IETF новых типов записей. Например, записи SPF и DKIM были впервые реализованы именно как записи TXT.

Есть и другие типы записей, которые либо устарели, либо являются экспериментальными, либо не нашли широкого применения. Полный их список приведен в документации. Большинство записей поддерживается вручную (путем редактирования текстовых файлов), но защищенные записи о ресурсах требуют криптографической обработки и поэтому управляются с помощью программного обеспечения. Эти записи описаны в подразделе, посвященном спецификации DNSSEC, в разделе 17.13.

Порядок записей о ресурсах почти произволен, но по традиции запись SOA должна идти первой. Последующие записи могут располагаться в любом порядке, но обычно сразу же после записи SOA следуют записи NS. Записи по каждому узлу, как правило, группируются. Вообще, принято упорядочивать записи по полю *имя*, хотя некоторые организации упорядочивают их по IP-адресам, чтобы было легче идентифицировать неиспользуемые адреса. Зонные файлы на подчиненных серверах не поддерживаются вручную. Они записываются программным обеспечением сервера имен; порядок записей скрыт.

Подробно описывая все типы записей о ресурсах в следующих разделах, мы в качестве примера рассмотрим записи из базы данных домена `atrust.com.`. Поскольку домен по умолчанию — `atrust.com.`, то имя узла **`bark`** в действительности означает `bark.atrust.com.`

🕮 Более подробная информация о документах RFC приведена в разделе 14.1.

Формат и интерпретация каждого типа записей о ресурсах описаны организацией IETF в ряде документов RFC. В следующих разделах укажем конкретные документы RFC, относящиеся к каждой записи (вместе с годом их выпуска), в специальных примечаниях.

## Запись SOA

Запись SOA (Start of Authority — начало полномочий) обозначает начало *зоны* — группы записей о ресурсах, расположенных в одной точке пространства имен DNS. Этот узел дерева DNS называют также точкой передачи полномочий. Как мы увидим ниже, домен DNS обычно охватывает минимум две зоны: для прямого преобразования имен компьютеров в IP-адреса и обратного преобразования. Часть дерева DNS, которая служит целям прямого преобразования, упорядочена по именам, а ветвь обратного преобразования — по IP-адресам.

🕮 Записи SOA определены в документе RFC1035 (1987).

Для каждой зоны создается только одна запись SOA. Запись SOA содержит имя зоны, контактный адрес администрации зоны, порядковый номер и различные параметры обновления данных. Комментарии начинаются с точки с запятой. Рассмотрим следующий пример.

```
; Начало зоны для домена atrust.com

atrust.com     IN  SOA  ns1.atrust.com. hostmaster.atrust.com. (
                          2009070200  ; Порядковый номер
                          10800       ; Refresh  (3 часа)
                          1200        ; Retry    (20 минут)
                          360000      ; Expire   (больше 40 дней)
                          3600)       ; Minimum  (1 час)
```

Поле *имя* записи SOA (в нашем примере `atrust.com.`) часто содержит символ `@`, обозначающий сокращенное имя текущей зоны. Текущее имя задается в инструкции `zone` файла **`named.conf`** или в элементе `name` зонного файла **`nsd.conf`**. Оно может быть изменено в зонном файле посредством директивы синтаксического анализатора `$ORIGIN`.

В показанном фрагменте нет поля *ttl*. Класс зоны — IN (Интернет), тип записи — SOA, а остальные элементы образуют поле *данные*. Числовые параметры в скобках представляют собой продолжительность временных интервалов и часто записываются в одной строке с комментариями.

Имя `ns1.atrust.com.` указывает на главный сервер имен этой зоны.[11]

Имя `hostmaster.cs.colorado.edu.` определяет адрес электронной почты для контактов с администратором домена. Адрес дан в формате "*пользователь.узел.*" (а не *пользователь@узел*). Если необходимо отправить сообщение администратору, просто замените первую точку символом @ и уберите точку, стоящую в конце. Вместо реального регистрационного имени часто используется псевдоним, например **`admin`** или **`hostmaster`**. Дело в том, что на должность администратора может заступить другой человек. В таком случае проще поменять одну запись в файле **`aliases`** (описывается в разделе 20.5), чем выискивать все упоминания старого адреса в зонных файлах.

Круглые скобки позволяют разбить запись SOA на несколько строк.

[11] На самом деле в записи SOA может быть указан любой сервер имен зоны, если вы не используете динамическую систему DNS. В этом случае запись SOA должна содержать имя главного сервера.

Первый числовой параметр — это порядковый номер конфигурации зоны. С его помощью подчиненные серверы определяют, когда следует загружать новые данные. Порядковым номером может быть любое 32-разрядное целое число, причем оно должно увеличиваться при каждом изменении зонного файла. Во многих организациях в этом номере зашифровывается дата последней модификации файла. Например, значение 2009070200 указывает на первое изменение в зонном файле, сделанное 2 июля 2009 года.

Порядковые номера могут не быть последовательными, но должны монотонно возрастать. Если случайно задать на главном сервере очень большое число и передать его подчиненным серверам, то исправить порядковый номер на главном сервере не удастся. Подчиненные серверы запрашивают новые данные только в том случае, когда порядковый номер записи SOA главного сервера больше, чем у них.

Существует два способа решения этой проблемы.

- Можно воспользоваться особенностями последовательного пространства, из которого выбираются порядковые номера. Суть в том, что к имеющемуся номеру добавляется "магическое" число ($2^{31}$), заставляющее подчиненные серверы обновить свои данные, после чего устанавливается требуемый номер. Этот прием описывается в книге Albitz, Paul and Cricket Liu, *DNS and BIND* и в документе RFC1982.
- Более хитроумный и утомительный способ — изменить порядковый номер на главном сервере, удалить процессы на подчиненных серверах вместе с их резервными базами данных, а затем перезапустить серверы. При отсутствии кеша подчиненные серверы повторно загрузят базы данных с главного сервера. Этот метод трудно реализовать, если, следуя полезным советам, вы распределили подчиненные серверы по географическому признаку, особенно если на этих подчиненных серверах нет системных администраторов.

Часто пользователи делают одну и ту же ошибку: меняют файлы данных, забывая при этом исправить порядковый номер. "В наказание" сервер имен откажется распространить внесенные изменения на подчиненные серверы.

Следующие четыре элемента записи SOA — значения интервалов времени (по умолчанию в секундах), определяющих, как долго данные могут находиться в кеше в различных точках глобальной базы данных DNS. Можно поменять единицы измерения, воспользовавшись суффиксами `m` (минуты), `h` (часы), `d` (дни) и `w` (недели). Например, выражение `1h30m` означает "1 час 30 минут". Выбор интервала времени требует компромисса между эффективностью (использование старого значения дешевле выборки нового) и точностью (новые значения более точные). Эти четыре поля называются *`refresh`*, *`update`*, *`expire`* и *`minimum`*.

Первый элемент задает периодичность обновления данных. Он показывает, как часто подчиненные серверы должны связываться с главным сервером и проверять, не поменялся ли порядковый номер конфигурации зоны. Если зонная база данных изменилась (порядковый номер главного сервера стал *больше*, чем у подчиненного сервера), подчиненные серверы должны обновить свои копии базы данных. Общепринятые значения для этого интервала — от одного до шести часов (3600–21600 секунд).

Вместо того чтобы пассивно ждать истечения периода обновления, современные серверы BIND (всегда) и NSD (иногда) самостоятельно уведомляют свои подчиненные серверы об изменениях зон. Уведомление об обновлении может быть потеряно из-за перегруженности сети, поэтому значение параметра *`refresh`* должно быть разумным.

Если подчиненный сервер пытается узнать порядковый номер у главного сервера, а тот не отвечает, то через некоторое время будет сделана повторная попытка. Как показывает опыт, нормальное значение для второго элемента — от 20 до 60 минут (1200–3600 секунд).

Если главный сервер длительное время отключен, то подчиненные серверы будут безуспешно пытаться обновить свои данные. По истечении определенного времени все подчиненные серверы должны "решить", что главный сервер не включится никогда и его данные наверняка устарели. Параметр *expire* определяет, как долго подчиненные серверы будут обслуживать домен в отсутствие главного сервера. Система должна сохранить работоспособность, даже если главный сервер не работает неделю, поэтому третьему элементу следует присваивать большое значение. Мы рекомендуем выбирать интервал от недели до месяца.

Четвертый параметр *minimum* в записи SOA определяет время существования в кеше отрицательных ответов. Время существования положительных ответов (т.е. собственно записей) устанавливается директивой `$TTL` в начале зонного файла. Как свидетельствует практика, значение `$TTL` должно быть от нескольких часов до нескольких дней, а минимальное время существования отрицательных ответов — один-два часа (но не более трех).

Значения параметров `$TTL`, *expire* и *minimum* в конечном итоге вынуждают всех пользователей DNS удалять старые данные. Изначально система доменных имен основывалась на том, что информация об узлах относительно стабильна и меняется нечасто. Однако с появлением протокола DHCP и портативных компьютеров все изменилось. Теперь разработчики серверов имен отчаянно пытаются угнаться за временем, внедряя механизмы инкрементных зонных передач и динамических обновлений, которые будут описаны позднее. Концепция синхронизации была описана в этой главе ранее.

## Записи NS

Записи NS (name server — сервер имен) идентифицируют серверы имен, которые авторитетны для зоны (т.е. все главные и подчиненные серверы), а также делегируют полномочия по управлению поддоменами другим организациям. Обычно эти записи следуют сразу после записи SOA.

🕮 Записи NS определены в документе RFC1035 (1987).

Эти записи имеют следующий формат.

*зона* [*ttl*] [IN] NS *имя_узла*

Рассмотрим пример.

```
          NS    ns1.atrust.com.
          NS    ns2.atrust.com.
booklab   NS    ubuntu.booklab.atrust.com.
          NS    ns1.atrust.com.
```

Первые две строки определяют серверы имен для домена `atrust.com`. Здесь параметр *name* не указан, потому что он совпадает с полем *name* записи SOA, которая предшествует этим записям; поэтому поле *name* оставлено пустым. Параметр *class* также не указан, потому что по умолчанию он равен IN и его не обязательно задавать явно.

Третья и четвертая строки делегируют поддомен с именем `booklab.atrust.com.` серверам имен `ubuntu.booklab` и `ns1`. Эти записи на самом деле являются частью поддомена `booklab`, но они должны также появляться в родительской зоне `atrust.com`, чтобы делегирование было возможным. Аналогично записи NS для домена `atrust.com` хранятся в файле зоны `.com`, чтобы определить поддомен `atrust.com` и идентифицировать его серверы. Серверы домена `.com` отсылают запросы об узлах в домене `atrust.com` серверам, перечисленным в записях NS для домена `atrust.com` внутри домена `.com`.

🕮 Подробнее о делегировании рассказывается далее.

Список серверов имен, расположенных в родительской зоне, должен содержать актуальную информацию, если это возможно. Несуществующие серверы, перечисленные в родительской зоне, замедляют работу службы имен, хотя клиенты в конце концов все равно отправляются на один из функционирующих серверов. Если ни один из серверов, перечисленных в родительской зоне, не существует в дочерней зоне, возникает так называемое некорректное делегирование (см. раздел 17.15).

Дополнительные серверы в дочерней зоне допускаются, поскольку хотя бы один из дочерних серверов имеет запись NS в родительской зоне. Проверьте делегирование с помощью команд **dig** или **drill**, чтобы убедиться, что оно определяет корректный набор серверов (см. раздел 17.15).

## Записи A

Записи A (address — адрес) являются основной частью базы данных DNS. Они обеспечивают перевод имен компьютеров в IP-адреса (ранее эта информация хранилась в файле **/etc/hosts**). Для каждого из сетевых интерфейсов компьютера должна существовать одна запись A. Она имеет следующий формат.

*имя_узла* [ttl] [IN] A *IP-адрес*

Рассмотрим пример.

```
ns1       IN  A   63.173.189.1
```

В этом примере поле *имя_узла* не завершается точкой, поэтому сервер имен добавляет домен, заданный по умолчанию, и образует полностью определенное имя "ns1.atrust.com.". Эта запись связывает данное имя с IP-адресом 63.173.189.1.

## Записи PTR

Записи PTR обеспечивают обратный перевод IP-адресов в доменные имена. Как указывалось ранее, записи обратного отображения существуют в домене in-addr.arpa и именуются байтами IP-адреса, следующими в обратном порядке. Например, зона для подсети 189 в данном примере имеет имя 189.174.63.in-addr.arpa.

Общий формат записи PTR таков.

адрес [*ttl*] IN PTR имя_узла

Например, запись PTR в зоне 189.173.63.in-addr.arpa, соответствующая приведенной выше записи A для узла ns1, будет иметь следующий вид.

```
1         IN  PTR   ns1.atrust.com.
```

Имя 1 не заканчивается точкой и потому является относительным. Вопрос: относительно чего? Не относительно домена atrust.com., потому что, для того чтобы эта запись была точной, домен по умолчанию должен называться 189.173.63.in-addr.arpa.

Этого можно добиться, поместив записи PTR для каждой подсети в отдельный файл. Домен, связанный по умолчанию с этим файлом, задан в файле конфигурации сервера имен. Другой способ выполнить обратное преобразование — включить в файл зоны записи вида

```
1.189     IN  PTR   ns1.atrust.com.
```

с доменом по умолчанию 173.63.in-addr.arpa. В некоторых организациях все записи обратного преобразования помещаются в один файл, а подсеть задается посредством директивы $ORIGIN. Обратите внимание на то, что имя узла ns1.atrust.com должно заканчиваться точкой, иначе к нему будет добавлена строка 173.63.in-addr.arpa.

Поскольку atrust.com и 189.173.63.in-addr.arpa — разные области пространства имен DNS, они составляют две отдельные зоны. У каждой зоны должна быть своя запись SOA и свои записи о ресурсах. Помимо определения зоны in-addr.arpa, для каждой реальной сети нужно также задать зону, которая охватывала бы интерфейс обратной связи (127.0.0.0), по крайней мере при работе с сервером BIND. Пример показан в разделе 17.10.

Описанные привязки прекрасно работают, когда маски подсетей проходят по границе байтов. Но как выполнять обратные преобразования для подсети вида 63.173.189.0/26, где последний байт может находиться в одной из четырех подсетей: 0-63, 64-127, 128-191 и 192-255? В документе RFC2317 описан остроумный прием, основанный на применении записей CNAME; подробнее об этом речь пойдет ниже.

Обратные преобразования, выполняемые записями PTR, используются всеми программами, которые аутентифицируют входящий сетевой трафик. Например, демон **sshd** может допускать[12] удаленную регистрацию без пароля, если исходный узел указан по имени в пользовательском файле **~/.shosts**. Когда узел-адресат получает запрос на установление соединения, он "знает" узел-отправитель только по IP-адресу. Пользуясь услугами DNS, он преобразует IP-адрес в имя, которое сравнивается с содержимым соответствующего файла. Программы **netstat**, **tcpd**, **sendmail**, **sshd**, **syslogd**, **X Windows** и **ftpd** получают имена компьютеров из IP-адресов с помощью механизма обратного преобразования.

Важно, чтобы записи A соответствовали записям PTR. Несовпадение и отсутствие последних приводит к ошибкам аутентификации, в результате чего система замедляет работу. Это само по себе неприятно, но еще хуже то, что появляется почва для атак типа на основе отказа от обслуживания, которые направлены на любое приложение, требующее соответствия записям A при обратном преобразовании.

## Записи MX

Записи MX (mail exchanger — обмен почтой) используются системой электронной почты для более эффективной маршрутизации почтовых сообщений. Запись MX подменяет адресата сообщения, в большинстве случаев направляя сообщение концентратору электронной почты на сервере получателя, а не прямо на его рабочую станцию.

📖 Записи MX определены в документе RFC1035 (1987).

Запись MX имеет следующий формат.

```
имя [ttl] [IN] MX приоритет узел ...
```

Записи, приведенные ниже, направляют почту, предназначенную для получателя user@server.atrust.com, на машину mail.server.atrust.com, если она включена и доступна.

```
somehost  IN  MX  10 piper mailserver.atrust.com.
          IN  MX  20 mail-relay3.atrust.com.
```

Сначала опрашиваются узлы с низким приоритетом: наиболее высокий приоритет — 0, наиболее низкий — 65535. (Может показаться, что конфигурация, описанная в примере, не слишком надежна, поскольку оба почтовых сервера расположены в домене atrust.com. Однако эти два сервера расположены в разных подсетях и находятся в разных местах.)

[12] Но на самом деле, по соображениям безопасности, он этого делать не должен.

Записи MX полезны во многих ситуациях:

- если в системе есть центральный концентратор почты;
- если вы хотите фильтровать почту от спама и вирусов перед ее доставкой;
- если узел-адресат выключен;
- если адресат не подключен к Интернету;
- если администратор локальной сети лучше знает, куда следует рассылать сообщения (т.е. всегда).

Каждый узел, известный во внешнем мире, должен иметь записи MX. Они необходимы другим сущностям в системе DNS. Например, узлы, которые никогда не получали или не должны получать электронную почту (скажем, сетевые принтеры), должны иметь записи MX. Сам домен должен иметь запись MX, указывающую на концентратор почты, так чтобы почта, посылаемая на адрес `user@domain`, приходила туда, куда ее адресовал отправитель. (Однако обратите внимание на то, что в этой конфигурации все машины, существующие в домене, должны иметь уникальные пользовательские имена.)

Компьютер, принимающий почту для другого узла, должен настроить свою программу пересылки соответствующим образом. Как задать такую конфигурацию для программы **`sendmail`** и почтовых серверов Postfix, показано в разделах 20.10 и 20.15 соответственно.

В базе данных DNS иногда можно встретить метазаписи MX.

```
*          IN  MX   10 mailserver.atrust.com.
```

На первый взгляд кажется, что такая запись позволяет избежать многократного ввода данных и является стандартной для всех узлов. Однако метасимвол интерпретируется совсем не так, как можно ожидать. Он соответствует полю имени, которое еще *не* было указано в явном виде в других записях о ресурсах.

Следовательно, нельзя использовать звездочку с целью задания стандартного значения для всех своих компьютеров. С ее помощью можно задать стандартные имена для "чужих" компьютеров. В результате на концентратор будет поступать масса почтовых сообщений, тут же отвергаемых по той причине, что имя узла, обозначенное звездочкой, не принадлежит домену. Поэтому необходимо избегать применения метасимвола в записях MX.

## Записи CNAME

Записи CNAME (canonical name — каноническое имя) позволяют назначать узлу дополнительные мнемонические имена. Псевдонимы широко применяются для закрепления за компьютером какой-либо функции либо просто для сокращения его имени. Реальное имя иногда называют *каноническим*. Приведем несколько примеров.

```
ftp        IN  CNAME   anchor
kb         IN  CNAME   kibblesnbits
```

🕮 Записи CNAME определены в документе RFC1035 (1987).

Запись CNAME имеет следующий формат.

*псевдоним* `[ttl] [IN] CNAME` *имя_узла*

Обнаружив запись CNAME, программное обеспечение системы DNS перестает посылать запросы по мнемоническому имени и переключается на реальное имя. Если у

компьютера есть запись CNAME, то другие записи для узла (A, MX, NS и др.) должны ссылаться на его реальное, а не мнемоническое имя.[13]

Длина цепочки записей CNAME не должна составлять больше восьми элементов. Цепочка — это когда одна запись CNAME ссылается на другую, а та — на третью и т.д. Последним, восьмым, элементом должно быть реальное имя узла. При использовании записей CNAME запись PTR должна ссылаться на реальное имя, а не псевдоним.

Для того чтобы избежать использования записей CNAME, можно опубликовать записи A как для реальных имен, так и для псевдонимов. Эта конфигурация работает немного быстрее, поскольку в ней отсутствует дополнительный уровень косвенной адресации.

## Специальное применение записей CNAME

С помощью записей CNAME можно организовать поддержку зон обратного преобразования для сетей, где маски подсетей не проходят по границе байтов. До того как протокол CIDR получил широкое распространение, такая организация подсетей не применялась или по крайней мере "неправильные" подсети существовали в рамках одной организации, поэтому управлять зонами обратного преобразования было несложно. Например, если сеть `128.138` класса B разделить на группу подсетей класса C, то каждая подсеть займет строго оговоренное место в домене `in-addr.arpa`. В частности, зоной обратного преобразования для подсети `243` будет `243.138.128.in-addr.arpa`.

🕮 Протокол CIDR описан в разделе 14.4.

Но что произойдет, если подсеть `243` разделить еще, допустим, на четыре подсети `/26`? Если все они находятся в пределах одной организации, то это — не проблема: они по-прежнему описываются в одном файле, содержащем все их записи PTR. Однако может случиться так, что подсеть `243` принадлежит интернет-провайдеру, который делегирует подсети `/26` разным клиентам. В этом случае требуется более сложное решение. Провайдер должен либо управлять записями зон обратного преобразования на стороне каждого клиента, либо найти способ разделить третий байт IP-адреса (в рассматриваемом примере — `243`) на четыре разных элемента, каждый из которых делегируется независимо.

Когда административная граница проходит не по границе байтов, приходится быть изворотливым. Нужно также тесно взаимодействовать с администрацией домена, находящегося выше или ниже в иерархии. Трюк заключается в следующем: для каждого возможного адреса узла в зоне `in-addr.arpa` следует добавить запись CNAME, которая переводит операцию поиска в зону, управляемую владельцем соответствующей подсети. Подобная схема приводит к образованию громоздких зонных файлов в родительском домене, но она же позволяет передавать полномочия реальным пользователям каждой подсети.

Рассмотрим описываемый процесс подробнее. Родительская организация (в нашем случае — провайдер) создает для каждого возможного IP-адреса записи CNAME со специальным искусственным компонентом (отделяется точкой), представляющим подсеть. Например, для первого из четырех блоков адресов подсети `/26` дополнительный компонент будет называться "0-63", для второго — "64-127" и т.д. Вот как это выглядит.

```
$ORIGIN 243.138.128.in-addr.arpa.
1           IN  CNAME   1.0-63
2           IN  CNAME   2.0-63
...
```

[13] Это правило, касающееся записей CNAME, явным образом ослаблено в спецификациях DNSSEC, которые добавили цифровые подписи к каждому набору записей о ресурсах DNS.

```
63          IN  CNAME   63.0-63
64          IN  CNAME   64.64-127
65          IN  CNAME   65.65-127
...
```

Для того чтобы передать управление адресами 0-63 зоны обратного преобразования клиенту, за которым закреплена эта подсеть, нужно добавить следующие записи NS.

```
0-63        IN  NS      ns1.customer1.com.
0-63        IN  NS      ns2.customer1.com.
```

На узле `customer1.com` будет находиться зонный файл, содержащий привязки для зоны обратного преобразования `0-63.243.138.128.in-addr.arpa`.

```
1           IN  PTR     host1.customer1.com.
2           IN  PTR     host2.customer1.com.
```

Добавив дополнительный компонент, мы создали новую точку передачи полномочий. Когда, например, кто-то ищет доменное имя, соответствующее адресу `128.138.243.1`, запись CNAME в точке `1.243.138.128.in-addr.arpa` перенаправит поиск в точку `1.0-63.243.138.128.in-addr.arpa`, а этим именем уже управляет реальный клиент.

Клиентские зонные файлы не содержат ничего лишнего: со всеми конфигурационными записями приходится иметь дело провайдеру. Ситуация усложняется, если клиент сам является провайдером и дальше делит свое адресное пространство. Клиент 1 может сам быть интернет-провайдером, желающим разделить свои адреса. Однако непреодолимых препятствий не существует: в BIND поддерживаются цепочки записей CNAME длиной до восьми элементов, а поскольку в байте, как известно, восемь битов, превышения допустимого предела не произойдет. Различные документы RFC не одобряют, но и не запрещают применять такие цепочки. К негативным последствиям можно отнести то, что скорость распознавания имен замедляется, так как распознаватель должен пройти по каждому звену цепочки, послав соответствующее число запросов серверу.

Когда схема стала получать распространение, набор команд сервера BIND пополнился директивой `$GENERATE`, которая упрощает создание записей о ресурсах в родительской зоне.

Например, чтобы сгенерировать все необходимые записи для первой подсети, достаточно следующих строк.

```
$ORIGIN 243.138.128.in-addr.arpa.
$GENERATE 0-63 $ CNAME $.0-63
0-63        IN  NS      ns1.customer1.com.
0-63        IN  NS      ns2.customer1.com.
```

Метасимвол `$` в директиве `$GENERATE` является счетчиком цикла и приводит к созданию 64 различных записей CNAME. Остальные три подсети `/26` обрабатываются аналогично.

## Записи SRV

Эти записи определяют местонахождение служб в пределах домена. Например, благодаря записи SRV можно запросить удаленный домен и узнать имя его FTP-сервера. Раньше в подобной ситуации приходилось действовать наугад в надежде на то, что администратор удаленного домена, следуя традиционной практике, добавил запись CNAME для имени "ftp" в базу данных DNS.

🕮 Записи SRV определены в документе RFC2782 (2000).

Гораздо разумнее применять для этих целей записи SRV. С их помощью администраторам намного удобнее менять адреса служб и контролировать их использование. Однако требуется, чтобы сами клиенты знали, как найти и проанализировать записи SRV, поэтому эффект от их применения пока не столь ощутим.

Записи SRV напоминают обобщенные записи MX с дополнительными полями, которые позволяют администратору DNS управлять внешними соединениями и распределять нагрузку на сервер. Формат записей выглядит следующим образом.

*служба.протокол.имя* [*ttl*] [IN] SRV *приоритет вес порт сервер*

Аргумент *служба* представляет собой имя службы, определенное в базе данных IANA (Internet Assigned Numbers Authority — Агентство по выделению имен и уникальных параметров протоколов Интернета). Получить доступ к этой базе данных можно по адресу `iana.org/numbers.htm`. Аргумент *протокол* должен быть равен `tcp` либо `udp`. Аргумент *имя* — это домен, на который ссылается запись SRV. Аргумент *приоритет* имеет тот же смысл, что и в записи MX. Аргумент *вес* используется для распределения нагрузки между несколькими серверами, *порт* — это номер порта, на котором выполняется служба, а *сервер* — имя сервера, предоставляющего данную услугу. Для того чтобы избежать повторного обращения, в ответ на запрос к записи SRV обычно возвращается запись A сервера.

Если вес равен 0, то специального распределения нагрузки не требуется. Имя сервера, равное '.', означает, что служба на данном узле недоступна.

Ниже показан пример из документа RFC2052, адаптированный для домена `atrust.com`.

```
_ftp.tcp        SRV  0  0  21   ftp-server.atrust.com.

; одна четверть соединений обслуживается старым компьютером,
; а три четверти - новым
_ssh.tcp        SRV  0  1  22   old-slow-box.atrust.com.
                SRV  0  3  22   new-fast-box.atrust.com.

; основной сервер доступен через порт 80,
; а резервный -- через порт 8000 на новом компьютере
_http.tcp       SRV  0  0  80   www-server.atrust.com.
                SRV  10 0  8000 new-fast-box.atrust.com.

; в адресной строке можно указывать как http://www.atrust.com,
; так и http://atrust.com
_http.tcp.www   SRV  0  0  80   www-server.atrust.com.
                SRV  10 0  8000 new-fast-box.atrust.com.

; все остальные службы блокированы
*._tcp          SRV  0  0  0    .
*._udp          SRV  0  0  0    .
```

В этом примере демонстрируется использование аргументов *вес* (служба SSH) и *приоритет* (служба HTTP). Задействованы оба сервера SSH, причем нагрузка между ними распределяется в соответствии с производительностью серверов. Все остальные службы заблокированы, включая службы для протоколов TCP и UDP. Однако тот факт, что эти службы не доступны в системе DNS, не означает, что они на самом деле не выполняются, просто вы не можете найти их с помощью системы DNS.

Серверы MS Exchanges используют записи SRV для того, чтобы помочь клиентам приложения Outlook найти их и автоматически задать конфигурацию функции Outlook Anywhere.

## Записи TXT

Эти записи добавляют в базу данных DNS произвольный текст. Например, в нашей базе данных имеется следующая запись, идентифицирующая организацию.

```
IN          TXT  "Applied Trust Engineering, Boulder, CO, USA"
```

Эта запись стоит непосредственно после записей SOA и NS зоны `atrust.com`, наследуя от них поле *имя*.

🕮 Записи TXT определены в документе RFC1035 (1987).

Запись TXT имеет такой формат.

*имя* `[ttl] [IN] TXT` *информация* `...`

Все информационные элементы должны быть заключены в кавычки. Это может быть как одна, так и несколько строк, каждая из которых взята в кавычки. Будьте внимательны: одна пропущенная закрывающая кавычка может привести к разрушению базы данных DNS, поскольку все последующие записи, вплоть до очередной кавычки, загадочным образом исчезнут.

Как и у других записей о ресурсах, у записей TXT нет внутреннего порядка, поэтому серверы возвращают их в произвольном порядке. Для того чтобы закодировать длинные элементы, такие как адреса, следует использовать длинные строки, а не набор нескольких записей TXT.

Поскольку записи TXT не имеют конкретного формата, их иногда используют для тестирования новых типов записей в системе DNS без изменения самой системы DNS. Например, записи SPF (см. далее) сначала были реализованы как записи TXT. В настоящее время уже созданы специальные типы записей, поэтому использование их вариантов в виде записей TXT не рекомендуется, но многие организации по-прежнему делают это.

## Записи ресурсов IPv6

IPv6 — новая версия протокола IP. Процесс принятия спецификации длится уже более 15 лет, он описан в более чем 250 документах RFC, но до сих пор не завершен.[14] Своим появлением протокол IPv6 обязан острой потребности в дополнительных IP-адресах.

🕮 Записи IPv6 описаны в документе RFC1886 (1995).

Однако решения этой проблемы, считавшиеся временными (протокол CIDR, система NAT и строгий административный контроль над адресным пространством), оказались столь успешными, что массовый переход к стандарту IPv6 вряд ли произойдет в скором будущем. Принятие протокола IPv6 в настоящее время стимулируется Азией, где адреса протокола IPv4 распределены более экономно.

🕮 Стандарт IPv6 подробно описан в главе 14.

Записи IPv6 в системе DNS полностью отделены от протокола транспорта, используемого для их доставки. Публикация записей IPv6 в ваших зонах DNS не означает, что вы должны отвечать на запросы к ним по протоколу IPv6. Около половины запросов к корневому серверу имен K (`k.root-servers.net`) — запросы к записям A протокола IPv4, а одна четверть — это запросы к записям AAAA по протоколу IPv6. Однако 99% всех фактических запросов используют транспорт по протоколу IPv4.

[14] Тони Ли (Toni Li), активный член сообщества IETF, описал протокол IPv6 так: "слишком мало, слишком рано".

### *Прямые записи IPv6 — AAAA*

Формат записи AAAA имеет следующий вид.

*имя_узла* [*ttl*] [IN] AAAA *IP-адрес*

Рассмотрим пример.

```
f.roo-servers.net.   IN  AAAA   2001:500:2f::f
```

Каждый фрагмент адреса, отделенный двоеточием, состоит из четырех шестнадцатеричных цифр (ведущие нули обычно опускают). Если несколько фрагментов включают в себя одни нули, они заменяются двумя двоеточиями. В адресе может быть не более одного такого сокращения.

### *Обратные записи IPv6 — PTR*

В протоколе IPv6 информация обратного преобразования, соответствующая адресу AAAA, представляет собой запись PTR в домене верхнего уровня `in-addr.arpa`, а зоны прямого преобразования — в других ветвях дерева доменов (например, в доменах "com" или "edu").

В "полубайтовом" формате адрес, указанный в записи AAAA, инвертируется путем разложения каждого адресного фрагмента на четыре шестнадцатеричные цифры и записи их в обратном порядке с добавлением суффикса `ip6.arpa`. Например, следующая запись PTR соответствует показанной выше записи AAAA для узла **anchor**.

```
f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.f.2.0.0.0.0.5.0.1.0.0.2.ip6.arpa
PTR f.roo-servers.net
```

(Для того чтобы эта строка поместилась на странице, она была сокращена.) К сожалению, они не очень удобны для работы системных администраторов, которые должны их писать и читать. Разумеется, в ваших реальных зонных файлах DNS задачу можно существенно упростить, воспользовавшись директивой `$ORIGIN`.

## Записи SPF

Записи SPF предназначены для идентификации сообщений электронной почты с подделанными заголовками From, которые часто используются для рассылки спама или фишинга. Если сайт, получивший сообщение, определяет, что его заголовки подделаны, он может удалить его, профильтровать или пометить, перед тем как передать получателю. Эта функциональная возможность впервые была реализована с помощью записей TXT, но в настоящее время уже существует специальный тип записей SPF, использующий тот же самый синтаксис, что и записи TXT. Сайты могут использовать записи SPF, TXT, обе версии или ни одну из них.

🕮 Записи SPF определены в документах RFC4406 и RFC4408 (2006).

К сожалению, существует два конкурирующих способа использования записей SPF: система Sender ID компании Microsoft и альтернативная. Рабочая группа консорциума IETF не достигла консенсуса относительно наилучшего использования записей SPF, поэтому оба подхода были опубликованы в документах RFC в качестве экспериментальных. Основное различие между ними заключается в том, где выполняются проверки адреса отправителя: на конверте или в заголовке.

Обе спецификации записей SPF имеют существенный недостаток: пересылаемые сообщения электронной почты не проходят проверку, потому что получатель сравнивает за-

пись SPF отправителя с IP-адресом пользователя, осуществляющего пересылку. По этой причине следует быть осторожным при удалении сообщений, не прошедших проверку.

Здесь мы описываем часть синтаксиса и семантики записей SPF в соответствии в документом RFC4408. Полная спецификация является чрезмерно гибкой и содержит макросы, а также директивы переадресации и включения, предоставляя пользователям десятки возможностей для формирования собственных правил. Мы сосредоточим внимание на простом и эффективном подмножестве этих возможностей.

Запись SPF содержит IP-адреса серверов, создавших легитимное в пределах зоны сообщение электронной почты. Например, команда **dig gmail.com spf**[15] возвращает следующую запись.

```
gmail.com    300  IN  SPF  "v=spf1 redirect=_spf.google.com"
```

Эта запись SPF перенаправляет клиентов на поддомен `_spf` домена `google.com`. Запрос **dig_spf.google.com spf** возвращает следующий ответ.

```
_spf.google.com    300 IN SPF "v=spf1 ip4:216.239.32.0/19
    ip4:64.233.260.0/19 ip4:66.249.80.0/20 ip4:72.14.192.0/18
    ip4:209.85.128.0/17 ip4:66.102.0.0/20 ip4:74.125.0.0/16
    ip4:64.18.0.0/20 ip4:207.126.144.0/20 ?all"
```

Вместо перечисления конкретных узлов, эта запись SPF содержит IP-номера сетей почтовых серверов компании Google. Эти строки представляют собой одну длинную запись SPF, которая разделена на строки для того, чтобы она поместилась на странице.

Длина строки записи SPF не должна превышать 255 байт, поэтому если в нее необходимо ввести больше 255 символов, то следует использовать несколько пар кавычек. Такие строки конкатенируются без дополнительных пробелов. Желательно, чтобы общая длина строки не превышала 450 байт, чтобы ответы на запросы могли поместиться в отдельном UDP-пакете, размер которого составляет 512 байт.

Рассмотрим нашу запись SPF более подробно.

- Выражение `v=spf1` означает, что запись соответствует версии 1 протокола SPF, описанной в документе RFC4408. Выражение `v=spf2.0` означало бы, что запись соответствует системе Sender ID компании Microsoft, описанной в документе RFC4406.
- Дескрипторы `ip4` означают, что за ним следует обычный IP-номер сети или адрес узла. Запись может содержать несколько фраз с дескрипторами `ip4`, как в данном примере.
- Выражение `?all` означает "сделано" по отношению к функции проверки, интерпретирующей запись.

Сложность языка SPF обескураживает. Остальные дескрипторы позволяют перечислять имена узлов, записи MX, записи PTR, адреса IPv6 и многое другое. Некоторые из них требуют, чтобы сервер DNS выполнил повторный просмотр, поэтому, несмотря на дополнительное удобство и гибкость, они менее эффективны, чем дескриптор `ip4`. Если вы хотите пофантазировать о своих записях SPF, посмотрите на примеры, приведенные в конце документа RFC4408.

Рассмотрим еще один пример записи SPF, полученной при выполнении команды **dig sendmail.com txt**.

[15] Компания Google и другие организации реализуют свои собственные записи SPF на основе записей TXT, потому что тип записей SF является новым и лишь недавно стал поддерживаться программным обеспечением популярных серверов имен. Однако заглядывая в будущее, мы получили образовательную лицензию на программное обеспечение и демонстрируем записи SPF, а не TXT. На самом деле команда ***dig*** работает с записями TXT, а не SPF.

```
sendmail.com      IN TXT "v=spf1 ip4:209.246.26.40 ip4:209.246.26.41
     ip4:63.211.143.38 ip4:209.246.26.36 ip4:209.246.26.39 ip4:209.246.26.24
     ip4:209.246.26.25 ip4:209.246.26.10 ip4:209.246.26.53
     ip4:72.32.154.224/27 ptr:constantcontact.com ~all"
```

Эта запись содержит полные IP-адреса серверов, а не сетей. В конце списка IP-адресов приведена фраза `ptr:`, позволяющая серверу `constantcontact.com` послать почтовое сообщение, отправителем которого подразумевался сервер `sendmail.com`. Эта фраза может вступить в силу, только если сервер `constantcontact.com` имеет соответствующую запись PTR. В настоящее время это условие не выполняется. Во время написания книги запись PTR отправляла нас обратно на сервер `www.constantcontact.com`, а не на сервер `constantcontact.com`. Следовательно, либо получатель сообщений электронной почты не проверял запись PTR достаточно строго, либо запись SPF имеет небольшую ошибку.

Согласно результатам обзорного исследования (`sendmail.org/dkim/survey`), около тысячи банков США и тысяча компаний, входящих в список журнала Fortune, сообщили, что записи SPF поддерживались более чем 90% сайтов, а формат Sender ID — 1-2%. Некоторые почтовые серверы (например, Gmail) выводят на экран ярко-красную полосу поперек сообщения, которое не прошло проверку SPF и может быть элементом фишинга.

Почтовые агенты **`sendmail`**, Postfix и **`exim`** поддерживают обработку записей SPF, а программа Microsoft Exchange поддерживает систему Sender ID. Более подробная информация о записях SPF приведена в разделе 20.5.

## Записи DKIM и ADSP

Аббревиатура DKIM (DomainKeys Identified Mail) представляет собой систему, результатом которой является слияние двух систем: DomainKeys от компании Yahoo! и Identified Internet Mail от компании Cisco, обеспечивающих подпись сообщений электронной почты. Получатель сообщения может аутентифицировать его отправителя (подлинность) и гарантировать его целостность (отсутствие постороннего вмешательства).

🕮 Записи DKIM определены в документах RFC487 (2007) и RFC5617 (2009).

Спецификации DKIM являются результатом большой работы, поэтому, например, списки рассылки и внешние сообщения электронной почты реализуются правильно. Еще одной целью проекта была легкость реализации этой спецификации. Она не требует никаких изменений со стороны пользователей или узлов. Здесь мы рассмотрим лишь те аспекты спецификации DKIM, которые связаны с системой DNS. Особенности спецификации DKIM, связанные с электронной почтой, будут рассмотрены в разделе 20.15.

Записи о ресурсах DKIM пока не стандартизованы так, как это сделано для записей типа DNS. Вместо этого используется особый формат записей TXT. Этот формат состоит из имени записи DNS, сформированной путем конкатенирования "селектора" и строки `_domainkey`. В записи хранится открытый ключ DKIM для организации. Может существовать несколько селекторов, поэтому эти ключи легко продлить и отменить.

Сайт, использующий подписи DKIM, вычисляет их с помощью специальных полей заголовка и тела сообщения, содержащего закрытый ключ DKIM. Он помещает подпись в отправляемое сообщение в виде поля заголовка, называемого DKIM-Signature.

Соответствующий открытый ключ доступен в зоне DNS сайта-отправителя в виде записи TXT, связанной с именем `селектор._domainkey.`*`домен`*. Сайт-получатель поручает системе DNS найти ключ и использует его для проверки корректности подписи

в сообщении. Успешная верификация подписи удостоверяет подлинность сообщения и гарантирует, что оно поступило из предусмотренного домена отправления и по пути не было изменено.

Рассмотрим пример строки заголовка подписи DKIM-Signature из подписанного сообщения.

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=gmail.com; s=gamma;
  h=domainkey-signature:mime-version;received:reply-to:date:message-id:
  subject:from:to:content-type;
  bh=24HfUvt1AO4JxRNBmg94pN6ZJUPdqSbkOdZppou4sI=;
  b=UtYWupx/Udqi7Sd1n0h5zIDKq7R/Gg+HwYBxM0Lcsh1whqrHhyHylea3So8
  EnMXJEYI3jyzj3VNGGemOAOSUqHlMmdlSLdp7AvxptY0VfLgYGM9ID4uw
  B014a7ZJuoiVHJmsEA/ExK48rvq10ZJY+AgRdDpbx6/56phSfVJt+a+A=
```

Разные дескрипторы, используемые в этой подписи, обясняются в табл. 17.7

**Таблица 17.7. Дескрипторы в заголовке подписи сообщений электронной почты DKIM-Signature**

| Раздел | RFC | Описание |
|---|---|---|
| v | 1 | Номер версии; должен быть равным 1 |
| a | rsa-sha256 | Алгоритм шифрования: `rsa-sha1` или `rsa-sha256` |
| c | relaxed/relaxed | Алгоритм каноникализации: [a] `simple` или `relaxed` |
| d | gmail.com | Домен отправителя |
| s | gamma | Селектор или имя ключа |
| h | domain | Поля заголовка для включения в подпись заголовка |
| bh | 24HfUvt1... | Криптографический хеш тела сообщения |
| b | UtYWupx... | Криптографическая подпись всего сообщения |

[a] Этот алгоритм определяет, как изменяются заголовок и тело перед шифрованием.

Дескриптор селектора `s=gamma` сообщает имя открытого ключа, а дескриптор `d=` задает родительский домен. Для того чтобы получить открытый ключ, следует применить команду **`dig`** к записи TXT для псевдо-узла `gamma._domainkey.gmail.com`.

```
gamma._domainkey.gmail.com. 300 IN TXT "k=rsa\; t=y\;
  p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDIhyR3oItOy22ZOaBrI
  Ve9m/iME3RqOJeasANSpg2YTHTYV+Xtp4xwf5gTjCmHQEMOs0qYu0FYiNQP
  QogJ2t0Mfx9zNu06rfRBDjiIU9tpx2T+NGlWZ8qhbiLo5By8apJavLyqTLavyPSrv
  sx0B3YzC63T4Age2CDqZYA+OwSMWQIDAQAB"
```

Это слишком запутанная запись, чтобы самостоятельно записывать ее в свои зонные файлы. К счастью, в нашем распоряжении есть возможность копирования и вставки. Дескриптор `k=` идентифицирует тип ключа; единственное значение этого дескриптора равно `rsa`. Флаг `t=y` означает, что вы уже выполнили проверку по спецификации DKIM и сайты-получатели должны быть снисходительны, если ваша подпись не будет верифицирована. Дескриптор `p=` представляет собой сам открытый ключ. Перед точкой с запятой должна стоять обратная косая черта, поскольку в файлах данных DNA с точки с запятой начинаются комментарии.

Запись TXT в системе DKIM часто содержит дескриптор версии `v=DKIM1`. Как и все записи TXT, она должна быть заключена в двойные кавычки.

Для того чтобы сгенерировать пару ключей RSA в соответствующем формате для ваших зонных файлов, следует использовать следующие команды **openssl**, генерирующие закрытый ключ и извлекающие из него соответствующий открытый ключ.

```
$ openssl genrsa -out rsa.private 1024
$ openssl rsa -in rsa.private -out rsa.public -pubout -outform PEM
```

Затем необходимо вырезать открытый ключ из файла **rsa.public** и вставить его в дескриптор p= вашей текстовой записи. Он не должен содержать ни пробелов, ни символов перехода на новую строку, поэтому при копировании и вставке следует быть осторожным и не добавлять дополнительных символов.

В качестве селектора можно выбрать любое имя.

В приведенном выше примере сегмент `_domainkey.gmail.com` имени `gamma_domainkey.gmail.com` на самом деле не является именем подзоны домена `gmail.com` с точки зрения системы DNS. Это можно подтвердить путем поиска зонных серверов имен (**dig _domainkey.gmail.com ns**), которые должны существовать, если имеется собственная делегированная подзона. Приведенный ниже пример, касающийся сайта `yahoo.com`, реализует `_domainkey` как собственный поддомен.

Документ RFC5617 определяет запись TXT, которую можно спрятать в специальном поддомене, чтобы задать свои собственные правила, касающиеся подписания сообщений. Эта запись недавно была стандартизована (2009) и называется ADSP (Author Domain Signing Policy). Специальным поддоменом является домен `_adsp._domainkey.`*домен*.

Внутри записи TXT содержится выражение `dkim=`, которое объявляет правила подписания сообщений на сайте. Оно может принимать следующие возможные значения.

- `all` — для доменов, подписывающих все исходящие сообщения электронной почты;
- `unknown` — для доменов, которые могут подписывать некоторые сообщения электронной почты;
- `discardable` — для доменов, подписывающих все сообщения электронной почты и рекомендующих получателям игнорировать сообщения, подписи которых не прошли верификацию.

Например, дескриптор `discardable` может использоваться банком, который посылает клиенту конфиденциальную информацию о его счете из поддомена, созданного для этой цели. Если бы пользователь выполнил инструкции, содержащиеся в сообщении с поддельной подписью, это имело бы катастрофические последствия, поэтому лучше всего не принимать такие сообщения и не доставлять их адресату.

Текстовая запись ADSP также может содержать выражение `t=y`, если вы желаете просто протестировать спецификацию DKIM и на самом деле не хотите, чтобы получатели слишком серьезно относились к вашей подписи.

В ходе разработки системы ADSP, до появления документа RFC5617, доменная текстовая запись ADSP хранилась в другом поддомене (`_domainkey.`*domain*, без префикса `_adsp`) и имела немного другой синтаксис. Выражение `o=~` означает, что домен подписывает некоторые сообщения электронной почты, а `o=-` – что он подписывает все сообщения.

Поскольку в этих двух соглашениях используются разные поддомены, они могут сосуществовать. На момент написания книги доминировала исходная форма. Если вы серьезно относитесь к тому, чтобы получатели тщательно рассматривали вашу подпись, то, вероятно, лучше всего использовать оба соглашения на протяжении следующих нескольких лет, пока один из них не будет приведен в соответствие документу RFC5617.

Рассмотрим еще один пример. Сайт Gmail не имеет записи ADSP, а сайт Yahoo! имеет.

```
_domainkey.yahoo.com.    7200  IN  TXT  "t=y\; o=~\;
  n=http://antispam.yahoo.com/domainkeys
```

Выражение n= представляет собой комментарий, который предоставляет пользователю дополнительную информацию о том, как сайт Yahoo! использует записи DKIM.[16] Некоторые сайты вместо этого включают адрес сообщения электронной почты (без символа @). Рассмотрим текстовую запись DKIM для ключа (селектора) s1024.

```
s1024._domainkey.yahoo.com. 86400 IN TXT "k=rsa\; t=y\;
  p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDrEee0Ri4Juz+QfiWYui
  /E9UGSXau/2P8LjnTD8V4Unn+2FAZVGE3kL23bzeoULYv4PeleB3gfm"
  "JiDJOKU3Ns5L4KJAUUHjFwDebt0NP+sBK0VKeTATL2Yr/S3bT/xhy+1xtj4Rkd
V7fVxTn56Lb4udUnwuxK4V5b5PdOKj/+XcwIDAQAB\; n=A 1024 bit key\;"
```

Здесь, как и раньше, выражение n= является комментарием, но на этот раз о самом ключе.

Все записи DKIM, представленные в этом разделе, были записями TXT, но в конце концов они обязательно превратятся в записи специального типа DKIM, хотя и будут иметь тот же самый формат. Тем временем сайты могут использовать оба типа записей, чтобы решить проблемы перехода.

Реализация спецификации DKIM в вашей почтовой системе рассматривается в главе 20.

## Записи о ресурсах SSHFP

Оболочка безопасности SSH позволяет обеспечить безопасность дистанционного входа в систему в незащищенной сети. Эта программа использует две схемы аутентификации: для самого узла и для пользователя, осуществляющего дистанционный вход. К сожалению, пользователи обычно слепо доверяют ключам узла, полученным от программы **ssh**. Запись DNS типа SSHFP позволяет программе **ssh** автоматически проверить ключ узла, гарантируя, что пользователь будет соединен с требуемой машиной, а не с посторонним компьютером.

🕮 Записи SSHFP определены в документе RFC4255 (2006).

Для того чтобы уменьшить размер пакета, записи SSHFP не хранят полные копии открытых ключей узла. Вместо этого они содержат дайджест (т.е. криптографический хеш) этих ключей. Их синтаксис выглядит следующим образом.

```
name [ttl] [IN] SSHFP algorithm# fingerprint_algorithm# fingerprint
```

Параметр *algorithm#* идентифицирует криптосистему открытого ключа, используемую для генерирования ключа данного узла. На самом деле этот алгоритм не принимает участия в процессе верификации; он просто сравнивается с ключом, предоставленным удаленным узлом, чтобы убедиться в том, что обе стороны имеют в виду один и тот же вид ключа. Система RSA — это алгоритм 1, а DSA — алгоритм 2.

Параметр *fingerprint* — это хеш, подлежащий сравнению, а параметр *fingerprint_algorithm#* задает способ, которым обрабатывается ключ, предоставленный удаленным узлом для создания хеша, подлежащего сравнению. В настоящее

[16] Правда, данный URL в настоящее время переадресовывает пользователей на страницу sourceforge.net, посвященную системе DomainKeys, которая представляет собой практически устаревший стандарт.

время определен только один алгоритм (SHA-1), поэтому содержание этого поля всегда равно 1.

Рассмотрим пример, приведенный в документе RFC4255.

```
host.example  SSHFP  2  1  1235456789abcdef67890123456789abcdef67890
```

Программа SSH обычно создает пары ключей узла RSA и DSS; как правило, они хранятся в каталоге **/etc/ssh** с именами вроде **ssh_host_rsa_key.pub** и **ssh_host_dsa_key.pub**. На стороне пользователя программа SSH хранит принятые открытые ключи узла в файле **.ssh/known_hosts** в домашнем каталоге пользователя.

Вы можете извлечь ключи узла из этого каталога, чтобы создать свои собственные записи о ресурсах типа SSHFP. Последние версии программы **ssh-keygen** могут генерировать записи DNS SSHFP с флагами **-r** и **-g**. Кроме того, в системах Linux и UNIX (`freshports.org/dns/sshfp`) существует команда **sshfp**, которая конвертирует ключи в отпечатки и создает необходимые записи DNS.

🕮 Более подробная информация о записях SSHFP приведена в разделе 22.10.

Вы можете попросить программу OpenSSH использовать записи SSHFP для установки опции VerifyHostKeyDNA, равной "yes". Программа SSH поддерживает несколько методов аутентификации и верификации. Поскольку записи SSHFP пока используются не очень широко, их не стоит делать единственно возможным вариантом. Сначала попробуйте использовать запись SSHFP, и если это не получится, позвольте пользователю подтвердить ключ узла вручную, как это делалось до появления записей SSHFP. Более подробно конфигурация программы SSH описывается в разделе 22.10.

Подобно записям SPF и DKIM, система SSHFP в той или иной степени подразумевает, что вы используете спецификации DNSSEC и поэтому записи DNS заслуживают доверия. Возможно, в данный момент это и не соответствует действительности, но спецификации DNSSEC постепенно получают признание и в конце концов будут реализованы. Их описание можно найти в разделе 17.13.

## Записи о ресурсах DNSSEC

В настоящее время со спецификациями DNSSEC ассоциируются шесть типов записей. Записи типов DS, DVL и DNSKEY предназначены для хранения разнообразных ключей и их отпечатков. Записи RRSIG содержат подписи других записей в зоне (по существу, наборов записей). И наконец, записи типов NSEC и NSEC3 позволяют серверам DNS подписывать несуществующие записи, обеспечивая криптографическую безопасность при отрицательных ответах на запросы. Эти шесть записей отличаются от большинства других записей тем, что они генерируются программами, а не набираются вручную.

Спецификации DNSSEC представляют собой большую тему, заслуживающую отдельного изучения, поэтому мы обсудим записи DNSSEC и их использование в разделе 17.13.

## Связующие записи: связи между зонами

Каждая зона имеет свой набор файлов, серверов имен и клиентов. Но зоны должны быть связаны друг с другом, чтобы образовывать иерархии: зона `booklab.atrust.com` — это часть зоны `atrust.com`, и мы должны иметь определенную связь между ними, обеспечиваемую средствами системы DNS.

Поскольку ссылки DNS направлены только от родительских доменов к дочерним, серверу имен не обязательно знать все о доменах (точнее, о зонах), расположенных выше в иерархии. Однако серверы дочернего домена должны знать IP-адреса серверов

имен для всех своих поддоменов. Фактически родительской зоне известны *только* серверы имен, на которые можно сослаться в ответ на внешние запросы.

В терминах системы DNS это значит, что родительская зона должна содержать записи NS для каждой делегированной зоны. Поскольку записи NS содержат имена узлов, а не их IP-адреса, родительский сервер также должен иметь способ преобразования имен узлов либо с помощью обычных запросов DNS (если при этом не возникает замкнутый круг), либо сохраняя копии соответствующих записей A.

Существует два способа, с помощью которых можно удовлетворить эти требования: непосредственно включить нужные записи или использовать зоны-заглушки.

В первом методе вы просто включаете требуемые записи NS и A в родительскую зону. Например, файл зоны `atrust.com` может содержать следующие записи.

```
; информация о поддомене

booklab          IN NS ns1.atrust.com.
                 IN NS ubuntu.booklab.atrust.com.
                 IN NS ns.cs.colorado.edu.
testlab          IN NS ns1.atrust.com.
                 IN NS ns.testlab.atrust.com.

; связующие записи

ubuntu.booklab   IN A 63.173.189.194
ns.testlab       IN A 63.173.189.17
```

"Внешние" записи A называются связующими (glue records), поскольку на самом деле они не принадлежат этой зоне. Они только воспроизводятся здесь, чтобы обеспечить связь с новым доменом в дереве имен. Пропуск или неправильное воспроизведение связующей записи делает ваше пространство имен недоступным, и пользователи, пытающиеся обратиться к ней, получат сообщение об ошибке "host unknown".

Довольно часто пользователи ошибочно включают связующие записи для имен узлов, которые этого не требуют. В приведенном выше примере зона `ns1.atrust.com` является частью зоны `atrust.com`, и ее записи A хранятся где-то в файле. Адрес `ns.cs.colorado.edu` в этом связующем разделе не требуется, поскольку его можно определить с помощью обычного запроса DNS. Запись A в этой зоне, на первый взгляд, не нужна, но позднее она может устареть и стать неправильной, если адрес `ns.cs.colorado.edu` изменится. Рекомендуется включать записи A только для узлов, которые находятся в текущем домене или в любом из его поддоменов. Серверы BIND и NSD игнорируют ненужные связующие записи, причем сервер BIND распознает их наличие как ошибку.

Описанная выше схема представляет собой стандартный способ связывания зон, но он требует, чтобы дочерняя зона поддерживала контакт с родительской и сообщала ей о любых изменениях иди дополнениях, касающихся своих серверов имен. Поскольку родительские и дочерние зоны часто функционируют на разных сайтах, обновление информации может превратиться в утомительное занятие, требующее координации между организациями, разделенными административными границами. Вследствие этого в реальном мире этот вид координации применяется редко.

Второй способ поддержки связей основан на использовании тупиковых зон. Тупиковая зона, по существу, представляет собой подчиненную зону, но она содержит только записи NS и соответствующие записи A ее серверов имен. Аналогично подчиненной зоне тупиковая зона обновляется автоматически и таким образом исключает необходимость обмена информацией между администраторами родительской и дочерних зон.

Следует помнить, что тупиковые зоны должны конфигурироваться идентично главному и подчиненному серверам *родительской зоны*. Возможно, проще всего было бы поддерживать контакт с родительской зоной и проверять ее конфигурацию несколько раз в году (особенно если она является локальной).

Для того чтобы увидеть, какие серверы родительского домена в настоящее время являются доступными, можно использовать команду **dig**. Сначала надо выполнить команду

```
$ dig родительский-домен ns
```

и определить серверы имен в своем родительском домене. Затем следует выбрать один из них и выполнить команду

```
$ dig @имя_сервера.родительский_домен дочерний_домен ns
```

для просмотра списка публичных серверов имен. Тупиковые зоны оказываются очень полезными, когда ваша внутренняя сеть использует пространство частных IP-адресов, определенных в документе RFC1918, и вам необходимо поддерживать синхронное делегирование в соответствии с документом RFC1918.

Мы осветили большинство вопросов, касающихся системы Domain Name System в целом и ее базы данных в частности. В следующем разделе опишем детали конфигурации, характерные для реализации сервера BIND. Реализация сервера NSD/Unbound будет описана в разделе 17.11.

# 17.9. Программное обеспечение системы BIND

BIND (Berkeley Internet Domain Name System) — это созданное консорциумом ISC открытое программное обеспечение, реализующее протокол DNS для операционных систем Linux, UNIX, Mac OS и Windows. Существует три основные версии пакета BIND: BIND 4, BIND 8 и BIND 9. Система BIND 10 в настоящее время еще разрабатывается консорциумом ISC. В этой книге мы рассмотрим только пакет BIND 9.

## Определение версии

Довольно часто поставщикам не приходит в голову явно указать, какую версию пакета они реализовали в своей системе, поэтому приходится изощряться, чтобы точно выяснить, с каким именно программным обеспечением мы имеем дело. Иногда номер версии можно определить с помощью хитроумного запроса на основе утилиты **dig**, которая является частью пакета BIND. Команда

```
$ dig @сервер version.bind txt chaos
```

возвращает номер версии, содержащийся в конфигурационном файле пакета BIND. Сначала определим имя сервера имен для исследуемого домена

```
$ dig домен ns
```

и затем выполним запрос `version.bind`. Например, вот как эта команда работает с сайтом `isc.org`.

```
$ dig @ns-ext.isc.org version.bind txtx chaos
version.bind.          0   CH   TXT "9.5.1"
```

Однако она не работает с сайтом `cs.colorado.edu`.

```
$ dig @mroe.cs.colorado.edu version.bind txt chaos
version.bind.          0  CH  TXT "wouldn't you like to know..."[17]
```

Некоторые организации так конфигурируют систему BIND, чтобы замаскировать номер версии, исходя из того, что это как бы обеспечивает определенную степень "безопасности, обеспечиваемой неясностью". Мы не одобряем этого, но иногда таким образом можно отразить атаки взломщиков-дилетантов. Более подробно этот вопрос обсуждается немного ниже.

Этот запрос работает и с другим программным обеспечением DNS. Например, запрос

```
$ dig @k.root-servers.net version.bind txt chaos
version.bind.          0  CH  TXT "NSD 2.3.7"
```

показывает, что на корневом сервере имен K выполняется система NSD.

Другой фрагмент информации в классе CHAOS идентифицирует имя запрашиваемого сервера. Однако если вы только что послали серверу запрос, значит, вы должны знать его имя, не так ли? На самом деле некоторые очень загруженные серверы (например, корневые серверы имен) представляют собой множество машин, распределенных по земному шару и объединенных одним именем сервера и IP-адресом. Эта схема репликации называется "альтернативной маршрутизацией" ("antcast routing"). Система маршрутизации находит по вашему запросу "ближайший" экземпляр. Если вы — системный администратор, пытающийся решить проблему, то может оказаться важным идентифицировать сервер, к которому вы получили доступ. Например, можно выполнить такие команды.

```
$ dig @k.root-servers.net hostname.bind txt chaos
hostname.bind.         0  CH  TXT "k2.nap.k.ripe.net"
```

или

```
$ dig @k.root-servers.net id.server txt chaos
id.server.             0  CH  TXT "k2.nap.k.ripe.net"
```

Сообщество IETF пытается стандартизовать эти странные имена класса CHAOS в формах, которые не зависели бы от реализации `version.server` или `id.server`, но только форма `id.server` прошла всю процедуру до конца. Форма `version.server` существует в виде черновика и никогда не была описана документом RFC. Система NSD использует все четыре формы, а пакет BIND — только три одобренные.

🕮 Более подробная информация о стандарте syslog изложена в главе 11.

В качестве системного администратора вы можете запустить сервер имен (**named** или **nsd**) с флагом **-f**, чтобы он напечатал номер своей версии в стандартном виде, и выйти. В системе Linux вы можете запросить у своего менеджера пакетов номер инсталлированной версии. Кроме того, вы обычно можете узнать номер версии системы BIND, заглянув в файлы регистрации в каталоге **/var/log** или их эквивалент. Сервер имен системы BIND регистрирует свой номер версии в системе syslog при загрузке. Команда **grep** в системе BIND выводит на экран примерно такие строки.

```
Jul 13 07:19:55 nubark named[757]: starting BIND 9.5.0-P2 -u named
```

Если эти попытки не принесли результаты, следует помнить, что номер версии команды **dig** обычно совпадает с номером версии сервера **named**, причем утилита **dig** часто устанавливается даже тогда, когда сервер **named** не инсталлирован. Первая строка вывода утилиты **dig** содержит номер версии в качестве комментария.

[17] Цифра 0 в ответе представляет собой значение TTL. Один из наших рецензентов сообщил, что однажды видел такой ответ на этот запрос: "Name is Bind, James Bind!"

Версии системы BIND, содержащейся в наших эталонных операционных системах, перечислены в табл. 17.8. Безопаснее всего использовать текущую версию.

**Таблица 17.8. Версии системы BIND, поставляемой с нашими эталонными операционными системами**

| Система | Версия ОС | Версия BIND | Дата выпуска BIND |
|---|---|---|---|
| ISC | – | 9.6.1-P3 | Январь 2010 |
| Ubuntu | 9.04 | 9.6.1-P2 | Март 2009 |
| SUSE | 10.2 | 9.4.2 | Ноябрь 2007 |
| RHEL | 5.3 | 9.3.4-P1 | Июль 2007 |
| Solaris | 5.10 | 9.3.4-P1 | Июль 2007 |
| OpenSolaris | 2009.06 | 9.6.1-P1 | Июль 2009 |
| HP-UX | 11 | 9.3.2 | Декабрь 2005 |
| AIX | 6.1 | 8.3.3+ или 9.2.1 | Январь 2004 |


Операционная система AIX поставляется с исполняемыми файлами пакетов BIND 8 и BIND 9, которые называются **named8** и **named9** соответственно. При поставке обобщенная форма **named** связывается с файлом **named8**. Символ + в номере версии обозначает сокращение "+Fox_fo_CERT_till_07_15_04".

Большинство поставщиков вносят исправления, касающиеся безопасности, в старые версии вместо обновления последней версии, поступившей от консорциума ISC, поэтому номера версий могут вводить в заблуждение. Легко убедиться, что многие версии являются довольно старыми, поэтому первейшей задачей системного администратора DNS является обновление программного обеспечения.

## Компоненты системы BIND

В пакет BIND входит четыре основных компонента:

- демон сервера имен **named**, отвечающий на запросы;
- библиотечные функции, контактирующие с серверами распределенной базы данных DNS от имени пользователей;
- утилиты **nslookup**, **dig** и **host**, позволяющие выполнять DNS-запросы из командной строки;
- программа **rndc** для дистанционного управления демоном **named**.

Главнейшей задачей системного администратора, работающего с пакетом BIND, является упорядочение многочисленных опций и функциональных возможностей, предоставляемых пакетом BIND, а также выбор наиболее подходящих для текущей ситуации.

## Файлы конфигурации

Полная конфигурация демона **named** включает его конфигурационный файл, файлы данных зоны, содержащие адресные привязки для каждого узла, и файл подсказок для корневого сервера имен. Авторитетные серверы должны иметь файл конфигурации и файлы данных зоны для каждой зоны, относительно которых они являются главными серверами. Кеширующие серверы должны иметь файл конфигурации и файл подсказок для корневого сервера имен. Файл конфигурации демона **named** имеет специфический

формат; все остальные файлы представляют собой коллекции отдельных записей данных DNS, рассмотренных в разделе 17.8.

В конфигурационном файле **named.conf** демона **named** задается роль узла (главный, подчиненный, тупиковый или только кеширующий) и определяется способ, которым он должен получать копию записей о данных каждой зоны, которую он обслуживает. Здесь же приводятся всевозможные параметры — как глобальные, связанные с работой самого демона **named**, так и локальные, связанные с серверами и зонами и относящиеся только к части трафика DNS.

Файл конфигурации состоит из набора инструкций, каждая из которых оканчивается точкой с запятой и описывается в последующих разделах книги. К сожалению, формат файла довольно "хрупкий": достаточно одной пропущенной точки с запятой, чтобы все перестало работать.

К счастью, в пакете BIND имеются удобные средства проверки синтаксиса конфигурационного файла (**named-checkconf**) и зонных файлов (**named-checkzone**). Эти утилиты ищут не только синтаксические ошибки, но и очевидные пропуски. Например, утилита **named-checkzone** выдаст предупреждение, если в файл не включена директива `$TTL`. Но в каждой бочке меда есть ложка дегтя. Например, не выявляются пропущенные связующие записи (см. раздел 17.8), хотя их отсутствие приводит к повышению нагрузки на корневые серверы и серверы организационных доменов gTLD.

Комментарии допускаются везде, где могут стоять пробелы. Поддерживаются комментарии в стиле языков C, C++ и командного интерпретатора, но лучше выбрать один стиль и придерживаться только его.

```
/* Это комментарий, который может занимать несколько строк. */
// Весь текст до конца строки является комментарием.
# Весь текст до конца строки является комментарием.
```

Каждая инструкция начинается с ключевого слова, определяющего ее тип. Может присутствовать несколько инструкций одного типа, за исключением `options` и `logging`. Отдельные инструкции, а также их части могут отсутствовать; в этом случае будут приняты установки по умолчанию. Список поддерживаемых инструкций приведен в табл. 17.9.

**Таблица 17.9. Инструкции, используемые в файле named.conf**

| Инструкция | Назначение |
|---|---|
| `include` | Подключает внешний файл |
| `options` | Задает глобальные параметры конфигурации сервера имен, а также установки по умолчанию |
| `acl` | Формирует списки управления доступом |
| `key` | Определяет параметры аутентификации |
| `trusted-keys` | Задает заранее установленные ключи шифрования |
| `server` | Задает параметры сервера |
| `masters` | Определяет список главных серверов для тупиковых и подчиненных зон |
| `logging` | Устанавливает категории журнальных сообщений и каналы их распространения |
| `statistics-channels` | Выводит текущую статистику в виде XML-файла |
| `zone` | Определяет зону записей о ресурсах |
| `controls` | Определяет, как утилита **ndc** будет управлять сервером имен named |
| `view` | Определяет представление данных зоны |
| `lwres` | Конфигурирует сервер имен named в качестве упрощенного распознавателя |

Прежде чем приступать к описанию этих инструкций и их использования для конфигурирования демона имен **named**, необходимо рассказать о специальной структуре данных, используемой во многих инструкциях, — *список соответствия адресов* (address match list). Этот список является обобщением понятия IP-адреса и может включать:

- IP-адрес, либо v4, либо v6 (например, 199.165.145.4);
- адрес сети с маской CIDR[18] (например, 199.165/16);
- имя ранее определенного списка управления доступом (задается с помощью инструкции `acl`);
- криптографический ключ аутентификации;
- оператор отрицания `!`.

Списки соответствия адресов используются в качестве аргументов инструкций и опций. Приведем ряд примеров.

```
{ ! 1.2.3.13; 1.2.3/24; };
{ 128.138/16; 198.11.16/24; 204.228.69/24; 127.0.0.1; };
```

В первом списке из числа адресов исключается узел 1.2.3.13, но разрешаются другие адреса в сети 1.2.3.0/24. Во втором списке указаны сети, закрепленные за университетом штата Колорадо. Фигурные скобки и завершающая точка с запятой не являются частью списка; они относятся к инструкции, в которой содержится список.

Когда IP-адрес или адрес сети проверяется на соответствие списку, содержимое списка просматривается по очереди до тех пор, пока не будет найдено совпадение. Порядок элементов списка важен, так как ищется первое совпадение. Например, если в первом из показанных выше списков поменять элементы местами, результат не будет достигнут, поскольку адрес 1.2.3.13 удовлетворит первому условию (сетевому адресу 1.2.3.0/24) и второе условие никогда не будет проверяться.

Перейдем к изучению инструкций! Некоторые из них лаконичны и ясны, а для описания других может потребоваться отдельная глава.

## Инструкция include

Когда конфигурационный файл становится слишком большим, можно разбить его на части, поместив их в отдельные файлы. Дополнительные компоненты подключаются к файлу **named.conf** посредством инструкции `include`.

```
include "путь";
```

Если указан относительный *путь*, то он добавляется к имени каталога, заданному в параметре `directory`. Очень часто инструкцию `include` применяют для подключения файлов, содержащих криптографические ключи, которые должны быть недоступными для посторонних. Для того чтобы не запрещать доступ ко всему файлу **named.conf**, ключи хранят в отдельных файлах, которые может читать лишь демон **named**. Эти файлы затем включаются в файл **named.conf**.

Многие организации размещают инструкции `zone` в отдельных файлах, а затем с помощью инструкции `include` объединяют их. Это позволяет отделить части конфигурации, являющиеся относительно статичными, от частей, подвергающихся частым изменениям.

[18] Сетевые маски CIDR описываются в разделе 14.4.

## Инструкция options

Эта инструкция задает глобальные параметры конфигурации, часть из которых впоследствии может быть переопределена для конкретных зон или серверов. Общий формат инструкции имеет следующий вид.

```
options {
  параметр;
  параметр;
    ...
};
```

Если в файле **named.conf** нет инструкции options, принимаются значения по умолчанию.

В пакете BIND существует много параметров, даже слишком много. В версии 9.7 их более 150, что очень затрудняет их изучение. Полный список параметров можно найти в документации. К сожалению, ходят слухи, что разработчики пакета BIND подумывают о том, чтобы удалить часть параметров, которые оказались неудачными или больше не нужны. Это будет ударом для организаций, которые все же используют такие параметры и нуждаются в них. Ниже будут описаны лишь те параметры, которые рекомендуется задавать. (Мы опросили разработчиков пакета BIND и учли их мнения и советы.)

Для более полного изучения параметров рекомендуем обратиться к одной из посвященных системе DNS и пакету BIND книг, упомянутых в конце главы. Кроме того, советуем обратиться к документации, сопровождающей пакет BIND. Все параметры, включая их синтаксис и значения по умолчанию, описаны в документе **ARM**, находящемся в каталоге **doc** и входящем в дистрибутивный набор пакета. Кроме того, полный список параметров содержится в файле **doc/misc/options**.

По мере изложения будем сопровождать описание параметров краткими комментариями. Значения по умолчанию указываются в квадратных скобках после самого параметра. В большинстве случаев значения по умолчанию вполне приемлемы. Сами параметры перечисляются без определенного порядка.

### Местоположение файлов

```
directory "путь";                [каталог запуска сервера]
key-directory "путь";            [совпадает с параметром directory]
```

Инструкция directory задает каталог, в который должен перейти демон **named**. Когда в конфигурационном файле встречаются относительные пути к файлам, они интерпретируются относительно этого каталога. Параметр *путь* должен быть абсолютным. В этот каталог помещаются все выходные файлы (отладочные, статистические данные и т.д.). Параметр key-directory определяет место хранения криптографического ключа. Он не должен быть доступен для посторонних.

Мы рекомендуем хранить все конфигурационные файлы пакета BIND (кроме **named.conf** и **resolv.conf**) в подкаталоге каталога **/var** (или любого другого каталога, где содержатся конфигурационные файлы других программ). Мы предпочитаем каталоги **/var/named** или **/var/domain**.

### Идентификация сервера имен

```
version "строка"          [реальный номер версии сервера]
hostname "строка"         [реальное имя сервера]
server-id "строка"        [реальный номер версии сервера]
```

Строка `version` идентифицирует версию программного обеспечения сервера имен, а строка `hostname` — сам сервер, как и строка `server-id`. Эти параметры позволяют скрывать истинные значения. Каждый из них позволяет записывать данные записи TXT в класс CHAOS, где любопытные хакеры могут найти их с помощью команды **dig**.

Мы не советуем скрывать эти значения. С их помощью удобно запрашивать серверы имен и выяснять номера их версий, например, если вам необходимо узнать, установил ли ваш поставщик текущую версию или не надо ли установить новейшую версию всех серверов. Если все же приходится прятать номер версии, то, по крайней мере, введите строку, которая неявно сообщала бы вашему системному администратору информацию о версии. (Именно для этого предназначена новая запись о ресурсах NSID.) Данные, хранящиеся в этой текстовой записи, представляют собой строку, смысл которой понятен только вашему системному администратору, но не постороннему человеку.

Параметры `hostname` и `server-id` были включены относительно недавно и предназначены для дублирования экземпляров корневого и общих доменов верхнего уровня при альтернативной маршрутизации.

🕮 Синхронизация зон

```
notify yes | master-only | explicit | no;                [yes]
also-notify id_адреса_серверов;                           [пусто]
allow-notify список_соответствующих_адресов;              [пусто]
```

Параметры `notify` и `also_notify` применяются только к главным серверам, а параметр `allow-notify` — только к подчиненным.

В ранних версиях сервер BIND синхронизировал файлы зон между главным и подчиненным серверами только по истечении тайм-аута для обновления, заданного в зонных записях SOA. В настоящее время главный сервер **named** автоматически уведомляет подчиненные серверы при перезагрузке соответствующей зонной базы данных, если параметр `notify` равен `yes`. В ответ на полученное уведомление подчиненные серверы связываются с главным сервером, чтобы проверить, изменились ли файлы, и обновляют свои копии данных.

Параметр `notify` может устанавливаться как на глобальном уровне, так и на уровне отдельных зон. Это позволяет быстрее обновлять зонные файлы при внесении изменений. По умолчанию каждый авторитетный сервер посылает обновления всем другим авторитетным серверам (такой способ Роль Викси (Paul Vixie) назвал "разбрызгиванием"). Если параметр `notify` установлен равным `master-only`, то "разговорчивость" сервера ограничена и уведомления посылаются только подчиненным серверам зон, по отношению к которым данный сервер является главным. Если параметр `notify` установлен равным `explicit`, то демон **named** уведомляет только серверы, указанные в разделе `also-notify`.

🕮 Более подробно тупиковые зоны обсуждаются в подразделе "Инструкция zone".

Демон **named** выясняет, какие компьютеры являются подчиненными серверами зоны, просматривая записи NS этой зоны. При наличии параметра `also-notify` будут также уведомляться дополнительные серверы, которые не зарегистрированы посредством записей NS. Подобный прием используется, если в организации имеются внутренние серверы. В списке `also-notify` не нужно указывать тупиковые серверы: их интересуют только записи NS, поэтому они могут участвовать в обычном цикле обновления.

В списке `also-notify` должны присутствовать только IP-адреса и, возможно, порты. Для серверов с несколькими интерфейсами дополнительные параметры задают IP-адреса и порты, предназначенные для исходящих уведомлений. Уведомления желатель-

но также отключать для зон обратного преобразования узла **localhost**, поскольку они никогда не изменяются. Раздел `also-notify` следует использовать только в том случае, когда вы хотите, чтобы вторичные серверы уведомлял сервер имен, а не главный сервер.

```
recursion yes | no;                                    [yes]
allow-recursion { список_соответствия_адресов }; [все узлы]
```

Параметр `recursion` указывает на то, должен ли демон **named** обрабатывать запросы пользователей рекурсивно. Этот параметр можно задать для авторитетного сервера своей зоны, но мы не рекомендуем это делать. Лучше всего разделить авторитетные и кеширующие серверы.

Если сервер имен должен быть рекурсивным, то установите параметр `recursive` равным `yes` и включите раздел `allow-recursion`, чтобы демон **named** мог отличать запросы, исходящие из вашей организации, от удаленных запросов. Демон **named** будет действовать рекурсивно для пользователей вашей организации и нерекурсивно — для всех остальных. Если ваш сервер имен рекурсивно обрабатывает все запросы, то он называется открытым распознавателем (open resolver) и может стать рефлектором для некоторого вида атак (см. документ RFC5358).

### Использование кеширующей памяти

Если ваш сервер имеет ограниченный объем памяти, то иногда необходимо использовать параметры `recursive-clients` и `max-cache-size`. Параметр `recursive-clients` управляет многочисленными рекурсивными процессами поиска, которые сервер выполняет одновременно. Каждый из них требует около 20 двоичных Кбайт памяти. Параметр `max-cache-size` ограничивает объем памяти сервера, который будет использоваться для кеширования ответов на запросы. Если объем кеша увеличивается слишком сильно, то демон **named** удаляет записи до истечения их периода TTL, чтобы предотвратить превышение лимита.

### Использование IP-порта

```
use-v4-udp-ports [ range начало конец];          [range 1024 65535]
use-v6-udp-ports [ range начало конец];          [range 1024 65535]

avoid-v4-udp-ports [ список_портов ];            [пусто]
avoid-v6-udp-ports [ список_портов ];            [пусто]

query-source-v4 v4-адрес [ порт ];  [любой] #ВНИМАНИЕ, не используйте port
query-source-v6 v6-адрес [ порт ];  [любой] #ВНИМАНИЕ, не используйте port
```

Порты источника стали играть важную роль в системе DNS после обнаружения ошибки протокола DNS, выявленной Дэном Камински (Dan Kaminsky). Этот недостаток допускает атаку кеша DNS, когда серверы имен используют предсказуемые порты источника и идентификаторы запросов. Параметры `use-` и `avoid-` для портов UDP в сочетании с изменениями программного обеспечения демона **named** смягчили последствия таких атак. Не используйте параметр `query-address` для того, чтобы задать фиксированный порт источника для запросов DNS, иначе вы не сможете воспользоваться "защитой Каминского", которая предусматривает использование большого количества случайных портов.

Значения, задаваемые по умолчанию для параметров `use-v*-udp-ports`, не требуют изменений. Если ваш брандмауэр блокирует некоторые порты из заданного диапазона (например, порт 2049 для файловой системы SunPRC), то может возникнуть небольшая проблема. Когда ваш сервер имен посылает запрос, используя в качестве источника один

из заблокированных портов, брандмауэр блокирует ответ и сервер имен в конце концов прекращает ожидание и посылает запрос снова. Это не фатально, но раздражает.

Во избежание этого следует использовать параметр `avoid-v*-udp-ports`, заставляющий систему BIND избегать заблокированных портов. Любые порты UDP с крупными номерами, заблокированные вашим брандмауэром, должны быть включены в этот список.[19] Если вы обновляете свой брандмауэр в ответ на опасную атаку, то не забудьте обновить список портов тоже.

Параметры `query-source` позволяют указывать IP-адреса, предназначенные для исходящих запросов. Например, может возникнуть необходимость использовать конкретный IP-адрес, чтобы пройти через брандмауэр или отличить внутреннее представление от внешнего.

Запросы исходят из случайно выбранных портов с крупными номерами, а ответы возвращаются обратно через те же самые порты. Следовательно, ваш брандмауэр должен быть подготовлен к приему UDP-пакетов, поступающих на порты с крупными номерами. Некоторые системные администраторы используют специальный номер порта источника, который можно указать брандмауэру, чтобы он распознавал его и принимал UDP-пакеты только через этот порт. Однако эта конфигурация больше не является безопасной из-за дефекта протокола DNS, обнаруженного Камински.

Если вы используете параметр `query-source`, то укажите только IP-адрес, с которого хотите посылать запросы; не указывайте номер порта.

🕮 Использование переадресации

```
forwarders { ip_адрес; ip_адрес; ... };    [пустой список]
forward only | first;                      [first]
```

Вместо того чтобы каждый сервер имен выполнял собственные внешние запросы, можно сделать один или несколько серверов *переадресующими*. В такой конфигурации обычный сервер просматривает в своем кеше записи, для которых он авторитетен, и, если не находит ответ, посылает запрос переадресующему серверу. Последний создает кеш, которым пользуются все остальные серверы. Назначение происходит неявно — в файле конфигурации сервера нет никакой информации о том, что он является переадресующим.

В параметре `forwarders` приводится список IP-адресов компьютеров, являющихся переадресующими серверами. Они опрашиваются по очереди. При использовании переадресующего сервера традиционная процедура поиска домена, которая начинается с корневого сервера, а затем продолжается по цепочке отсылок, нарушается, поэтому нужно внимательно следить за тем, чтобы не возникало циклов переадресации.

Сервер, для которого установлен атрибут `forward only`, опрашивает переадресующие серверы и кеширует их ответы самостоятельно. Если переадресующий сервер не ответит, запрос потерпит неудачу. Сервер с атрибутом `forward first` связывается с переадресующими серверами, но при необходимости может обработать запрос самостоятельно.

Поскольку у параметра `forwarders` нет значения по умолчанию, переадресация происходит только в том случае, когда она задана явно. Включать переадресацию можно либо глобально, либо в пределах отдельных зон.

🕮 Разрешения

```
allow-query { список_соответствия_адресов };          [все узлы]
allow-query-cache { список_соответствия_адресов };    [все узлы]
```

[19] Некоторые брандмауэры запоминают состояние и могут распознавать ответ системы DNS на запрос, посланный секунду назад. Таким серверам не требуется помощь, обеспечиваемая параметрами `avoid-v*-udp-ports`.

```
allow-transfer { список_соответствия_адресов };      [все узлы]
allow-update { список_соответствия_адресов };        [нет]
blackhole { список_соответствия_адресов };           [нет]
```

Эти параметры позволяют указать, какие узлы (или сети) могут обращаться к серверу имен и запрашивать пересылку зонных баз данных, а также динамически обновлять зоны. Списки соответствия адресов представляют собой слабый способ обеспечения безопасности и уязвимы для подделки IP-адресов, поэтому полагаться на них рискованно. Вероятно, не составит никакого труда заставить ваш сервер ответить на DNS-запрос, но следует избегать использования параметров `allow_update` и `allow_transfer`; вместо них следует использовать криптографические ключи.

В списке `blackhole` перечислены серверы, которые никогда не "удостоятся внимания" демона **named**: он не будет принимать от них запросы и обращаться к ним за ответом.

### Размеры пакетов

```
edns-udp-size номер;                [4096]
max-udp-size номер;                 [4096]
```

Все компьютеры в Интернете должны иметь возможность заново собирать фрагментированные UDP-пакеты, размер которых составляет 512 байт и меньше. Эти консервативные требования имели смысл в 1980-х годах, но в настоящее время такой размер смехотворно мал. Современные маршрутизаторы и брандмауэры могут обрабатывать намного более крупные пакеты, но достаточно всего одного неправильного звена в цепочке IP-адресов, чтобы прервать весь путь.

Поскольку система DNS использует для запросов пакеты UDP, а ответы часто превышают 512 байт, ее администраторы иногда опасаются утери крупных UDP-пакетов. Если большой по размеру пакет был фрагментирован, а ваш брандмауэр пропускает только первый фрагмент, то получатель получит усеченный ответ и повторно пошлет запрос по протоколу TCP. Протокол TCP связан с намного более крупными затратами ресурсов, в то же время загруженные серверы в корневом узле и серверы TLD не должны увеличивать трафик по протоколу TCP из-за того, что у кого-то вышел из строя брандмауэр.

Параметр `edns-udp-size` задает размер буфера для повторной сборки, который сервер имен сообщает посредством протокола EDNS0, представляющего собой расширенный вариант протокола DNS. Параметр `max-udp-size` задает максимальный размер пакета, который сервер может реально посылать. Оба размера измеряются в байтах. Разумным диапазоном является 512–4096 байт.

Оба значения по умолчанию равны 4096 байт. Это позволяет использовать новые функциональные возможности, предоставляемые спецификациями DNSSEC. Однако некоторые (неисправные) брандмауэры не допускают UDP-пакеты, размер которых превышает 512 байт, а другие брандмауэры настроены так, чтобы блокировать все, кроме первого пакета фрагментированного UDP-ответа. Единственным выходом из такой ситуации является ремонт брандмауэров.

Для того чтобы понять, какой размер пакета подходит для вашей организации, попробуйте выполнить команду **`dig rs.dns-oarc.net txt`** и посмотрите на ответ. Более подробно эта тема освещена в разделе 17.13, где речь идет о размере ответов, которые посылает сервер DNS-OARS. Если эта утилита показывает, что размер ответа невелик, то проблема, вероятно, заключается в брандмауэрах, которые следует исправить.

В качестве временного решения можно попытаться установить параметр `max-udp-size` равным значению, которое возвращает сервер DNS-OASR. Эта установка позволяет демону **named** внедрять свои ответы в пакеты, которые могут остаться нефрагментиро-

ванными. Параметр edns-udp-size также следует установить равным этому значению, чтобы ваши пакеты могли проходить в обоих направлениях. После исправления брандмауэров не забудьте восстановить прежние значения параметров — 4096 байт!

Не прибегайте к этим мерам, если не уверены, что проблема с размерами пакетов действительно существует, поскольку они также ограничивают размер пакета, передаваемого вдоль путей, которые на самом деле могли бы пропустить 4096 байт.

🕮 Управление спецификациями DNSSEC

```
dbssec-enable yes | no;                          [yes]
dbssec-validation yes | no;                      [yes]
dbssec-lookaside домен trust-anchor домен;       [".", "dlv.isc.org"]
dnssec-must-be-secure домен yes | no;            [none]
```

Эти параметры конфигурируют поддержку спецификаций DNSSEC. Обсуждение спецификаций DNSSEC и подробное описание их настроек приведены в разделе 17.13.

Параметр dsnsec-enable на авторитетном сервере должен быть включен. Для рекурсивных серверов должны быть включены параметры dnssec-enable и dnssec-validation, а также указана точка доверия с помощью инструкции trusted-key.

Если в домене нет точки доверия, то программное обеспечение попытается найти ее с помощью параметра dnssec-lookaside, обойдя проблему, которая возникает с родительским доменом, не использующим спецификаций DNSSEC.

По умолчанию параметры dnssec-enable и dnssec-validation включены. Это приводит к таким последствиям.

- Авторитетный сервер подписанной зоны с включенным битом DNSSEC, отвечая на запрос, возвращает ответ, содержащий требуемые записи о ресурсах и их подписи.
- Авторитетный сервер подписанной зоны с выключенным битом DNSSEC, отвечая на запрос, возвращает ответ, содержащий только требуемые записи о ресурсах, как это делалось до появления спецификаций DNSSEC.
- Авторитетный сервер неподписанной зоны возвращает ответ, содержащий только требуемые записи о ресурсах; подписи в него не включаются.
- Рекурсивный сервер посылает запросы от имени пользователей с включенным битом DNSSEC.
- Рекурсивный сервер проверяет подписи, включенные в подписанные ответы, прежде чем возвращать данные пользователю.

Параметрами в разделе dnssec-lookaside являются два домена. Например, значения по умолчанию эквивалентны следующей строке конфигурации.

```
dnssec-lookaside "." trust-anchor "div.isc.org";
```

Эта конфигурация сообщает серверам имен, пытающимся установить цепочку доверия, что они должны соединиться с узлом div.isc.org , если не могут получить информацию о безопасном делегировании от корневого узла дерева имен DNS. Как только корневые домены и домены верхнего уровня станут подписанными и будут обслуживаться с помощью протокола DNSSEC, ассоциативная проверка (lookaside validation) станет ненужна. Обсуждение преимуществ и недостатков структуры DLV (DNSSEC Lookaside Validation) и последствий ее применения можно найти в разделе 17.13.

Элемент конфигурации dnssec-must-be-secure позволяет указать, что вы будете принимать только безопасные ответы от конкретных доменов или, наоборот, вам все равно и вы согласны принимать небезопасные ответы. Например, вы можете задать значения yes для домена important-stuff.mybank.com и no — для домена marketing.

mybank.com. Домен, о котором идет речь, должен быть упомянут в разделе trusted-keys или зарегистрирован DLV-сервером.

🕮 Статистика

```
zone-statistics yes | no     [no]
```

Этот элемент конфигурации заставляет демона **named** накапливать статистические данные как о зонах, так и в целом. Более подробно вопросы накапливания статистических данных и их отображения обсуждаются в разделе 17.15.

🕮 Настройка производительности

```
clients-per-query целое_число; [10]        #Количество клиентов,
                                           #ожидающих один и тот же запрос
max-clients-per-query целое_число; [100]   #Максимально допустимое количество
                                           #клиентов до отказа сервера
datasize целое_число  [неограничено]       #Максимальное количество памяти,
                                           #которое может использовать сервер
files целое_число     [неограничено]       #Максимальное количество
                                           #одновременно открытых файлов
lame-ttl целое_число; [10min]              #Период времени до кеширования
                                           #сбойного сервера
max-acashe-size целое_число []             #Размер кеша для дополнительных
                                           #данных
max-cashe-size целое_число []              #Максимальный размер памяти
                                           #для кешированных ответов
max-cashe-ttl целое_число [1week]          #Максимальный период TTL для
                                           #кеширования позитивных данных
max-journal-size целое_число []            #Максимальный размер файла
                                           #для журнала трансакций
max-ncashe-ttl целое_число []              #Максимальный период TTL для
                                           #кеширования негативных данных
tcp-clients целое_число;   [100]           #Максимально допустимое количество
                                           #TCP-клиентов
```

Этот длинный список элементов конфигурации можно использовать для настройки демона **named**, чтобы он хорошо работал на вашем аппаратном обеспечении. Мы не описываем их подробно, но если у вас возникнут проблемы с производительностью, сначала следует попробовать устранить их с помощью этих настроек.

Итак, мы завершаем обзор параметров! Перейдем к остальным инструкциям.

## Инструкция acl

Список управления доступом — это просто именованный список соответствия адресов.

```
acl имя_списка {
   список_соответствия_адресов
};
```

Заданное имя можно указывать везде, где требуется список соответствия адресов.

Инструкция acl должна быть самой первой в файле **named.conf**, так что не пытайтесь ставить ее среди других объявлений. Файл **named.conf** читается за один проход, поэтому списки управления доступом должны быть определены до того, как на них встретится ссылка. Существует четыре стандартных списка:

- any — соответствует всем узлам;
- localnets — всем узлам локальной сети;

- `localhost` — самому компьютеру;
- `none` — не соответствует ни одному узлу.

Сети, входящие в группу `localnets`, определяются адресами интерфейсов компьютера с учетом сетевых масок.

## Инструкция key (TSIG)

Рассматриваемая инструкция определяет именованный ключ шифрования (т.е. пароль), используемый для аутентификации соединения между двумя серверами, например между главным и подчиненным серверами при передаче зоны или между сервером и процессом **rdns**, который им управляет. Подробнее о механизмах аутентификации, используемых в пакете BIND, речь пойдет в разделе 17.13. Здесь же мы лишь вкратце опишем суть процесса.

Для создания ключа нужно указать как алгоритм шифрования, так и совместный секретный ключ, который представлен в виде строки, закодированной в формате Base64 (см. раздел 17.13).

```
key идентификатор_ключа {
   algorithm строка;
   secret строка;
};
```

Как и в случае списков управления доступом, идентификатор ключа должен быть определен в файле **named.conf** с помощью инструкции `key` до того, как встретится ссылка на ключ. Для того чтобы связать ключ с конкретным сервером, включите идентификатор ключа в список `keys` соответствующей инструкции `server`. Ключ используется как для проверки запросов, поступающих от сервера, так и для подписи ответов на эти запросы.

Секретная информация, принадлежащая группе лиц, не должна храниться в файле, доступном для внешнего мира. Для включения его в файл **named.conf** следует использовать инструкцию `include`.

## Инструкция trusted-keys

С теоретической точки зрения родительские зоны DNSSEC аутентифицируют открытые ключи своих дочерних зон, позволяя создавать цепочки аутентифицированных подписей вплоть до корня системы DNS. На практике корневой домен и домены верхнего уровня еще не поддерживают спецификации DNSSEC, поэтому необходим другой метод проверки открытых ключей зон.

Инструкция `trusted-keys` является довольно грубым способом сообщить демону **named**, что правильным открытым ключом зоны XXX.com является YYY, обойдя тем самым механизм DNSSEC, предназначенный для получения и верификации зонных ключей. Такое объявление иногда называют "точкой доверия" ("trust anchor").

Разумеется, зона XXX.com должна быть достаточно важной для вашей организации, чтобы уделить ей особое внимание, и вы должны иметь безопасный, внеполосный способ определения правильного значения ее ключа. Нет волшебных способов получить правильный ключ; администратор внешней зоны должен прочитать его вам по телефону или послать как-то иначе. Для этой цели часто используется безопасная с точки зрения протокола HTTPS веб-страница. При изменении ключа этот процесс придется повторить с самого начала.

Формат инструкции `trusted-keys` имеет следующий вид.

```
trusted-keys {
  домен флаги протокол алгоритм ключ;
  домен флаги протокол алгоритм ключ;
  ...
```

Каждая строка представляет точку доверия для конкретного домена. Атрибуты *флаги*, *протокол* и *алгоритм* являются неотрицательными целыми числами. Атрибут *ключ* — это строка, закодированная в формате Base64.

Инструкция `trusted-keys` используется в тех случаях, когда зона имеет цифровую подпись, а ее родительская зона — нет, поэтому нельзя быть уверенным в том, что открытый ключ, получаемый от DNS-сервера, действительно надежен.

Подробнее о технологии DNSSEC рассказывается в разделе 17.13.

## Инструкция server

Демон **named** способен общаться с серверами, которые не используют последнюю версию пакета BIND или просто неправильно сконфигурированы. Инструкция `server` сообщает демону характеристики удаленных серверов. Она может заменять стандартные значения параметров конкретного сервера, хотя это не обязательно, пока вы не захотите сконфигурировать ключи для передачи зоны.

```
server ip-адрес {
    bogus yes | no;                              [no]
    provide-ixfr yes | no;                       [yes]
    request-ixfr yes | no;                       [yes]
    keys {идентификатор_ключа; идентификатор_ключа; ...}; [нет]
    transfer-source ip-адрес       [ближайший интерфейс]
    transfer-source-v6 ipv6-адрес  [ближайший интерфейс]
};
```

Посредством инструкции `server` можно переопределять значения глобальных конфигурационных параметров, относящихся к серверам. Просто укажите нужные параметры и их новые значения. Мы показали не все параметры инструкции `server`, а только самые необходимые. Полный список параметров можно найти в документации пакета BIND.

Если для сервера задан атрибут `bogus`, демон **named** не посылает ему запросы. Этот атрибут следует устанавливать для серверов, которые работают неправильно. Параметр `bogus` отличается от глобального атрибута `blackhole` тем, что подавляет только внешние запросы. В отличие от него, атрибут `blackhole` полностью исключает все формы взаимодействия с перечисленными серверами.

Сервер версии, являющийся главным сервером динамически обновляемой зоны, выполняет инкрементные зонные передачи, если атрибут `provide-ixfr` равен `yes`. Точно так же подчиненный сервер запрашивает инкрементные зонные передачи от главного сервера, если атрибут `request-ixfr` равен `yes`. Динамическая система DNS обсуждается в разделе 17.12.

В списке `keys` задаются идентификаторы ключей, которые ранее были определены в инструкции `key` для использования в сигнатурах транзакций TSIG (о них рассказывается в разделе 17.13). К любому запросу, посылаемому удаленному серверу, будет добавлена сигнатура, зашифрованная посредством этого ключа. Запросы, поступающие от удаленного сервера, могут не иметь цифровой подписи, но если она есть, то будет проверена.

Атрибут `transfer-source` определяет IPv4- и IPv6-адреса интерфейса (и, возможно, порта), которые следует использовать в качестве адреса источника (порта) для запросов на передачу зоны. Эти атрибуты необходимы только тогда, когда система имеет несколько интерфейсов и IP-адрес удаленного сервера указан в списке `allow-transfer`.

## Инструкция masters

Эта инструкция позволяет задавать имена одного или нескольких главных серверов, указывая их IP-адреса и криптографические ключи. Затем это имя можно использовать в разделе `masters` инструкций `zone` вместо повторения IP-адресов и ключей.

🕮 В каких случаях главных серверов бывает несколько? Ответ см. через несколько страниц.

Инструкция `masters` удобна, когда несколько подчиненных или тупиковых зон получают данные от одного и того же удаленного сервера. Если адреса и криптографические ключи удаленного сервера изменились, вы можете обновить инструкцию `masters`, которая их задает, а не изменять многочисленные инструкции `zone`.

Синтаксис этой инструкции выглядит следующим образом.

```
masters имя [ ip-адрес [port ip-порт] [key ключ]; ...]
```

## Инструкция logging

Демон **named** в настоящее время заслуживает награды "За наиболее конфигурируемую подсистему журнальной регистрации на Земле". Система Syslog предоставляет программистам контроль над приоритетами сообщений, а системным администраторам — контроль над местом хранения этих сообщений. Но в рамках заданного приоритета администратор не может указать: "Это сообщение меня интересует, а это — нет". В систему BIND были добавлены категории, позволяющие классифицировать сообщения по типам, и каналы, расширяющие возможности в плане хранения сообщений. Категории назначаются программистом, а каналы — администратором.

Поскольку вопросы журнальной регистрации лежат несколько в стороне от нашего повествования (особенно если учесть объем материала), мы рассмотрим их позднее.

## Инструкция statistics

Эта инструкция `statistics-channels` позволяет соединиться с выполняемым демоном **named** с помощью браузера и просмотреть статистические показатели, которые на нем собраны. Поскольку статистические показатели сервера имен могут быть конфиденциальными, следует ограничить доступ к этим данным, открыв их только для доверенных узлов вашей собственной организации. Синтаксис этой инструкции выглядит следующим образом.

```
statistics-channels {
    inet(ip-адрес | *) port порт# allow { список_соответствия_адресов } ;
    ...
}
```

В файл конфигурации можно включить несколько последовательностей `inet-port-allow`. По умолчанию IP-адреса заданы значением `any`, а порт — значением 80 (обычное значение для порта HTTP). Для использования каналов сбора статистики демон **named** следует компилировать с помощью программы **libxml2**.

## Инструкция zone

Это "сердце" файла **named.conf**, которое сообщает демону **named** о зонах, для которых он авторитетен, и задает параметры управления каждой зоной. Инструкция `zone` также используется для предварительной загрузки "подсказок" с корневого сервера ("подсказки" — это имена и адреса корневых серверов, участвующих в инициализации DNS-поиска).

Точный формат инструкции `zone` зависит от роли, которую демон **named** должен играть в отношении этой зоны. Возможными типами зоны являются `master`, `slave`, `hint`, `forward`, `stub` и `delegation-only`. Мы не будем рассматривать зоны типов `stub` (используется только в системе BIND) и `delegation-only` (применяется для прекращения использования записей с шаблонными символами в зонах верхнего уровня при извещении служб регистратора). Остальные типы зон описаны в последующих разделах.

Многие из рассмотренных ранее глобальных параметров могут быть частью инструкции `zone`, т.е. переопределять глобальные установки. Мы не будем повторно упоминать о них, за исключением тех параметров, которые наиболее часто используются.

### *Конфигурирование главного сервера зоны*

Ниже показан формат инструкции `zone` для зоны, в которой демон **named** является главным сервером имен.

```
zone "имя_домена" {
   type master;
   file "путь";
};
```

Доменное имя в спецификации зоны всегда дается в двойных кавычках.

Зонная база данных хранится на диске в текстовом файле, доступном для редактирования. Поскольку нет соглашения об именовании этого файла, в объявлении зоны должна присутствовать директива `file`. Зонный файл представляет собой набор записей о DNS-ресурсах; их формат описан в разделе 17.8.

В инструкции `zone` часто задается также ряд других серверных атрибутов.

```
allow-query { список_соответствия_адресов };     [все узлы]
allow-transfer { список_соответствия_адресов }; [все узлы]
allow-update { список_соответствия_адресов };    [none]
zone-statistics yes | no;                        [no]
```

Параметры, связанные с управлением доступом, не являются обязательными, но мы рекомендуем их использовать. Они содержат либо IP-адрес, либо криптографический ключ TSIG. Как правило, криптографический ключ является более безопасным. Если для зоны поддерживаются динамические обновления, в параметре `allow-update` должен содержаться список узлов, от которых разрешено принимать данные. Динамические обновления применимы только в отношении главных зон; параметр `allow-update` не может присутствовать в объявлении подчиненной зоны. Убедитесь, что в списке указаны лишь локальные компьютеры (например, DHCP-серверы), а не весь Интернет[20].

Параметр `zone-statistics` заставляет демон **named** отслеживать статистику запросов/ответов, например точное число и общий процент отсылок, рекурсивных и ошибочных запросов. Конкретные примеры приведены в разделе 17.15.

[20] Необходима также входная фильтрация на брандмауэре (см. раздел 22.11), а еще лучше использовать технологию аутентификации TSIG.

При наличии столь большого числа зонных параметров (есть еще порядка сорока, о которых мы не рассказали) конфигурация зоны кажется довольно сложной. На самом деле допускается объявление главной зоны, состоящее только из пути к зонному файлу. В BIND 4 больше ничего нельзя задать. Ниже показан пример, взятый из документации и слегка модифицированный нами.

```
zone "example.com" {
    type master;
    file "forward/example.com";
    allow-query { any; };
    allow-transfer { my-slaves; };
};
```

Имя `my-slaves` относится к списку управления доступом, определенному ранее.

### *Конфигурирование подчиненного сервера зоны*

Формат инструкции `zone` для подчиненного сервера будет почти таким же, как и в случае главного сервера.

```
zone "имя_домена" {
    type slave;
    file "путь";
    masters { ip_адрес [port ip_порт] [key имя_ключа]; ... }; [нет]
    allow-query { список_соответствия_адресов };     [все узлы]
};
```

Подчиненные серверы обычно хранят полную копию зонной базы данных. Директива `file` задает имя локального файла, в котором сохраняется реплицированная база данных. Получив новую копию данных зоны, сервер сохраняет ее в этом файле. В случае сбоя и перезагрузки сервера файл можно просто прочитать с диска, а не пересылать по сети.

Редактировать файл реплицированной базы данных не нужно, так как он управляется демоном **named**. Тем не менее имеет смысл просмотреть его, если есть подозрение, что в базу данных главного сервера закралась ошибка. Файл подчиненной зоны создается уже после того, как демон **named** интерпретировал исходные данные зоны и раскрыл все относительные имена. Когда в файле присутствуют имена наподобие

```
128.138.243.151.cs.colorado.edu.
anchor.cs.colorado.edu.cs.colorado.edu.
```

можно быть уверенным в том, что где-то пропущена завершающая точка.

В списке `masters` перечислены IP-адреса компьютеров, с которых может быть загружена зонная база данных. Она также может содержать имя списка главных серверов, определенных предыдущей инструкцией `masters`.

Ранее мы говорили, что только один компьютер может быть главным сервером зоны, так почему же разрешается список, состоящий из нескольких адресов? Во-первых, у главного компьютера может быть несколько сетевых интерфейсов и, соответственно, несколько IP-адресов. Когда один из интерфейсов становится недоступным (проблемы с сетью или маршрутизацией), остаются в резерве другие интерфейсы. Следовательно, рекомендуется указывать все топологически различные адреса главного сервера.

Во-вторых, демону **named** в действительности все равно, откуда поступают данные зоны. Он так же легко загружает базу данных с подчиненного сервера, как и с главного. Этой особенностью демона можно воспользоваться для того, чтобы сделать подчиненный сервер, с которым легко устанавливается связь, резервной копией главного серве-

ра. В любом случае IP-адреса проверяются по очереди до тех пор, пока не будет найден работающий сервер. Теоретически можно даже создать иерархию серверов, в которой главный сервер обслуживает несколько серверов второго уровня, а те, в свою очередь, обслуживают множество серверов третьего уровня.

Советуем в директиве `masters` указывать только адреса главного сервера.

### Задание корневых "подсказок"

Инструкция `zone` типа `hint` сообщает демону **named** местонахождение файла, из которого он может загрузить имена и адреса корневых серверов имен, чтобы предварительно заполнить свой кеш.

```
zone "." {
   type hint;
   file "путь";
};
```

"Подсказки" — это набор DNS-записей, в которых перечислены серверы корневого домена ("."). Они нужны для того, чтобы демон **named** знал, откуда начинать поиск доменов. Не имея "подсказок", демон знал бы только о тех доменах, которые сам обслуживает, а также об их поддоменах.

Когда демон **named** начинает работу, он повторно загружает подсказки с одного из корневых серверов. Следовательно, все будет в порядке, если файл подсказок содержит хотя бы одну запись с правильным и доступным корневым сервером. Для резервирования подсказки о корневом сервере также компилируются в демон **named**.

Файл "подсказок" чаще всего называется **root.cache**. В нем содержатся результаты ответов, которые можно получить, запросив у корневого сервера список серверов имен в домене ".". На самом деле вы можете сгенерировать файл подсказок, выполнив утилиту **dig**. Рассмотрим пример.

```
$ dig @f.root-servers.net . ns > root.cache
```

Обратите внимание на точку. Если сервер `f.root-servers.net` не отвечает, вы можете выполнить запрос, не указывая конкретный сервер.

```
$ dig . ns > root.cache
```

Результат будет аналогичный. Однако вы получите список корневых серверов от кеша локального сервера имен, а не от авторитетного сервера. Это должно быть удобно — даже если вы не перезагружали или перезапускали сервер имен год или два, он периодически обновляет свои записи о корневых серверах по истечении их периода TTL.

### Задание зоны переадресации

Зона типа `forward` переопределяет глобальные параметры переадресации демона **named** (сначала опрашиваем корневой сервер, а потом следуем по цепочке отсылок) для конкретного домена.

```
zone "имя_домена" {
   type forward
   forward only | first;
   forwarders { ip_адрес; ip_адрес; ...}
};
```

Создавать такую зону имеет смысл, если у организации имеется деловой партнер и нужно направлять трафик непосредственно его серверам имен в обход стандартного

пути распространения запросов. Благодаря этому можно получить доступ к серверам имен, которые невидимы внешнему миру.

## Инструкция controls для команды rdnc

Инструкция `controls` определяет, каким образом команда **rndc** будет управлять работой демона **named**. Эта команда может запускать и останавливать демон, выводить отчет о его состоянии, переводить демон в режим отладки и т.д. Будучи сетевой утилитой, она требует должной конфигурации, чтобы злоумышленники не могли через Интернет влиять на работу сервера имен. Формат инструкции `controls` имеет следующий вид.

```
controls {
   inet ip_адрес port ip_порт allow { список_соответствия_адресов }
     keys {список_ключей}
}
```

В BIND также разрешается указывать порт, но если параметр `ports` опущен, по умолчанию принимается порт 953.

Удаленное управление сервером имен кажется одновременно и удобной, и опасной процедурой. Аутентификация необходима, причем ключи, указанные в списке соответствия адресов, игнорируются, а используются только те ключи, которые перечислены в списке `keys`.

В системе BIND существует команда **rndc-confgen**, позволяющая сгенерировать ключ аутентификации, используемый в диалоге между командой **rndc** и демоном **named**. Это можно сделать двумя способами. Первый способ — заставить обе стороны загрузить ключ из общего конфигурационного файла (**/etc/rndc.key**) или поместить ключ в два разных файла (**/etc/rndc.conf** для **rndc** и **/etc/named.conf** для **named**). Второй способ сложнее, но он необходим, если демон **named** и команда **rndc** запускаются на разных компьютерах. Команда **rndc-confgen -a** задает ключи для доступа к локальному узлу.

При отсутствии инструкции `controls` в список соответствия адресов заносится адрес интерфейса обратной связи, а ключ ищется в файле **/etc/rndc.conf**. Поскольку аутентификация в этой версии пакета обязательна, команда **rndc** не сможет управлять работой демона **named** без ключа. Это кажется чрезмерным требованием, но подумайте вот о чем: даже если команда **rndc** работает только на узле **localhost** (127.0.0.1) и его адрес заблокирован для внешнего мира на брандмауэре, вы все равно выдаете всем локальным пользователям определенный кредит доверия, подразумевая, что они не будут делать ничего противозаконного с сервером имен. В то же время любой из них может подключиться с помощью утилиты **telnet** к управляющему порту и ввести "stop".

Ниже показаны результаты работы команды **rndc-confgen**, генерирующей 256-разрядный ключ (размер ключа объясняется тем, что он позволяет уместить результаты на странице!). Обычно вывод команды направляется не на экран, а в файл **/etc/rndc.conf**. Комментарии в нижней части включают строки, которые необходимо добавить в файл **named.conf**, чтобы демон **named** и команда **rndc** могли взаимодействовать.

```
% ./rndc-confgen -b 256
# Start of rndc.conf
key "rndc-key" {
     algorithm hmac-md5;
     secret "orZuz5amkUnEp52zlHxD6cd5hAC1dOGsG/elP/dv2IY=";
};
```

```
options {
   default-key "rndc-key";
   default-server 127.0.0.1;
   default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list
# as needed:
# key "rndc-key" {
#    algorithm hmac-md5;
#    secret "orZuz5amkUnEp52zlHxD6cd5hAC1dOGsG/elP/dv2IY=";
# };
#
# controls {
#    inet 127.0.0.1 port 953
#       allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

## Расщепление DNS и инструкция view

Во многих организациях требуется, чтобы внутреннее представление сети отличалось от представления этой сети с точки зрения пользователей Интернета. Например, внутренние пользователи могут видеть все компьютеры зоны и лишь несколько разрешенных внешних серверов. Или другой вариант: внутреннее и внешнее представления одинаковы по охвату узлов, однако внутренним пользователям предоставляются дополнительные (либо отличающиеся) записи. В частности, записи MX, предназначенные для маршрутизации электронной почты, за пределами домена ссылаются на почтовый концентратор, а внутри домена — на отдельные рабочие станции.

О частных адресных областях рассказывалось в разделе 14.4.

Расщепленная конфигурация DNS особенно удобна организациям, которые во внутренних сетях применяют IP-адреса из частного диапазона (RFC1918). Например, запрос об имени узла с IP-адресом 10.0.0.1 не может быть обработан глобальной системой доменных имен, но он вполне допустим в контексте локальной сети. Среди запросов, поступающих на корневые серверы имен, 4-5% либо имеют исходный IP-адрес из частного диапазона, либо ссылаются на такой адрес. Ни на один из этих запросов нельзя ответить. Запросы обоих типов являются следствием неправильной конфигурации либо пакета BIND, либо "доменов" Microsoft.

Инструкция `view` содержит список адресов, определяющий, кто из клиентов имеет доступ к данному представлению, а также ряд параметров, применимых ко всем зонам в представлении, и, наконец, определения самих зон. Синтаксис инструкции имеет следующий вид.

```
view имя_представления {
   match-clients { список_соответствия_адресов };          [все узлы]
   match-destinations { список_соответствия_адресов };     [все узлы]
   match-recursive-only yes | no;                          [no]
   параметр_представления;
   инструкция_zone; ...
}
```

Инструкция `view` всегда содержит атрибут `match-clients`, который фильтрует IP-адреса источников, указанные в пакете запроса, и обычно используется для обслуживания внутренних и внешних представлений данных DNS, принадлежащих организации. Для более качественного контроля можно также фильтровать адреса пунктов назначения и требовать рекурсивные запросы. Атрибут `match-destinations` просматривает адреса назначения в пакете запроса и является полезным в компьютерах с несколькими интерфейсами, когда вы хотите обрабатывать разные данные DNS в зависимости от интерфейса, из которого поступил запрос. Атрибут `match-recursive-only` требует, чтобы запросы были рекурсивными, а также высылались разрешенными клиентами. Итеративные запросы позволяют просмотреть кеш сайта; этот атрибут предотвращает такие ситуации.

Представления просматриваются по порядку, поэтому инструкции `view` с самыми большими ограничениями доступа должны идти впереди. Зоны в разных представлениях могут иметь одинаковые имена. Представления налагают ограничение на структуру файла **named.conf**: если они присутствуют, *все* инструкции `zone` должны находиться в контексте представлений.

Ниже приводится взятый из документации к BIND 9 пример, имитирующий разделение пространства DNS-имен на внутреннее и внешнее. В обоих представлениях задается одна и та же зона, но с разной базой данных.

```
view "internal" {
    match-clients { наши_сети; };     // только внутренние сети
    recursion yes;                    // только для внутренних клиентов
    zone "example.com" {              // полное представление зоны
        type master;
        file "example-internal.db";
    };
};

view "external" {
    match-clients { any; };           // допускаются любые запросы
    recursion no;                     // рекурсия запрещена
    zone "example.com" {              // только "общедоступные" узлы
        type master;
        file "example-external.db";
    };
};
```

Если поменять порядок представлений, никто не сможет получить доступ к внутреннему представлению. Адреса внутренних узлов пройдут проверку на соответствие условию `any` в предложении `match-clients` внешнего представления до того, как начнет проверяться внутреннее представление.

Второй пример представления DNS, который будет рассмотрен ниже, демонстрирует еще один вид представлений.

# 17.10. Примеры конфигурации системы BIND

После подробного знакомства с файлом **named.conf** настало время рассмотреть ряд готовых примеров. В следующих подразделах мы опишем три типичные конфигурации:

- зона локального узла;

- сервер небольшой компании, занимающейся вопросами безопасности и применяющей расщепленную конфигурацию DNS;
- эксперты: сайт `isc.org` консорциума Internet System Consortium.

## Зона локального узла

Адрес 127.0.0.1 относится к самому локальному узлу и должен быть преобразован в имя "`localhost.`"[21] Одни организации преобразовывают адрес в имя "`localhost.`*локальный_домен*", а другие делают и то, и другое. Соответствующий адрес в протоколе IPv6 равен ::1.

Если вы забыли задать конфигурацию зоны локального узла, то ваша организация может прекратить посылать запросы корневым серверам на получение информации о локальном узле. Корневые серверы получают так много таких запросов, что операторы предлагают просто добавить обобщенное отображение между локальным узлом и адресом 127.0.0.1. на корневом уровне. По данным о корневом сервере K, в Европе в январе 2010 года (`k.root-servers.org/statistics`) запрос домена "local" занимает четвертое место по распространенности, отставая лишь от `com`, `arpa` и `net`. Это огромное количество бесполезных запросов (1500 в секунду), которые посылаются занятым серверам имен. Другими необычными именами в популярной категории "фиктивные домены TLD" являются `lan`, `home`, `localdomain` и `domain`.

Прямое преобразование имени локального узла можно определить в файле зоны прямого преобразования (forward zone) для домена (с помощью соответствующей инструкции `$ORIGIN`) или в своем собственном файле. Каждый сервер, даже кеширующий, обычно является главным для своего собственного обратного домена локального узла.

Ниже приведено несколько строк из файла **`named.conf`**, описывающих конфигурацию локального узла.

```
zone "localhost" { // зона прямого преобразования локального узла
    type master;
    file "localhost";
    allow-update { none; };
};

zone "0.0.127.in-addr.arpa" {    // зона обратного преобразования
                                 //  локального узла
    type master;
    file "127.0.0";
    allow-update { none; };
};
```

Соответствующий файл прямой зоны `localhost` содержит следующие строки.

```
$TTL 30d
; localhost.
@               IN    SOA   localhost. postmaster.localhost. (
                                1998050801      ;регистрационный номер
                                3600            ;тайм-аут обновления
                                1800            ;тайм-аут повторения
                                604800          ;срок действия
                                3600 )          ;минимум
```

[21] На самом деле к локальному узлу относится весь класс сетей A 127/8, но большинство использует просто адрес 127.0.0.1.

```
                NS    localhost.
                A     127.0.0.1
```

Обратный файл 127.0.0 имеет следующий вид.

```
$TTL 30d
; 0.0.127.in-addr.arpa
@               IN    SOA   localhost. postmaster.localhost. (
                                1998050801          ;регистрационный номер
                                3600                ;тайм-аут обновления
                                1800                ;тайм-аут повторения
                                604800              ;срок действия
                                3600 )              ;минимальный тайм-аут
                NS    local host.
1               PTR   localhost.
```

Отображение для адреса локального узла (127.0.0.1) никогда не изменяется, поэтому тайм-ауты могут быть большими. Обратите внимание на регистрационный номер , который кодирует дату; данный файл последний раз изменялся в 1998 году. Кроме того, обратите внимание на то, что для домена локального узла указан только главный сервер имен. Символ @ здесь означает "0.0.127.in-addr.arpa.".

## Небольшая компания, предоставляющая консалтинговые услуги в области безопасности

Наш следующий пример посвящен небольшой компании, специализирующейся на предоставлении консультаций по компьютерной безопасности. На ее сервере Red Hat Enterprise Linux установлен пакет BIND 9 и применяется расщепленная конфигурация DNS, в которой внутренние и внешние пользователи получают разную информацию. В компании используются адреса из частного диапазона. Запросы, касающиеся этих адресов, не должны выходить в Интернет и засорять глобальную систему доменных имен. Ниже приведен соответствующий файл **named.conf**, слегка переформатированный и снабженный комментариями.

```
options {
     directory "/var/domain";
     version "root@atrust.com";
     allow-transfer {82.165.230.84; 71.33.249.193; 127.0.0.1;};
     listen-on { 192.168.2.10; 192.168.2.1; 127.0.0.1; 192.168.2.12;}
};

include "atrust.key"; // файл с режимом доступа 600, содержащий
                      // определение ключа "atkey"
controls {
    inet 127.0.0.1 allow { 127.0.0.1; } keys { atkey; };
};

view "internal" {        // внутреннее представление

    match-clients { 192.168.1.0/24; 192.168.2.0/24; };
    recursion yes;

    include "infrustructure.zones"; // Корневые подсказки, прямое и обратное
                                    // преобразование адреса локального узла
```

```
    zone "atrust.com" {              // Внутренняя зона
                                     // прямого преобразования
                                     // узла localhost
       type master;
       file internal/atrust.com";
    };

    zone "1.168.192.in-addr.arpa" { // внутренняя зона
                                     // обратного преобразования
       type master;
       file "internal/192.168.1.rev";
       allow-update { none; };
    };
   ...                                // Тут пропущено много других зон
include "internal/trademark.zones"l   // atrust.net, atrust.org и другие
                                     // подчиненные серверы
 };                                  // конец внутреннего представления

view "world" {        // внешнее представление

    match-clients { any; };
    recursion no;

    zone "atrust.com" {              // внешняя зона прямого преобразования
        type master;
        file "world/atrust.com";
        a;;ow-update { none; };
    };

    zone "189.172.63.in-addr.arpa" {  // внешняя зона
                                      // обратного преобразования
        type master;
        file "world/63.173.189.rev";
        allow-update { none; };
    };
    include "world/trademark.zones";    // atrust.net, atrust.org и другие
                                        // подчиненные серверы
    zone "admin.com" {                  // Главные зоны только
                                        // во внешнем представлении
       type master;
       file "worls/admin.com";
       allow-update [ none; ] ;

   };
   ...                                  // Тут пропущено много главных и
                                        // подчиненных зон
// конец внешнего представления
```

Файл **atrust.key** содержит определение ключа “atkey”.

```
key "atkey" {
   algorithm hmac-md5;
   secret "совместный секретный ключ";
};
```

Конфигурационный файл **infrastructure.zones** содержит корневые подсказки и файлы локального узла, а файл **trademark.zones** — варианты имени atrust.com как

в разных доменах верхнего уровня (net, org, us, info и т.д.), так и с разным написанием (appliedtrust.com и т.д.).

Зонные файлы определяют два разных представления (внутреннее и внешнее) и тип сервера (главный или подчиненный). Это отображается в соглашении об именах файлов зон. Во внутреннем представлении данный сервер является рекурсивным. Это представление содержит все локальные узлы, многие из которых используют частные адреса. Во внешнем представлении этот сервер не является рекурсивным. Это представление содержит только выбранные узлы домена atrust.com и внешние зоны, для которых они обеспечивают DNS-услуги главных или подчиненных серверов.

Ниже приведены фрагменты файлов **internal/atrust.com** и **world/atrust.com**. Сначала рассмотрим файл **internal**.

```
; файл atrust.com - внутренний файл

$TTL 86400
$ORIGIN com.
atrust              3600    SOA  ns1.atrust.com. trent.atrust.com. (
                                   2001110500 10800 1200 3600000 3600 )
                    3600    NS   NS1.atrust.com.
                    3600    NS   NS2.artust.com.
                    3600    MX   10 mailserver.atrust.com.
                    3600    A    66.77.22.161

$ORIGIN atrust.com.
ns1                         A    192.168.2.1
ns2                         A    66.77.122.161
www                         A    66.77.122.161
mailserver                  A    192.168.2.11
exchange                    A    192.168.1.100
secure                      A    66.77.122.161
```

Здесь использованы частные адреса RFC1918. Кроме того, обратите внимание на то, что, вместо записей CNAME, для определения имен локального узла эта организация использует несколько записей A. Эта схема работает быстрее, поскольку она не учитывает результат проверки записи CNAME в дополнительном запросе. Записи PTR должны ссылаться только на одно из многих имен, которые отображаются в один и тот же IP-адрес. Эта организация также делегирует поддомены для своих DHCP-сетей, нашей лаборатории и своей Microsoft-инфраструктуры (это во фрагментах не показано).

Рассмотрим текст внешнего представления той же зоны atrust.com из файла **world/atrust.com**.

```
; файл atrust.com - внешний файл

$TTL 57600
$ORIGIN atrust.com.
@              3600    SOA  ns1.atrust.com. trent.atrust.com. (
                                2010030400 10800 1200 3600000 3600 )
                       3600  NS   NS1.atrust.com.
                       3600  NS   NS2.atrusr.com.
                       3600  MX   10 mailserver.atrust.com.
                       3600  A    66.77.122.161

ns1                          A    206.168.198.209
ns2                          A    66.77.122.161
www                          A    69.77.122.161
```

```
mailserver              A     206.168.198.209
secure                  A     66.77.122.161

; обратные преобразования
exterior1          IN   A     206.168.198.209
209.198.168.206    IN   PTR   exterior1.atrust.com.
exterior2          IN   A     206.168.198.213
213.198.168.206    IN   PTR   exterior2.atrust.com.
```

Как и во внутреннем преобразовании, имена реализованы с помощью записей A. Очень немногие узлы на самом деле видны во внешнем представлении, хотя из этих фрагментов это не следует. Обратите внимание на то, что компьютеры, которые видны в обоих представлениях (например, `ns1.atrust.com`), во внутреннем представлении имеют частные адреса RFC1918, а во внешнем — реальные IP-адреса.

Параметр TTL в зонных файлах по умолчанию установлен равным 16 часам (57 600 секунд). В то же время для большинства записей зонных файлов значение TTL не назначается явно.

Загадочные записи PTR в конце файла внешнего представления позволяют провайдеру компании `atrust.com` делегировать ей полномочия по управлению обратными преобразованиями в очень маленьких областях адресного пространства. Это делается с помощью записей CNAME на узле провайдера. Подробнее о специальном применении записей CNAME рассказывается в разделе 17.8.

## Консорциум The Internet Systems Consortium (isc.org)

Консорциум ISC — создатель и распространитель системы BIND, а также оператор корневого сервера E. Он также владеет сервером TLD, обслуживающим многие домены верхнего уровня. Именно поэтому мы называем его экспертом!

Ниже приведены фрагменты конфигурационных файлов. Обратите внимание на то, что они используют оба протокола IPv4 и IPv6. Кроме того, они используют шифрование TSIG для аутентификации главного и подчиненного серверов при передаче зон. Параметры `transfer-source` гарантируют, что IP-адреса источника для исходящей зоны передачи соответствуют требованиям спецификаций, установленных инструкциями `allow-transfers` на главном сервере.

Файл **`named.conf`** имеет следующее содержание.

```
// TLD-сервер имен организации isc.org

options {
    directory "/var/named";
    datasize 1000M;
    listen-on { 204.152.184.64; };
    listen-on-v6 { 2001:4f8:0:2::13; };
    recursion no;
    transfer-source 204.152.184.64;
    transfer-source-v6 2001:4f8:0:2::13;
};

// ключ rndc
key rndc_key {
   algorithm hmac-md5;
   secret "<secret>";
};
```

```
// Ключ TSIG для сервера имен ns-ext
key ns-ext {
   algorithm hmac-md5;
   secret "<secret>";
};

server 204.152.188.234 { keys { ns-ext; }; };

controls {
   inet 204.152.184.64 allow { any; } keys { rndc_key; };
};

include "inf/named.zones";  // Корневой, локальный, 127.0.0.1, ::1
include "master.zones";     // Управляемые зоны
include "slave.zones";      // Подчиненные зоны
```

Инструкции `include` позволяют сохранить файл `named.conf` коротким и аккуратным. Если вы обслуживаете много зон, постарайтесь разбить свою конфигурацию на небольшие части, похожие на эти. И, что еще более важно, организуйте иерархию ваших файлов так, чтобы в ней не было каталога, содержащего тысячу зонных файлов. Современные файловые системы эффективно обрабатывают многочисленные каталоги, но управление этими файлами может быть очень утомительным.

Рассмотрим содержимое файла `master.zones`.

```
zone "isc.org" {
    type master;
    file "master/isc.org";
    allow-update { none; };
    allow-transfer { none; };
};

zone "sfo2.isc.org" {
    type master;
    file "master/sfo2.isc.org";
    allow-update { none; };
    allow-transfer { none; };
};

// Остальные зоны пропущены
Содержимое файла slaves.zones выглядит так.
zone "vix.com" {
    type slave;
    file "secondary/vix.com";
    masters { 204.152.188.234; };
};

zone "cix.net" {
    type slave;
    file "secondary/cix.net";
    masters { 204.152.188.234; };
};
```

Параметр `allow-transfer`, заданный в файле **`master.zones`** равным `none`, означает, что консорциум ISC использует несколько главных серверов — какой-то из них должен реализовать передачу зоны подчиненным серверам.

# 17.11. Программное обеспечение NSD/Unbound

Завершив рассмотрение системы BIND, перейдем к описанию альтернативной реализации сервера DNS, обладающей, наряду с высокой производительностью, еще рядом прекрасных свойств. Система NSD (Name Server Daemon) была разработана организацией NLnet Labs в 2003 году. Изначально цель проекта заключалась в том, чтобы разработать альтернативную реализацию сервера, независимую от системы BIND, которую можно было бы использовать на корневых серверах, обеспечивая более высокую устойчивость корневой зоны за счет разнообразия программного обеспечения. В настоящее время систему NSD используют три корневых сервера и несколько доменов верхнего уровня, но вам не обязательно управлять корневым сервером или доменом TLD, чтобы оценить надежность, производительность и простоту этой реализации.

Ядро пакета NSD образуют две программы: **zonec** (прекомпилятор зонных файлов, преобразующий текстовые зонные файлы системы DNS в базу данных) и **nsd** (собственно демон сервера имен). Сервер NSD заранее вычисляет и индексирует все возможные ответы на корректные запросы, которые он может получить, поэтому, в отличие от системы BIND, создающей эти ответы на лету, система NSD уже имеет эти ответы в исходящем пакете в единственном экземпляре, хранящемся в памяти, что делает ее поразительно быстрой.

Unbound — рекурсивный сервер DNS, дополняющий систему NSD. Он был разработан на языке C организацией NLnet Labs на основе реализации, выполненной на языке Java компаниями VeriSign, Nominet, Kirei и EP.NET. Сочетание систем NSD и Unbound создает гибкую, высокопроизводительную и безопасную DNS-службу, подходящую для большинства организаций. Компоненты, разработанные организацией NLnet Labs, не такие зрелые, как система BIND, и не имеют так много "прибамбасов", но для большинства сайтов они могут стать прекрасным решением.

Библиотека стандартных программ **ldns**, упрощающая написание инструментов для программного обеспечения DNS, также доступна для использования в дистрибутивных пакетах NSD и Unbound. Она содержит каталог примеров: несколько инструментов предназначены, в основном, для спецификаций DNSSEC, в частности программа для создания подписи DNSSEC и утилита **drill**, предназначенная для отладки и аналогичная утилите **dig** из пакета BIND. Все эти программы можно загрузить с сайта `nlnetlabs.nl`. Организация NLnet Labs также создала инструмент под названием Autotrust, обеспечивающий управление ключом RFC5011 и продление его срока действия. Эта программа интегрирована в систему Unbound, однако мы ее описывать не будем.

Код DNSSEC в системе NSD/Unbound более надежен и лучше протестирован, чем аналогичный код в системе BIND. Кроме того, он быстрее работает. Например, система Unbound примерно в пять раз быстрее, чем система BIND, проверяет подписи DNSSEC. Тем не менее в некоторых аспектах, особенно что касается документации и дополнительных функциональных возможностей, система BIND по-прежнему занимает лидирующие позиции. Для того чтобы обеспечить действительно устойчивую систему DNS, лучше запустить оба сервера!

## Инсталляция и конфигурирование системы NSD

Вначале создайте пользователя с именем nsd в системе, в которой будет выполняться сервер имен **nsd**. Затем зарегистрируйте его как nsd, загрузите пакеты NSD/Unbound (в настоящее время их три: NSD, Unbound и отдельная библиотека **ldns**) и распакуйте

их. Для того чтобы инсталлировать демон **nsd**, следуйте инструкциям, написанным в файле **doc/README**.

```
$ ./configure
$ make
$ sudo make install
```

В табл. 17.10 перечислены каталоги, в которые система NSD инсталлируется по умолчанию.

**Таблица 17.10. Каталоги для инсталляции системы NSD**

| Элемент | Каталог |
|---|---|
| Выполняемые файлы | **/usr/local/sbin** |
| Пример файла конфигурации | **/etc/nsd** |
| Справочные страницы | **/usr/local/share** |
| Текстовые зонные файлы | **/etc/nsd** |
| Скомпилированные зонные файлы | **/var/db/nsd** |
| Файлы PID | **/var/run** |

Иногда размещать зонные файлы в каталоге **etc** довольно неудобно, особенно если зона большая. В таком случае следует выбрать другое место, например каталог **/usr/local** или **/var**. Если вас не устраивает выбор системы NSD, предусмотренный по умолчанию, вы можете использовать команду **configure**; сборочный файл можно прочитать без проблем. Результаты, возвращаемые командой **configure**, записываются в файл **config.log** в каталоге инсталляции, поэтому при необходимости его можно проанализировать.

Пакет NSD инсталлирует семь программ.

- **nsd** — демон сервера имен
- **nsdc** — сценарий, управляющий демоном **nsd**, посылая ему сигналы
- **zonec** — преобразовывает текстовые зонные файлы в файлы базы данных
- **nsd-notify** — посылает уведомления (исключен)
- **nsd-xfer** — получает передачи зоны (исключен)
- **nsd-checkconf** — проверяет синтаксис файла **nsd.conf**
- **nsd-patch** — отражает постепенные обновления базы данных в зонных файлах

### *Фундаментальные отличия от системы BIND*

Если вы использовали систему BIND, то кое-что в системе NSD сначала покажется вам странным. Например, в ней нет файлов корневых подсказок и нет необходимости включать зоны локальных узлов. Кроме того, система NSD не поддерживает представления, поэтому если ваша организация публикует разные варианты данных DNS внутри и вне, то вам придется вернуться к системе BIND или использовать несколько экземпляров демона **nsd**. Динамические обновления также не поддерживаются. Поскольку **nsd** — это сервер, который может быть только авторитетным, многие возможности системы BIND для него недоступны.

Система BIND считывает зонные файлы и хранит их в памяти; система NSD заранее компилирует зонные файлы в формат базы данных и использует для ее хранения и память, и диск.

Язык описания конфигурации демона **nsd** проще, чем в системе BIND. В нем нет точек с запятой, которые можно забыть, нет фигурных скобок для объединения групп и есть только три инструкции верхнего уровня: `server`, `zone` и `key`. Комментарии отмечаются символом #. Параметры каждой из трех инструкций имеют следующий вид.

*атрибут*: *значение*

Инструкция `server` может существовать только в одном экземпляре. Она задает глобальные параметры. Инструкция `zone` перечисляет параметры зон, а инструкция `key` определяет криптографические ключи, необходимые для взаимодействия между главным и подчиненными серверами, а также для управления демоном **nsd**. Пробелы отделяют атрибуты от значений. Значения можно взять в кавычки, но это необязательно.

Как и система BIND, демон **nsd** обобщает IP-адреса, но несколько иначе, чем система BIND, использующая список соответствий адресов. Этот список в системе NSD называется *ip-spec* и может выглядеть следующим образом.

- Обычный IP-адрес (IPv4 или IPv6).
- Подсеть в системе обозначений CIDR, например `1.2.3.0/24`.
- Подсеть с явной маской, например `1.2.3.4&255.255.255.0`.
- Диапазон IP-адресов, например `1/2/3/4-1.2.3.25`.

В каждой из этих форм пробелы не допускаются. .

Еще одним основным отличием является использование значений ключа для аутентификации передачи зон и уведомлений. В системе BIND ключ ассоциируется с IP-адресом, и сообщения, поступающие из этого адреса и на этот адрес, подписываются и проверяются с помощью этого ключа. В системе NSD принята более точная классификация ключей, и демон **nsd** может использовать один ключ для уведомлений, другой — для передачи обновлений зоны, а третий — для получения этих обновлений. Полезно? Ну, как сказать...

Семантика команды `notify` демона **nsd** аналогична разделу `notify explicit` в системе BIND: уведомление отправляется только серверам, которые включены в список. (По умолчанию система BIND уведомляет все серверы имен, указанные в зонном файле целевого домена.)

Демон **nsd** предпочитает осмысленное описание поведения по умолчанию с помощью конфигурации. Например, спецификации DNSSEC включаются по умолчанию для подписанных зон и выключаются для неподписанных. По умолчанию демон **nsd** прослушивает сокеты как по протоколу IPv4, так и по протоколу IPv6. Регистрация либо включена, либо выключена и не предусматривает бесчисленное количество категорий и градаций, как это принято в парадигме BIND. Для взаимодействия между серверами (передачи зон, уведомлений и т.д.) демон **nsd** использует протокол TSIG.

Краткое описание различий между системами BIND и NSD, а также пример конфигурации содержатся в файле **doc/NSD-FOR-BIND-USERS**. Справочная страница для файла конфигурации **nsd.conf** и файл **doc/README** также содержат примеры, но, к сожалению, они не согласованы друг с другом. [22]

## Пример конфигурации системы NSD

Мы приобрели несколько образовательных лицензий и объединили три примера из дистрибутивов и добавили собственные комментарии, чтобы читатели могли получить

[22] Для того чтобы открыть справочную страницу, не инсталлируя ее, следует выполнить команду ***groff -man -T ascii*** *имя-файла-справочной-страницы* **| less**.

представление о конфигурации системы NSD еще до того, как мы приступим к детальному разбору разных параметров. Рассмотрим демон **nsd**, сконфигурированный для главного сервера домена atrust.com и подчиненного сервера домена admin.com.

```
server:
     username: nsd                  # Пользователь nsd должен быть запущен
                                    # после выполнения операции chroot
     database: /var/db/nsd/nsd.db   # Заранее скомпилированный
                                    # файл базы данных
     logfile: /var/log/nsd.log      # Системный журнал, по умолчанию
                                    # направляется в потоки stderr и syslog
     pidfile: /var/run/nsd.pid      # Идентификатор процесса nsd
key:
     name: tsig.atrust.com.         # Ключ для передачи зоны atrust..com
     algorithm: hmac-md5
     secret: "здесь содержится пароль, зашифрованный по методу base64"
zone:
     name: atrust.com                           # Имя зоны
     zonefile: /var/nsd/primary/atrust.com
     provide-xfr: 1.2.3.4 tsig.atrust.com.      # Адрес подчиненного сервера
                                                # зоны atrust.com
     notify: 1.2.3.4 tsig.atrust.com            # и ключ для уведомления или
                                                # xfr-файлов.
     provide-xfr: 1.2.30.40 tsig.atrust.com.    # Адрес другого подчиненного
                                                # сервера
     notify: 1.2.30.40 tsig.atrust.com          # и ключ для уведомления или
                                                # xfr-файлов.
key:
     name: tsig.admin.com.                  # Ключ для получения данных от
                                            # admin.com подчиненным сервером
     algorithm: hmac-md5
     secret: "здесь содержится пароль, зашифрованный по методу base64"
zone:
     name: admin.com                            # Зона, по отношению к которой
                                                # данный сервер является
                                                # подчиненным
     zonefile: "/var/nsd/secondary/admin.com.signed"
     allow-notify: 5.6.7.8 NOKEY                # Ее главная зона
     request-xfr: 5.6.7.8 tsig.admin.com.       # и ключ для xfr-файлов
```

В этом примере файла **nsd.conf** описана конфигурация сервера, который является главным для зоны atrust.com и осуществляет уведомление и передачу зоны двум подчиненным серверам, имеющим IP-адреса 1.2.3.4 и 1.2.30.40. Для аутентификации двух подчиненных серверов при передаче зон и уведомлении этот сервер использует ключ TSIG с именем tsig.atrust.com. (Вы можете и, возможно, даже обязаны использовать для каждого подчиненного сервера отдельный ключ.)

Этот сервер также является подчиненным сервером для зоны `admin.com`, главный сервер которой имеет IP-адрес `5.6.7.8`. Мы можем получать уведомления об изменении данных зоны без ключа, но при передаче зоны обязаны использовать ключ **tsig.admin.com**. Главные зоны имеют разделы `provide-xfr`, а подчиненные — раздел `request-xfr`.

После инсталлирования демона **nsd** и оформления файла конфигурации, для проверки синтаксиса файлов конфигурации следует применить команду **nsd-checkconf**. Его отчеты об ошибках довольно лаконичны, например "**error: syntax error**" и номер строки. Исправив ошибки, выявленные командой **nsd-checkconf**, выполните ее еще

раз и повторяйте этот процесс, пока команда **nsd-checkconf** не перестанет выдавать сообщения об ошибках.

### Определения ключа NSD

Раздел `key` определяет именной ключ, который можно использовать для последующего доступа к параметрам управления. Каждый ключ имеет три атрибута: имя, алгоритм и разделенный секрет (т.е. пароль). Посмотрим, как разместить определение ключа или, по крайней мере, его разделенный секрет в файлах, доступ к которым ограничен. Для их импорта в файл **nsd.conf** можем использовать инструкцию `include`. Достаточно просто поместить в файл строку `include:` *имя_файла* в то место, куда мы хотим вставить текст из файла с именем *имя_файла*.

🕮 Определения ключа

Синтаксис инструкции `key` выглядит следующим образом.

```
name: имя_ключа
algorithm: название_алгоритма
secret: пароль в кодировке base64
```

Допускается несколько инструкций `key`.

Поле `name` идентифицирует ключ. Выбирайте имена, идентифицирующие зону и серверы, участвующие в обменах секретной информацией. Можно использовать алгоритмы `hmac-md5`, `hmac-sha1` или `hmac-sha256`. Можно также сгенерировать ключи TSIG с помощью команды **ldns-keygen**, несмотря на то что на самом деле она предназначена для генерирования пар закрытых и открытых ключей DNSSEC. Для получения списка алгоритмов можно выполнить команду **ldns-keygen -a list**.

Рассмотрим пример использования алгоритма `hmac-sha1`.

```
$ ldns-keygen -a hmac-sha1 example.com
```

Эта команда создает файл с именем **Kexample.com.+158+*12345*.key**, содержащий ключ TSIG. Достаточно просто вырезать и скопировать секретную часть в вашу спецификацию ключа. Число **158** в имени файла означает алгоритм `hmac-sha1`. Для алгоритма `hmac-md5` зарезервировано число **157**, а для алгоритма `hmac-sha256` — **159**. Число *12345* — это просто заполнитель для случайного пятизначного дескриптора ключа. При использовании нескольких ключей эти дескрипторы помогают правильно с ними обходиться.

### Глобальные параметры конфигурации NSD

Параметры NSD подразделяются на две группы: глобальные параметры сервера и зонные параметры, которые можно применить к любой зоне, обслуживаемой конкретным экземпляром демона **nsd**. Некоторые параметры можно заменять с помощью флагов командной строки при запуске демона **nsd**.

Параметры сервера обычно имеют разумные значения, установленные по умолчанию, и требуют внимания, только если ваша структура каталогов отличается от стандартной или вы хотите сделать нечто необычное. Ниже мы приводим значения по умолчанию в квадратных скобках. Как и для параметров системы BIND, мы добавили примечания, описывающие каждую группу параметров, чтобы в них было легче разобраться.

🕮 Включение файла

```
include: имя_файла
```

Директива include: может находиться в любом месте файла конфигурации. Файл с именем *имя_файла* считывается в файл конфигурации, и его содержимое заменяет директиву.

🕮 IP-адреса и порт

```
ip-address: ip_адрес     [все IP_адреса]
ip4-only: yes | no       [no]
ip6-only: yes | no       [no]
port: номер_порта        [53]
```

По умолчанию система NSD связывает порт 53 со всеми сетевыми интерфейсами как по протоколу IPv4, так и по протоколу IPv6. При перечислении адресов с помощью директивы ip-address: демон **nsd** обходит таблицы маршрутизации ядра и гарантирует, что на запрос к одному из IP-адресов не будет выдан ответ от другого IP-адреса; этого требуют многие процедуры разрешения имен. Если ваш компьютер имеет только один сетевой интерфейс, то этот параметр бесполезен. Параметры ip4-only и ip6-only ограничивают демон **nsd** определенным протоколом, а параметр port устанавливает сетевой порт, прослушивающий входящие запросы.

🕮 Идентификаторы

Как правило, эти параметры не используются, потому что их значения по умолчанию подобраны очень удачно.

```
identity: строка                [имя_узла]
hide-version: yes | no           [no]
```

Эта пара параметров определяет, сообщать ли демону **nsd** правду о его версии и имени узла в ответ на запросы для имен классов CHAOS id.server и version.server. Как указывалось выше, мы рекомендуем не изменять эти значения.

🕮 Протоколирование, статистические показатели

```
logfile: имя_файла          [stderr и stdlog]
verbosity: уровень          [0]
debug-mode: yes | no        [no]
statistics: #сек            [0] # без статистических показателей
```

Протоколирование по умолчанию осуществляется в стандартном потоке ошибок и системном журнале (функциональная возможность демона), причем объем протоколирования определяется параметром verbosity. Диапазон возможных значений варьируется от 0 до 5; большие значения означают, что следует регистрировать больше данных. Параметр logfile направляет протокольные сообщения в файл, а не в стандартный системный журнал.

Если задан параметр debug-mode, то демон **nsd** не создает дополнительные копии самого себя и остается присоединенным к вашему терминалу, так что вы можете видеть сообщения, посланные в стандартный поток ошибок.

Если вы хотите накапливать статистические показатели, то установите параметр *#сек* равным количеству секунд между сбросами информации. Статистические показатели похожи на другие протокольные сообщения, поэтому они направляются в файл регистрации или системный журнал, если не задан другой конкретный файл. Лучше всего иногда просматривать статистические данные, чтобы убедиться в том, что интервал между сбросами информации выбран правильно и вы не перегружаете диски информацией, которую никто никогда не увидит.

🕮 Имена файлов

```
database: имя_файла     [/var/db/nsd/nsd.db]
difffile: имя_файла     [/var/db/nsd/ifxr.db]
xfrdfile: имя_файла     [/var/db/nsd/xfrd.state]
pid-file: имя_файла     [/зависит от ОС, обычно /var/run/nsd.pid]
zonesdir: каталог       [/etc/nsd]
```

Скомпилированные зонные файлы и информация о передаче зоны обычно хранятся в каталоге **/var/db/nsd**; эта установка изменяется очень редко. Файл PID должен находиться там, куда ваша операционная система записывает остальные файлы PID, обычно в каталоге **/var/run**. По умолчанию зонные файлы, редактируемые людьми, находятся в каталоге **/etc/nsd**, что кажется неудачным решением. Посмотрим, как переместить эти файлы в подкаталоги каталога **/var**, например в каталог **/var/nsd/zones**.

🕮 Настройка

```
tcp-count: целое_число      [0]
server-count: целое_число   [1]
xfrd-reload-timeout: #сек   [10]
```

Параметр `tcp-count` ограничивает количество одновременных TCP-соединений, которые сервер может использовать для передачи зоны. Если вы увидите сообщение "xfrd: max number of TC connections (10) reached" в системном журнале, значит, вы превысили максимально допустимое значение. Если это происходит часто, вы должны увеличить этот предел.

Параметр `server-count` задает количество запускаемых экземпляров демона **nsd**. При работе на многопроцессорных компьютерах у системного администратора может возникнуть желание увеличить это значение. Параметр `xfrd-reload-timeout` регулирует перезагрузку после передачи зоны, задавая количество секунд, которое должно пройти после предыдущей загрузки прежде, чем можно будет выполнить новую.

🕮 Безопасность

```
username: регистрационное_имя     [nsd]
chroot: каталог                   [нет]
```

Демон **nsd** должен запускаться как корневой процесс, чтобы открыть привилегированный сокет (порт 53), но потом он может потерять привилегии и вернуться на уровень обычного пользователя, поскольку все файлы, которые ему нужны, принадлежать данному пользователю. Следовательно, демон **nsd** должен иметь учетную запись.

Для того чтобы повысить безопасность, мы можем также запустить демон **nsd** в виртуальном окружении, используя команду **chroot**, поскольку зонные файлы, файлы баз данных, файлы **xfrdfile**, **difffile** и PID, а также сокет системного журнала (или регистрационный файл) доступны через виртуальный каталог.

### *Зонные параметры конфигурации NSD*

В отличие от глобальных параметров, зонные параметры обычно требуют определенного конфигурирования, особенно это относится к спискам управления доступом (ACL — access control list).

🕮 Определения зон

```
name: имя_зоны
zonefile: имя_файла
```

Зона определяется своим именем и файлом записей о ресурсах.

🕮 Главные ACL-серверы

```
notify: ip-адрес ( имя-ключа | NOKEY )[23]
provide-xfr: ip-спецификация ( имя-ключа | NOKEY | BLOCKED)
```

Главный сервер для зоны уведомляет свои подчиненные серверы об обновлении зоны. Затем, по запросам подчиненных серверов, он инициирует передачу зоны для пересылки модифицированных данных. Следовательно, зона, по отношению к которой сервер является главным, должна быть указана в списках `notify` и `provide-xfr`. Значения этих параметров обычно совпадают. Если не указан параметр `NOKEY`, то уведомления подписываются ключом, указанным в списке как *имя_ключа*.

Следует помнить, что, в отличие от демона **named**, демон **nsd** не уведомляет подчиненные серверы зоны автоматически. Вы должны явно указать их в списках `notify` и `provide-xfr`. Эти директивы могут повторяться.

🕮 Подчиненные ACL-серверы

```
allow-notify: ip-адрес ( имя-ключа | NOKEY | BLOCKED)[24]
request-xfr: [ AXFR | UDP] ip-адреса ( имя-ключа | NOKEY)
```

Подчиненный сервер зоны должен явно разрешить главному серверу посылать уведомления. Любые сообщения, полученные от серверов, не перечисленных в списке `allow-notify` (или помеченных как `BLOCKED`), игнорируются.

Раздел `request-xfr` создает запрос подчиненного сервера на передачу зоны от главного сервера по указанному в списке *ip-адресу* с помощью заданного ключа с именем *имя_ключа*. Если аргумент `AXFR` включен, то запрашиваться будут только AXFR-передачи (т.е. передачи всей зоны, в отличие от постепенных изменений). Аргумент `UDP` указывает, что запрос на передачу зоны должен быть послан с помощью UDP-транспорта, а не протокола TCP, заданного по умолчанию. Лучше всего все же использовать протокол TCP.

🕮 IP-адрес источника

```
outgouing-interface: ip-адрес
```

Эти списки управляют IP-адресами, используемыми подчиненными серверами для запроса на передачу зоны или главным сервером для посылки уведомлений. Эти адреса должны совпадать, иначе говоря, раздел управления доступом в конфигурации зоны главного сервера должен содержать те же адреса, что и соответствующий раздел в конфигурации зоны подчиненного сервера.

## Запуск демона nsd

Задав конфигурацию демона **nsd**, запустите программу **nsd-checkconf**, чтобы убедиться, что в вашем файле конфигурации **nsd.conf** нет синтаксических ошибок. Затем поместите ваши зонные файлы в соответствующий каталог (заданный в файле **nsd.conf**) и используйте управляющий сценарий **nsdc**, чтобы скомпилировать их в файлы базы данных. В заключение запустите сервер имен с помощью сценария **nsdc**.

```
$ sudo nsdc rebuild
$ sudo dsdc start
```

Проверьте сервер имен с помощью команд **dig** или **drill**. Если результаты вас удовлетворяют, добавьте в последовательность загрузки вашей операционной системы команду **nsdc start**, а в ее последовательность завершения работы — команду **nsdc**

[23] Здесь и в следующей строке скобки использованы лишь для того, чтобы показать группирование; на самом деле их использовать не надо.

[24] См. предыдущую сноску.

**stop**. Вы можете также настроить файл-расписание для службы Cron, чтобы раз в день выполнять команду **nscd patch** для обновления текстовых зонных файлов на основе файлов базы данных.

## Инсталляция и конфигурирование сервера Unbound

Unbound — это рекурсивный, кеширующий DNS-сервер имен, выполняющий авторизацию. Его разработчиком является та же самая организация, которая создала сервер NDS (NLnet Labs). Сначала он был предназначен для систем UNIX и Linux, но в настоящее время доступен и для системы Windows.

Для того чтобы инсталлировать его, создайте нового пользователя с именем "unbound", зарегистрируйте его как пользователя и загрузите дистрибутивный пакет с адреса `unbound.net`. Сервер Unbound требует наличия библиотек **ldns** и OpenSSL и может использовать библиотеку **libevent** (`monkey.org/~provos/libevent`), если она доступна. Как и сервер NSD, сервер Unbound сначала должен запускаться как корневой, а затем возвращаться на уровень пользователя со специальной учетной записью.

Для того чтобы создать сервер Unbound, следует выполнить следующие команды.

```
$ ./configure
$ make
$ sudo make install
```

Сервер Unbound сопровождается обширным пакетом тестов, которые можно запустить командой **make test**. Дистрибутивный пакет инсталлирует следующие выполняемые файлы.

- **unbound** — рекурсивный сервер имен.
- **unbound-checkconf** — инструмент для проверки синтаксиса файла **unbound.conf**.
- **unbound-control**, **unbound-control-setup** — инструменты для удаленного управления безопасностью.
- **unbound-host** — простой инструмент для запросов.

Команда **unbound-host** не является такой же многословной, как команды **dig** и **drill**.

В табл. 17.11 перечислены каталоги, в которые размещаются компоненты системы Unbound.

**Таблица 17.11. Каталоги для инсталляции системы Unbound**

| Элемент | Каталог |
|---|---|
| Выполняемые файлы | /usr/local/sbin |
| Библиотеки | /usr/local/lib |
| Файл конфигурации | /etc/local/etc/unbound/unbound.conf |
| Справочные страницы | /usr/local/share |
| Механизм виртуализации | /usr/local/etc/unbound |
| Файлы PID | /usr/local/etc/unbound |

Файл конфигурации сервера **unbound** под названием **unbound.conf** похож на файл конфигурации сервера **nsd**. Его синтаксис выглядит следующим образом.

```
атрибут: значение
```

Комментарии начинаются символом # и продолжаются до конца строки. Для проверки корректности файла конфигурации можно выполнить команду **unbiund-checkconf**.

Рассмотрим простой пример файла **unbound.conf**, адаптированный по сравнению со справочной страницей и содержащий несколько дополнительных комментариев.

```
server:
    directory: "/var/unbound/etc"
    username: unbound
    chroot: "/var/unbound"
    pidfile: "/var/run/unbound.pid"

root-hints: "root.cache"
    interface: 0.0.0.0                  # Прослушивает все интерфейсы IPv4
    interface: ::0                      # и все интерфейсы IPv6
    access-control: 10.0.0.0/8 allow    # Локальные частные сети
    access-control: 2001:DB8::/64 allow # Локальные сети IPv6
```

В этом примере сервер прослушивает все интерфейсы и допускает все запросы, поступающие от локальных IPv6-сетей и немаршрутизированных частных сетей 10. Он регистрируется в системном журнале с помощью функциональных возможностей демона (по умолчанию) и запускается в виртуальном окружении, созданном командой **chroot** от имени пользователя unbound.

Поскольку сервер **unbound** является рекурсивным, он имеет много параметров. Их количество (больше 70) составляет примерно половину количества параметров сервера BIND (больше 150). Мы опишем лишь некоторые из них. Полный обзор параметров можно найти в справочной системе и в документации, расположенной на сайте unbound.net.

Язык конфигурации сервера **unbound** предусматривает четыре директивы высшего уровня: `server`, `remote-control`, `stub-zone` и `forward-zone`.[25] Глобальные параметры находятся в разделе `server`. Рассмотрим наиболее важные из них.

🕮 Местоположение

```
directory: каталог                  [/usr/local/etc/unbound]
pidfile: имя_файла                  [/usr/local/etc/unbound/unbound.pid]
root-hints: имя_файла               [нет]
```

Параметр `directory` задает рабочий каталог сервера: по умолчанию сервер **unbound** устанавливается в каталог **/usr/local/etc**, но многие организации предпочитают каталог **/var**. Файлы PID по умолчанию записываются в рабочий каталог сервера **unbound**, но их можно поместить и в более традиционные места, например в каталог **/var/run**.

Корневые подсказки встраиваются в код сервера **unbound**, поэтому файл подсказок не нужен. Однако его можно создать, если вы хотите, поскольку адреса, записанные в коде, в конце концов могут устареть. Если вы хотите иногда получать свежие копии сервера, используйте команду **dig . ns**. Если же вы страдаете паранойей, попытайтесь выполнить команду **dig@a.root-server.net . ns**, чтобы получить авторитетную копию.

🕮 Протоколирование

```
use-syslog: yes | no                [yes]
logfile: имя_файла                  [нет]
log-time-ascii: yes | no            [no]
verbosity: уровень                  [1]
```

[25] Он напоминает язык сценариев Python.

Протокольная информация может поступать либо в системный журнал, либо в файл. Это зависит от параметров `use-syslog` и `logfile`. Если вы хотите знать обычное время, а не время UNIX (т.е. количество секунд, прошедших после 1 января 1970 года), включите параметр `log-time-ascii`. Параметр `verbosity` определяет объем протоколирования (детали описаны в разделе 17.15).

🕮 Статистические показатели

```
statistics-interval: секунд          [0, т.е. отключена]
statistics-cumulative: yes | no      [no]
extended-statistics: yes | no        [no]
```

Накопление статистических данных по умолчанию отключено, потому что они замедляют работу сервера имен. Если вы включите этот режим, установив ненулевое значение параметра `statistics-interval`, то статистические данные будут записываться в протокольный файл (или системный журнал) через указанные промежутки времени. По умолчанию счетчики статистических показателей каждый раз при их записи в файл обнуляются; для того чтобы этого не происходило, используйте параметр `statistics-cumulative`.

Установив параметр `extended-statistics` равным `yes`, вы сгенерируете больше данных, чем сможете записать в файл с помощью команды **`unbound-control`**.

🕮 Более подробная информация об утилитах Cacti и RRDTool приведена в разделе 21.12.

Каталог дистрибутивов **`contrib`** содержит надстройки, которые соединяют утилиты Cacti или Munin с работающим сервером имен и выводят графическую информацию о статистических данных с помощью утилиты RRDTool. Подробности можно найти на сайте `unbound.net`.

🕮 Разрешение запросов

```
access-control: блок_сети действие          [разрешает любой локальный узел]
```

Параметр `access-control` — это ключ для конфигурации сервера имен в рекурсивном режиме по отношению к своим собственным пользователям, но не для остальных пользователей. Для того чтобы разрешить несколько интерфейсов, следует использовать несколько разделов `allow-control`. Параметр *действие* может принимать четыре значения:

- `deny` — блокирует все запросы от указанных сети или узла;
- `refuse` — блокирует запросы и посылает обратно сообщение REFUSED;
- `allow` — отвечает на запросы, поступившие от клиентов, требующих рекурсивной обработки;
- `allow-snoop` — отвечает на запросы от рекурсивных и итеративных клиентов.

Действие `refuse` более точно соответствует спецификации системы DNS, чем действие `deny`, потому что клиенты предполагают, что запросы, не получившие ответа, были потеряны в сети, а не отклонены по решению администратора. Действие `allow` применяется к обычным клиентам системы DNS.

Параметр `allow-snoop` позволяет отвечать не только на итеративные, но и на рекурсивные запросы. Его можно использовать для того, чтобы исследовать содержимое кеша сервера, поскольку итеративный запрос достигает цели, только если ответ уже находится в кеше. Параметр `allow-snoop` можно также использовать для злонамеренных действий, поэтому на узлах системного администратора его следует ограничивать.

🕮 Безопасность

```
chroot: каталог          [/usr/local/etc/unbound]
username: имя            [unbound]
```

Параметр `username` задает непривилегированного пользователя, под именем которого должен запускаться сервер **unbound**, после завершения процесса загрузки.

Директива `chroot` заставляет сервер **unbound** выполняться в виртуальном окружении. Для того чтобы убедиться, что серверу **unbound** доступно все, что ему нужно из его виртуального окружения, приходится преодолеть несколько препятствий, но в последних версиях код стал намного проще.

Код очень аккуратно отображает глобальные пути в виртуальное окружение, созданное командой **chroot** (chrooted jail). Большинство путей можно задать в виде абсолютных глобальных путей, абсолютных путей в виртуальном каталоге или относительных путей по отношению к рабочему каталогу. Сервер **unbound** при необходимости выполняет соответствующее преобразование.

Отметим несколько важных моментов, касающихся запуска сервера **unbound** в виртуальной среде. Считывание файла конфигурации, разумеется, предшествует выполнению команды **chroot**, поэтому файл конфигурации, заданный командной строкой сервера **unbound**, должен быть задан глобальным путем. Файл PID, содержащий ключ `unbound-control`, и сокет системного журнала остаются за пределами виртуального каталога, поскольку они открываются до того, как сервер **unbound** выполнит команду **chroot**.

Сервер **unbound** считывает псевдоустройство **/dev/random** до выполнения команды **chroot**, но мы рекомендуем оставить его открытым и после выполнения этой команды. Дело в том, что сервер **unbound** может обратиться к нему позднее, чтобы получить другие случайные данные. Если сервер **unbound** не может открыть псевдоустройство **/dev/random**, он использует источник случайных чисел, заданный по умолчанию, и регистрирует предупреждение.

В системе Linux можно сделать псевдоустройство **/dev/random** доступным в виртуальной среде, выполнив следующую манипуляцию (предполагается, что файл **/var/unbound** находится в виртуальном каталоге).

```
linux$ sudo mount -bind -n /dev/random /var/unbound/dev/random
```

### Идентификаторы

Следующие параметры определяют, сообщать ли серверу **unbound** правду о номере своей версии и имени узла при запросе имен CHAOS `id.server` и `version.server`.

```
hide-identity: yes | no        [no]
identity: строка              [имя_узла]
hide-version: yes | no         [no]
version: строка               [версия сервера имен]
```

Как указывалось в начале раздела, мы рекомендуем не скрывать эти значения.

### IP-адреса

```
interface: ip-адрес                          [локальный_узел]
outgoing-port-avoid: число или диапазон      [нет]
```

Сервер **unbound** имеет несколько параметров, управляющих интерфейсами, которые он прослушивает в ожидании запросов, и номерами портов, используемых для получения и отправки запросов. Для большинства организаций настройки по умолчанию вполне приемлемы и не уязвимы для атак на кеш, изобретенных Камински. Однако параметры `interface` и `outgoing-port-avoid` должны конфигурироваться явно.

Параметр `interface` задает интерфейсы, которые сервер **unbound** прослушивает в ожидании запросов. Его необходимо задавать явно, поскольку по умолчанию задан локальный узел, что вполне удовлетворительно, если сервер **unbound** выполняется на каждой машине, но неприемлемо, если в каждой подсети или на каждом сайте должен быть один сервер имен. Добавьте инструкцию `interface` для каждого интерфейса, которому клиенты могут послать запрос.

Необходимо также задать конфигурацию `outgoing-port-avoid`, чтобы исключить все порты, блокируемые вашим брандмауэром и используемые другой программой. Сервер **unbound** уже исключил порты, номера которых меньше 1024, и порты, назначенные организацией IANA.

🕮 Спецификации DNSSEC

```
module-config: имена_модулей          [нет]
trust-anchor-file: имя_файла          [нет]
trust-anchor: запись_о_ресурсах       [нет]
trusted-keys-file: имя_файла          [нет]
dlv-anchor-file: имя_файла            [нет]
dlv-anchor: запись_о_ресурсах         [нет]
```

Все эти параметры относятся к процессу развертывания спецификаций DNSSEC; они позволяют задать точки доверия, указав файлы, в которых они находятся, или поместив запись о ресурсах непосредственно в значение параметра. Допускается только одна выделенная точка доверия DLV.

Установив параметр `module-config` равным `validator iterator`, можно включить проверку спецификаций DNSSEC. Это значение должно сопровождаться точками доверия, которые должны быть либо явно сконфигурированы, либо выделены. Более подробно спецификации DNSSEC обсуждаются в разделе 17.13.

🕮 Подписи

```
val-* <разное>          [параметры подписи, приемлемые по умолчанию]
```

Ряд параметров `val-*` регламентирует процесс проверки подписи подписанных зон. На процесс проверки могут влиять разные факторы (например, максимально допустимое отклонение временных показателей). Значения, установленные по умолчанию, вполне приемлемы, если вы не хотите заняться отладкой процесса развертывания спецификаций DNSSEC. Задав параметр `val-log-level` равным 1, можно отключить проверку системного журнала, что при отладке бывает полезным.

🕮 Настройка

Сервер **unbound** имеет несколько параметров, позволяющих настраивать его работу. Одним из самых важных является параметр `num-threads`, который должен быть равным количеству ядер, доступных серверу (т.е. количеству ядер на один процессор, умноженному на количество процессоров).

Значения параметров настроек, заданные по умолчанию, вполне подходят для большинства организаций. Мы не будем их перечислять, поскольку читатели могут обратиться к справочной странице для файла **unbound.conf** и интернет-странице `unbound.net`. В конце справочной страницы есть полезный пример, демонстрирующий настройку производительности компьютера с небольшим объемом памяти.

🕮 Частные адреса

```
private-address: ip_адрес_или_подсеть          [нет]
private-domain: имя_домена                     [нет]
```

Инструкция `private-address` блокирует указанные в списке IP-адреса, не позволяя отправлять им ответы на запросы. Обычно она используется в сочетании с пространством частных IP-адресов RFC1918 (см. раздел 14.4), чтобы не допустить их разглашение в Интернете. Это типичное поведение внешних организаций, но если вы на самом деле используете частные адреса RFC1918 внутри организации, то, вероятно, не захотите блокировать свои собственные внутренние адреса. Инструкция `private-domain` разрешает этот конфликт, позволяя указанному домену и всем его поддоменам иметь частные адреса.

Следующий набор параметров конфигурации относится к инструкции `remote-control`, управляющей взаимодействием между программами **unbound** и **unbound-control**. Это взаимодействие управляется самоподписанными сертификатами SSL/TLS в формате X.509, установленными программой **unbound-control-setup**. Эта инструкция имеет всего несколько параметров, поэтому мы перечислим их всех.

🕮 Управление работой программы **unbound**

```
control-enable: yes | no                [no]
control-interface: ip-адрес             [локальный узел (127.0.0.1 и ::1)]
control-port: порт                      [953]
server-key-file: файл_закрытого_ключа   [unbound_server.key]
server-cert-file: файл_сертификата_pem  [unbound_server.pem]
control-key-file: файл_закрытого_ключа  [unbound_control.key]
control-cert-file: файл_сертификата_pem [unbound_control.pem]
```

Управлять программой **unbound** можно из любой точки Интернета. Для того чтобы настроить аутентификацию, запустите программу **unbound-control**, чтобы создать соответствующие файлы сертификатов, установите параметр `control-enable` равным `yes` и параметр `control-interface` равным адресу интерфейса, который сервер будет прослушивать в ожидании команд. Для того чтобы разрешить все интерфейсы, можно использовать адрес `0.0.0.0` (и `::0`). По умолчанию требуется, чтобы контроллер был зарегистрирован на том же компьютере, что и программа **unbound**. Вероятно, это самый безопасный вариант.

🕮 Тупиковые зоны

```
stub-zone:
    name: имя_домена                    [нет]
    stub-host: имя_узла                 [нет]
    stub-addr: ip_адрес[@порт]          [нет]
```

Раздел `stub-zone` позволяет направлять запросы на конкретный домен, адресуя их авторитетному серверу, назначенному вами, а не разрешать их с помощью обычного иерархического поиска, начиная с корня. Например, вы можете пожелать, чтобы ваши пользователи видели закрытое представление вашей локальной сети, содержащее больше узлов, чем демонстрируют запросы DNS, поступающие извне. Имя "stub zone" ("тупиковая зона") является неудачным и не имеет отношения к тупиковым зонам, используемым в сервере BIND.

Для реализации этой конфигурации следует запустить авторитетный сервер, который будет обслуживать локальную версию зоны, на другом узле (или на том же самом узле, но на другом порту). Затем следует указать программе **unbound** этот сервер, используя параметры инструкции `stub-zone`. Вы можете задать либо *имя_узла* сервера (параметр `stub-host`), либо его IP-адрес (параметр `stub-addr`). Вы можете также указать *порт*, который по умолчанию имеет номер 53. Использование адреса защищает вас от зачи-

кливания в ситуациях, когда программа **unbound** не может найти имя, не имея доступ к целевой зоне.

Можно использовать сколько угодно зон.

🕮 Переадресация

```
forward-zone:
    name: имя_домена              [нет]
    forward-host: имя_сервера     [нет]
    forward-addr: ip_адрес        [нет]
```

Параметр `forward-zone` позволяет серверу **unbound** переадресовывать все запросы (или некоторые из них, в зависимости от значения параметра `name`) на другой сервер, для того чтобы помочь этому серверу создать более крупный кеш. Переадресация выполняется, только если сервер **unbound** не может ответить на запрос, используя свой собственный кеш. Общая информация о переадресации и причинах ее использования приведена в разделе 17.6.

# 17.12. Обновление файлов зон

Когда в домен вносится изменение (например, добавляется или удаляется компьютер), следует обновить файлы данных на главном сервере. Кроме того, необходимо увеличить порядковый номер в записи SOA для этой зоны. В заключение вы должны заставить программное обеспечение принять изменения. Последний этап зависит от программного обеспечения.

- **Сервер BIND**. Выполните команду **`rndc reload`**, чтобы демон **named** принял изменения. Вы можете также прекратить работу демона **named** и запустить его снова, но если ваш сервер является одновременно авторитетным по отношению к вашей зоне и рекурсивным по отношению к вашим пользователям, то эта операция аннулирует кешированные данные, полученные от других доменов.
- **Сервер NSD**. Выполните команду **`nsdc rebuild`**, а затем команду **`nsdc reload`**. Сервер **nsd** не выполняет кеширование, поэтому никаких побочных эффектов от перезагрузки не будет.

Обновленные данные немедленно передаются подчиненным серверам главных серверов BIND, поскольку параметр `notify` включен по умолчанию. При работе с сервером NSD необходимо конфигурировать списки управления доступом в разделе `notify`. Если же по какой-то причине он отключен, подчиненным серверам придется дождаться, пока истечет период обновления, установленный в записи SOA зоны (обычно один час).

Если параметр `notify` отключен, можно выполнить на каждом подчиненном сервере BIND команду **`ndc reload`**, которая заставит демон связаться с главным сервером, убедиться в том, что зонная база данных изменена, и запросить ее пересылку. Соответствующая команда сервера NSD — **`nsd reload`**.

При изменении имени или IP-адреса компьютера не забывайте модифицировать зоны как прямого, так и обратного преобразований. Игнорирование последних может привести к появлению скрытых ошибок: часть команд будет работать, а часть — нет.

Если изменить файлы данных, но не обновить порядковый номер в записи SOA, то изменения вступят в силу только на главном сервере (после перезагрузки), но не на подчиненных.

Не следует редактировать файлы данных, относящиеся к подчиненным серверам. Эти файлы ведет сам демон **named**, и системные администраторы не должны вмеши-

ваться в его работу. Лучше просматривать файлы данных сервера BIND, не делая в них никаких изменений. Сервер NSD ведет базы данных, которые невозможно проверить непосредственно. Однако по умолчанию изменения записываются в текстовые зонные файлы командой **nsd-patch**.

Документом RFC2136 разрешается менять зонные базы данных сервера BIND программным путем. Эта возможность, называемая динамическим обновлением, необходима для протоколов автоматического конфигурирования, например DHCP. О том, как работает этот механизм, речь пойдет чуть ниже.

## Передача зоны

Серверы DNS синхронизируются посредством механизма передачи зоны. Передача зоны может включать в себя всю зону (такая передача называется AXFR) или только последние изменения (такая передача именуется IXFR). По умолчанию передача зоны осуществляется по протоколу TCP через порт 53. Сервер BIND регистрирует соответствующую информацию в системном журнале с пометкой "xfer-in" или "xfer-out". Сервер NSD включает ее в обычный протокольный поток.

В процессе передачи зоны подчиненный сервер запрашивает информацию от главного сервера и создает резервную копию данных о зоне на диске. Если данные на главном сервере не изменились, что определяется по порядковым номерам (не по реальным данным), то обновления не выполняются и резервные файлы просто фиксируются. (Иначе говоря, их время изменения устанавливается равным текущему времени.)

И отправляющий, и получающий серверы способны отвечать на запросы при передаче зоны. Подчиненный сервер начинает использовать новые данные только после завершения передачи.

Если зона очень большая (например, `com`) или обновляется динамически (о чем пойдет речь в следующем подразделе), то объем изменений обычно мал в сравнении с размером зоны. При передаче IXFR пересылаются только изменения (когда размер изменений начинает превышать размер зоны, включается режим обычной передачи AXFR). Механизм IXFR напоминает программу **patch**: старая база данных сравнивается и синхронизируется с новой.

В системе BIND передача IXFR задается по умолчанию, а демон **named**, предназначенный для ведения журнала транзакций, — именем *имя_зоны*.**jnl**. В инструкции `server` любого сервера, которому нужны такие передачи, можно задать параметры `provide-ixfr` и `request-ixfr`. Параметр `provide-ixfr` включает и отключает службу IXFR для зон, по отношению к которым сервер является главным. Параметр `request-ixfr` включает и отключает службу IXFR для зон, по отношению к которым сервер является подчиненным.

```
provide-ixfr yes ;        # В инструкции сервера BIND
request-ixfr yes ;        # В инструкции сервера BIND
```

Механизм IXFR может работать и с зонами, редактируемыми вручную. Для того чтобы включить этот режим, следует включить параметр `ixfr-from-difference`. Механизм IXFR требует, чтобы зонные файлы были упорядочены в каноническом порядке. За это отвечает демон **named**, но при этом затрачивается определенная память сервера и ресурсы центрального процессора. Механизм IFXR компенсирует эти затраты за счет уменьшения сетевого трафика.

При запросе передачи зоны подчиненные серверы NSD используют передачу IXFR, но если все главные серверы поддерживают передачу AXFR, то подчиненные серверы

переключаются именно на этот режим. Поскольку данные в системе NSD хранятся в виде скомпилированных файлов базы данных, упорядочение файлов для передачи IXFR не требуется. Если передача была прервана, то сервер NSD сохраняет состояние демона передачи зоны в файле, заданном атрибутом `xfrdfile`.

Перезагрузкой после выполнения передач IXFR можно управлять с помощью атрибута `xfrd-reload-timeout`. По умолчанию он равен 10 секундам, так что изменения IXFR успевают в определенной степени сгруппироваться.

В системе BIND запрос IXFR к серверу, который не поддерживает его автоматически, заменяется стандартной передачей зоны AXFR. В системе NSD эту замену можно запретить, установив атрибут `allow-axfr-fallback` равным `no`.

Обе системы прилагают много сил для того, чтобы при сбое сервера во время передачи зоны информация не была повреждена.

## Динамические обновления в системе BIND

Система DNS основана на предположении, что соответствие между именами и адресами относительно стабильно и меняется нечасто. Но это правило постоянно нарушается, если в организации используется протокол DHCP и при подключении к сети компьютеру динамически назначается IP-адрес. Существует два классических решения этой проблемы: добавить обобщенные записи в базу данных DNS или непрерывно редактировать DNS-файлы. В большинстве случаев ни одно из решений не является удовлетворительным.

Первое решение должно быть знакомо каждому, кто имеет коммутируемый выход в Интернет. Конфигурация DNS в этом случае выглядит примерно следующим образом.

```
dhcp-host1.domain.    IN  A   192.168.0.1
dhcp-host2.domain.    IN  A   192.168.0.2
...
```

Это простое решение, но оно означает, что указанные имена постоянно связаны с IP-адресами и, следовательно, компьютер, получающий новый адрес, меняет имя. В такой среде трудно соблюдать требования безопасности и вести журнальную регистрацию.

Механизм динамических обновлений, появившийся в последних версиях BIND, предлагает альтернативное решение. Он позволяет демону DHCP уведомлять сервер BIND о сделанных адресных назначениях, обновляя, таким образом, содержимое базы данных DNS "на лету". Динамические изменения могут добавлять, удалять и модифицировать запросы о ресурсах. Если динамические изменения дозволены, демон **named** ведет журнал динамических изменений (*имя_зоны*.**jnl**), который может оказаться полезным при сбое сервера. Демон **named** восстанавливает состояние зоны, сохраненное в памяти, считывая исходные зонные файлы и воспроизводя изменения на основе журнала.

После того как зона была динамически обновлена, ее уже нельзя редактировать вручную. Необходимо сначала остановить сервер BIND с помощью команд **ndc freez** *зона* или **ndc freez** *зона класс представление*. Эти команды синхронизируют журнальный файл с файлом главной зоны на диске, а затем удаляют журнал. После этого можно отредактировать зонный файл вручную. К сожалению, исходное форматирование файла пропадет (он будет иметь вид файла, который ведется демоном **named** на подчиненных серверах).

При "заморозке" зоны попытки динамического обновления отменяются. Для того чтобы загрузить зонный файл с диска и все-таки выполнить динамическое обновление, следует использовать команду **rdns thaw** с теми же аргументами, которые были указаны при "заморозке" зоны.

Утилита **nsupdate**, входящая в дистрибутив BIND 9, позволяет осуществлять динамические обновления из командной строки. Утилита работает в пакетном режиме, принимая команды, вводимые с клавиатуры, или читая их из файла. Пустая строка или команда **send** являются признаком завершения обновления и пересылают изменения на сервер. Две пустые строки означают конец входного потока. В командном языке предусмотрена примитивная инструкция `if`, позволяющая формулировать условия вида "если имя неизвестно в DNS, добавить его". Искомое имя (или набор записей о ресурсах) может существовать либо отсутствовать.

Ниже показан простейший сценарий **nsupdate**, который заносит в базу данных информацию о новом узле, а также псевдоним существующего узла, при условии что этот псевдоним нигде не используется. Угловые скобки создаются программой **nsupdate** и не являются частью командного сценария.

```
% nsupdate
> update add newhost.cs.colorado.edu 86400 A 128.138.243.16
>
> prereq nxdomain gypsy.cs.colorado.edu
> update add gypsy.cs.colorado.edu CNAME evi-laptop.cs.colorado.edu
```

Динамические обновления — очень опасная возможность. Они потенциально способны предоставить право неконтролируемой записи важных системных данных. Не пытайтесь контролировать доступ на основании IP-адресов: их легко подделать. Лучше воспользоваться системой аутентификации TSIG с совместным секретным ключом. Например, в системе BIND 9 поддерживается команда

```
% nsupdate -k каталог_ключа:файл_ключа
```

или

```
% nsupdate -k каталог_ключа:секретный_ключ
```

Поскольку пароль задается в командной строке в виде `-y`, его может увидеть тот, кто запустит команду **w** или **ps** в правильный момент. По этой причине более предпочтительной является форма `-k`. Подробнее технология TSIG описана в разделе 17.13.

Динамические обновления зоны разрешаются в файле **named.conf** посредством параметра `allow-update` или `update-policy`. Параметр `allow-update` предоставляет право обновления любых записей клиентам, чьи IP-адреса и ключи шифрования указаны в списке. Параметр `update-policy` появился в BIND 9 и позволяет точнее управлять обновлениями на основании имен узлов и типов записей. Он требует использовать механизмы аутентификации. Оба параметра являются зонными.

С помощью параметра `update-policy` можно разрешить клиентам обновлять свои записи A и PTR, но не SOA, NS или KEY. Можно также позволить узлу обновлять только свои записи. Имена допускается задавать явно, в виде поддоменов, с метасимволами или с помощью ключевого слова `self`, которое задает правила доступа компьютера к собственным записям. Записи о ресурсах идентифицируются по классу или типу. Синтаксис параметра `update-policy` выглядит следующим образом.

```
update-policy { grant | deny } сущность имя_типа имя [типы] ;
```

Здесь *сущность* — это имя криптографического ключа, необходимого для авторизации обновлений. *Имя_типа* имеет четыре значения: `name`, `subdomain`, `wildcard` или `self`. *Имя* — это обновляемая зона, а *типа* — типы записей о ресурсах, которые можно обновить. Если типы не указаны, то могут обновляться записи всех типов, кроме SOA, NS, RRSIG и NSEC или NSEC3. Рассмотрим пример.

```
update-policy { grant dhcp-key subdomain dhcp.cs.colorado.edu A } ;
```

В такой конфигурации любому, у кого есть ключ dhcp-key, разрешается обновлять адресные записи в поддомене dhcp.cs.colorado.edu. Эта директива должна появиться в файле **named.conf** в инструкции zone домена dhcp.cs.colorado.edu. Потребуется также инструкция key, содержащая определение ключа dhcp-key.

Приведенный ниже фрагмент файла **named.conf** факультета компьютерных наук университета Колорадо (Computer Science Department at the University of Colorado) использует инструкцию update-policy для того, чтобы позволить студентам, находящимся в классе системного администратора, обновлять свои поддомены, не касаясь при этом остального окружения системы DNS.

```
// saclass.net
zone "saclass.net" {
   type master;
   file "saclass/saclass.net";
   update-policy {
      grant feanor_mroe. subdomain saclass.net.;
      grant mojo_mroe. subdomain saclass.net.;
      grant dawdle_mroe. subdomain saclass.net.;
      grant pirate_mroe. subdomain saclass.net.;
      ...
   };
...
```

# 17.13. Вопросы безопасности

DNS появилась как совершенно открытая система, но в процессе развития она становилась все более защищенной, приобретая необходимые средства защиты. По умолчанию каждый, у кого есть доступ в Интернет, может исследовать чужой домен отдельными запросами с помощью таких команд, как **dig**, **host** или **nslookup**. В некоторых случаях можно получить образ всей базы данных DNS.

Для устранения этих недостатков в последние версии пакета BIND были введены различные средства управления доступом, основанные на проверке адресов или криптографической аутентификации. В табл. 17.12 перечислены элементы подсистемы безопасности, которые настраиваются в файлах **named.conf**, **nsd.conf** или **unbound.conf**.

**Таблица 17.12. Средства защиты в файле named.conf**

| Параметр | Инструкции | Что определяет |
|---|---|---|
| | | BIND |
| acl | Разные | Списки управления доступом |
| allow-query | options, zone | Кто может посылать запросы зоне или серверу |
| allow-recursion | options | Кто может посылать рекурсивные запросы |
| allow-transfer | options, zone | Кто может запрашивать зонные передачи |
| allow-update | zone | Кто может выполнять динамические обновления |
| blackhole | options | Какие серверы нужно полностью игнорировать |
| bogus | server | Какие серверы никогда нельзя опрашивать |
| update-policy | zone | Какие обновления разрешены |

*Окончание табл. 17.12*

| Параметр | Инструкции | Что определяет |
|---|---|---|
| | | NSD/Unbound |
| access-control | server | Кто может посылать запросы (Unbound) |
| allow-notify | Подчиненная зона | Кто может посылать уведомления (NSD) |
| chroot | server | Корневой каталог |
| notify | Главная зона | Подчиненные серверы, которые должны получать уведомление (NSD) |
| provide-xfr | Главная зона | Получатели при передаче зон (NSD) |
| request-xfr | Подчиненная зона | Провайдеры передачи зон (NSD) |
| username | server | Пользователь, который должен работать в виртуальном окружении chroot |

Для того чтобы минимизировать риск, все три сервера имен могут запускаться в виртуальном окружении **chroot** под непривилегированным идентификатором; сервер имен **unbound** делает это по умолчанию. Все они могут использовать подпись транзакции, для того чтобы управлять взаимодействием между главными и подчиненными серверами (BIND и NSD), а также между серверами имен и их управляющими программами (BIND и Unbound). Кроме того, все они полностью поддерживают протокол DNSSEC. Эти вопросы рассмотрим в следующих разделах.

## Еще раз о списках управления доступом в сервере BIND

Список управления доступом — это именованный список соответствия адресов, который может служить аргументом различных директив, в частности `allow-query`, `allow-transfer` и `blackhole`. Синтаксис списков был рассмотрен ранее. Кроме того, списки управления доступом могут способствовать укреплению безопасности DNS-серверов.

В каждой организации должен существовать хотя бы один список для недоступных адресов и один — для локальных. Рассмотрим пример.

```
acl bogusnets {          // список недоступных и фиктивных сетей
    0.0.0.0/8;           // адреса по умолчанию
    1.0.0.0/8;           // зарезервированные адреса
    2.0.0.0/8;           // зарезервированные адреса
    169.254.0.0/16;      // канально-локальные делегируемые адреса
    192.0.2.0/24;        // тестовые адреса, наподобие example.com
    224.0.0.0/3;         // пространство групповых адресов
    10.0.0.0/8;          // частное адресное пространство (RFC1918)[26]
    172.16.0.0/8;        // частное адресное пространство (RFC1918)
    192.168.0.0/16;      // частное адресное пространство (RFC1918)
};

acl cunets {             // список сетей университета штата Колорадо
   128.138.0.0/16;       // основная кампусная сеть
   198.11.16.0/24;
   204.228.69.0/24;
};
```

[26] Не делайте частные адреса недоступными, если они используются для конфигурирования внутренних DNS-серверов!

Далее нужно в глобальном разделе `options` конфигурационного файла разместить следующие директивы.

```
allow-recursion { cunets; };
blackhole { bogusnets; };
```

Желательно также ограничить зонные передачи только легитимными подчиненными серверами. Это достигается с помощью следующих списков.

```
acl ourslaves {
   128.138.242.1;      // сервер anchor
   ...
};

acl measurements {
   198.32.4.0/24;      // проект Билла Маннинга, адрес v4
   2001:478:6:::/48;   // проект Билла Маннинга, адрес v6
};
```

Собственно ограничение реализуется такой строкой.

```
allow-transfer { ourslaves; measurements; };
```

Передачи разрешены только нашим подчиненным серверам, а также компьютерам глобального исследовательского проекта, который посвящен определению размеров сети Интернет и процента неправильно сконфигурированных серверов. Подобное ограничение лишает остальных пользователей возможности получать дамп всей базы данных с помощью команды **dig**.

Разумеется, нужно по-прежнему защищать сеть на более низком уровне с помощью списков управления доступом на маршрутизаторе и стандартных средств защиты на каждом узле. Если такой возможности нет, ограничьте трафик DNS-пакетов шлюзовым компьютером, который находится под постоянным административным контролем.

## Открытые распознаватели

Открытый распознаватель — это рекурсивный кеширующий сервер имен, получающий запросы от любого пользователя Интернета и отвечающий на них. Открытые распознаватели опасны. Внешние пользователи могут потреблять ваши ресурсы без вашего разрешения или ведома, и если они делают это со злым умыслом, кеш вашего распознавателя может выйти из строя.

Что еще хуже, открытые распознаватели иногда используются злоумышленниками для усиления распределенной атаки на основе отказа в обслуживании. Атакующие посылают запрос на ваш распознаватель, указывая фальшивый адрес источника, который является объектом атаки. Ваш распознаватель послушно отвечает на эти запросы и посылает огромный пакет жертве. Жертва не посылала запросы, но она обязана выполнить маршрутизацию и обработать сетевой трафик. Умножьте объем пакета на количество распознавателей, и вы осознаете реальную опасность, грозящую жертве.

Статистика свидетельствует о том, что от 75 до 80% кеширующих серверов имен в настоящее время являются открытыми распознавателями! Сайт `dns.measurement-factory.com/tools` может помочь вам проверить ваш сайт. Зайдите на него, выберите команду "open resolver test" и наберите IP-адрес вашего сервера имен. Вы можете также проверить все серверы имен вашей сети или все серверы вашей организации, используя идентификаторы `whois`.

Для того чтобы ваши кеширующие серверы имен отвечали на запросы только локальных пользователей, используйте список управления доступом в файлах **named.conf** и **unbound.conf**.

## Работа в виртуальном окружении chroot

Если хакеры взломают ваш сервер, то они смогут получить доступ к системе под видом законного пользователя. Для того чтобы уменьшить опасность, возникающую в этой ситуации, вы можете запустить сервер в виртуальном окружении **chroot** либо под видом непривилегированного пользователя, либо сделать и то, и другое одновременно.

Для демона **named** флаг команды **-t** задает каталог виртуального окружения **chroot**, а флаг **-u** указывает идентификатор пользователя, под которым работает демон **named**. Рассмотрим пример.

```
$ sudo named -u 53
```

Эта команда запускает демон **named** как корневой процесс, но после того как демон **named** закончит рутинные операции в роли корневого процесса, он потеряет привилегии и запустится под идентификатором 53.

Для серверов **nsd** и **unbound** того же эффекта можно добиться с помощью опций `username` и `chroot` команды `server` в файле конфигурации. Эти опции также можно указать в командной строке сервера **nsd** с теми же самыми флагами, что и в пакете BIND, — **-u** и **-t** соответственно.

Многие организации не используют флаги **-u** и **-t**, но если объявлена тревога, они должны реагировать быстрее, чем хакеры.

Виртуальное окружение **chroot** не может быть пустым каталогом, поскольку оно должно содержать все файлы, необходимые серверу имен для нормальной работы, — **/dev/null**, **/dev/random**, зонные файлы, файлы конфигурации, ключи, файлы системного журнала и доменного сокета UNIX, **/var** и др. Для того чтобы настроить их, требуется выполнить довольно большую работу. Вызов системы **chroot** осуществляется после загрузки библиотек, поэтому копировать общие библиотеки в виртуальное окружение не обязательно.

## Безопасные межсерверные взаимодействия посредством технологий TSIG и TKEY

Пока спецификация DNSSEC (описана в следующем подразделе) находилась на стадии принятия, группа IETF разработала более простой механизм, названный TSIG (RFC2845). Он позволял организовать безопасное взаимодействие серверов благодаря использованию сигнатур транзакций. Контроль доступа, основанный на таких сигнатурах, надежнее контроля на основе исходных IP-адресов. Сигнатура транзакции аутентифицирует пару "отправитель/получатель" и позволяет проверить, изменились ли данные в процессе передачи.

В технологии TSIG применяется симметричная схема шифрования, т.е. ключ шифрования совпадает с ключом дешифрования. Такой ключ называется *совместным секретным ключом* (shared secret). Спецификация TSIG допускает использование нескольких методов шифрования. В пакете BIND реализованы методы MD5, SHA-1, SHA-224 и SHA-256. Сервер NSD реализует те же методы, за исключением метода SHA-224. Для каждой пары серверов, между которыми организуется защищенный канал связи, должен создаваться свой ключ.

Спецификация TSIG гораздо менее затратная в вычислительном плане, чем шифрование с открытым ключом, но она подходит только для локальной сети, где число пар взаимодействующих серверов невелико. На глобальную сеть эта спецификация не распространяется.

## Настройка технологии TSIG для сервера BIND

Утилита **dnssec-keygen**, являющаяся частью пакета BIND, генерирует ключ для пары серверов. Рассмотрим, например, следующую команду.

```
$ dnssec-keygen -a HMAC-MD5 -b 128 -n HOST master-slave1
```

Флаг **-b 128** означает, что утилита **dnssec-keygen** должна создать 128-разрядный ключ. В данном случае мы используем 128-разрядный ключ только лишь для того, чтобы он поместился на странице. В реальной жизни следует использовать более длинный ключ; максимально допустимая длина ключа равна 512.

Данная команда создает два файла: **Kmaster-slave1.+157+09068.private** и **Kmaster-slave1.+157+09068.key**, где **157** — код алгоритма HMAC-MD5, а **09068** — случайное число, используемое в качестве идентификатора ключа на случай, если у одной пары серверов есть несколько ключей.[27]

Оба файла содержат один и тот же ключ, но в разных форматах. Файл **.private** выглядит примерно так.

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: jxopbeb+aPc71Mm2vc9R9g==
```

Содержимое файла **.key** будет таким.

```
master-slave1. IN KEY 512 3 157 jxopbeb+aPc71Mm2vc9R9g==
```

Обратите внимание на то, что утилита **dnssec-keygen** добавила точки в конец имени каждого ключа в обоих именах файлов и внутри файла **.key**. Это объясняется тем, что когда утилита **dnssec-keygen** использует ключи DNSSEC, добавляемые в зонные файлы, имена этих ключей должны быть полностью определены именами доменов и, следовательно, должны заканчиваться точкой. Таким образом, нам нужны два инструмента: один для совместных секретных ключей, а второй — для пар открытых ключей.

Вообще-то, файл **.key** на самом деле не нужен: он создается потому, что утилита **dnssec-keygen** имеет двойное назначение. Просто удалите его. Число 512 в записи KEY означает не длину ключа, а флаг, идентифицирующий запись как запись о главном ключе DNS.

После всех этих сложностей вы могли быть разочарованы тем, что сгенерированный ключ представляет собой просто длинное случайное число. Этот ключ можно было бы сгенерировать вручную, записав строку в кодировке ASCII, длина которой делится на четыре, и считать, что вы применили 64-разрядное кодирование, или использовать утилиту **mmencode** для того, чтобы зашифровать случайную строку. Способ, которым вы кодируете ключ, не имеет значения; он просто должен существовать на обеих машинах.

🕮 Команда **scp** является частью пакета SSH.

Скопируйте ключ из файла **.private** на оба сервера с помощью команды **scp** или просто скопируйте и вставьте его. *Не используйте* утилиты **telnet** и **ftp** для копирования ключа: даже внутренние сети могут быть небезопасными.

[27] Это число выглядит случайным, но на самом деле оно представляет собой хешированное значение ключа TSIG.

Ключ должен быть записан в файлах **named.conf** на обоих компьютерах. В связи с тем, что эти файлы — общедоступны, а ключ — нет, поместите ключ в отдельный файл, который будет подставляться в файл **named.conf**. Файл ключа должен иметь уровень доступа, равный 600, и принадлежать пользователю демона **named**.

Например, можно создать файл **master-slave1.tsig** и включить в него следующий фрагмент.

```
key master-slave1 {
   algorithm hmac-md5 ;
secret "сгенерированный_ключ" ;
};
```

В файл **named.conf**, ближе к началу, нужно добавить такую строку.

```
include "master-slave1.tsig";
```

На данном этапе лишь определен ключ. Для того чтобы его можно было использовать для подписи и проверки обновлений, нужно посредством инструкции `server` и директивы `keys` заставить каждый сервер идентифицировать другую сторону. Например, в инструкцию `zone` на главном сервере можно включить строку

```
allow-transfer { key master-slave1. ; };
```

а в файл **named.conf** подчиненного сервера — строку

```
server IP-адрес-главного_сервера { keys { master-slave1 ; } ; } ;
```

Имя ключа в нашем примере носит общий характер. Если вы используете ключи TSIG для многих зон, то, возможно, захотите включить в имя ключа имя зоны, чтобы все стало ясно.

Для того чтобы протестировать конфигурацию системы TSIG, выполните утилиту **named-checkconf** и проверьте синтаксис. Затем с помощью команды **dig** попытайтесь выполнить передачу зоны (`dig@master axfr`) от сервера slave1 и какой-нибудь другой машины. Первая попытка должна оказаться успешной, а вторая должна завершиться сбоем и сообщением об ошибке "Transfer failed". Для того чтобы убедиться, что все в порядке, удалите раздел `allow-transfer` и попытайтесь выполнить команду **dig** снова. На этот раз обе попытки должны завершиться успехом. (Не забудьте вернуть в файл раздел `allow-transfer`!) И наконец, увеличьте порядковый номер зоны на главном сервере, выполните команду **rndc reload** и просмотрите системный журнал на подчиненном сервере, чтобы убедиться, что изменения приняты и передача зоны произошла.

Если вы впервые используете подписи транзакций, запускайте демона **named** на первом уровне отладки (режимы отладки будут описаны позднее), пока сообщения об ошибках не исчезнут. В ранних версиях пакета BIND сервер не понимал подписанных сообщений и сообщал об ошибках, что иногда приводило к ошибочной передаче зоны.

🕮 Протокол NTP описан в разделе 9.5

При использовании ключей TSIG и подписей транзакций между главным и подчиненным серверами, необходимо синхронизировать часы на обоих серверах по протоколу NTP. Если часы слишком сильно расходятся (более чем на пять минут), то верификация подписи не будет выполнена. Эту проблему иногда очень сложно распознать.

TSIG — это механизм сервера BIND, позволяющий двум узлам автоматически генерировать совместный секретный ключ, не прибегая к телефонным звонкам или секретному копированию для распространения ключа. Он использует алгоритм обмена ключами Диффи-Хеллмана (Diffie-Hellman key exchange algorithm), в котором на каждой стороне генерируется случайное число, над ним выполняются определенные математи-

ческие операции, а результат посылается другой стороне. Затем каждая сторона объединяет свое и полученное числа по определенному математическому правилу и получает один и тот же ключ. Злоумышленник может перехватить сообщение, но он не сможет выполнить над ним требуемые математические операции.[28]

Серверы компании Microsoft используют нестандартный вариант механизма TSIG, который называется GSS-TSIG. В нем для обмена ключами используется технология TKEY. Если вам нужно, чтобы сервер компании Microsoft взаимодействовал с сервером BIND, используйте опции `tkey-domain` и `tkey-gssapi-credential`.

Другой механизм для подписи трансакций между серверами или службами динамического обновления и главным сервером называется SIG(0). Он использует криптографию, основанную на открытых ключах. Детали этой технологии описаны в документах RFC2535 и RFC2931.

## Механизм TSIG на сервере NSD

Для того чтобы сгенерировать ключи TSIG для списка управления доступом на сервере NSD, можно использовать утилиту **`ldns-keygen`** из каталога **`examples`** в дистрибутивном пакете **`ldns`**. Подробности описаны немного ниже. Сервер NSD не поддерживает ключи SIG(0) и обмен ключами TKEY по алгоритму Диффи-Хеллмана.

## Технология DNSSEC

DNSSEC — это набор расширений DNS, позволяющих аутентифицировать источник данных зоны и проверить их целостность, используя шифрование с открытым ключом. Таким образом, DNSSEC позволяет DNS-клиентам задавать вопросы вида "Действительно ли данные зоны поступили от владельца зоны?" и "Те ли это данные, которые послал владелец?".

Технология DNSSEC реализует три разных механизма:

- распределение ключей посредством записей KEY, хранящихся в зонных файлах;
- проверка подлинности серверов и данных;
- проверка целостности данных.

Здесь формируется цепочка доверия: корневые серверы предоставляют подтверждающую информацию для доменов верхнего уровня, те — для доменов второго уровня и т.д. По крайней мере, так это задумывалось. В начале 2010 года и корневой, и большинство доменов верхнего уровня оставались без подписей.

Корпорация ICAAN и Министерство торговли США стали делать определенные шаги в направлении внедрения подписей в корневом домене, но пока все ограничивается обещаниями. Возможно, это произойдет в средине 2010 года.[29] Со своей стороны, по-видимому, доменный регистратор VerySign не собирается подписывать зоны `.com` и `.net`.[30] Эти зоны уже слишком велики, а их подписанные версии будут еще больше и потребуют дополнительного серверного обеспечения. Более того, сертификаты X.509 компании VeriSign обеспечивают значительную долю ее прибыли, а технология DNSSEC

[28] Математическую основу этого алгоритма образует задача дискретного логарифмирования, особенность которой состоит в том, что в модулярной арифметике возвести число в степень довольно просто, а вычислить логарифм по этой степени практически невозможно.

[29] Подписание корневой зоны состоялось 15 июля 2010 года .— Примеч. ред.

[30] В конце 2010 года доменный регистратор Verisign сообщил о поддержке технологии DNSSEC для доменной зоны `.net`.

может заменить эти сертификаты для определенных приложений. Тем не менее компания VerySign обещала подписать зону .com в 2011 году, синхронизировав этот процесс с переговорами о своем контракте с корпорацией ICAAN, который должен быть заключен в 2012 году.

К счастью, концепция точек доверия позволяет развернуть процесс легализации технологии DNSSEC и расширить защищенные области в дереве DNS еще до внедрения подписанных корневых доменов и доменов верхнего уровня.

В системах шифрования с открытым ключом используются два ключа: один шифрует (подписывает) сообщение, а другой дешифрует (проверяет) его. Публикуемые данные подписываются секретным "личным" ключом. Любой может проверить правильность цифровой подписи с помощью соответствующего "открытого" ключа, который свободно распространяется. Если открытый ключ позволил правильно расшифровать зонный файл, значит, зона была зашифрована искомым личным ключом. Суть в том, чтобы убедиться в подлинности открытого ключа, используемого для проверки. Такие системы шифрования позволяют одной из сторон поставить свою "подпись" на открытом ключе, передаваемом другой стороне, гарантируя легитимность ключа. Отсюда термин "цепочка доверия".

Данные, составляющие зону, слишком велики для шифрования с открытым ключом; процесс шифрования получится слишком медленным. Вместе с тем данные сами по себе не секретны, поэтому они просто пропускаются через функцию хеширования (к примеру, вычисляется контрольная сумма MD5), а результат подписывается (шифруется) секретным ключом зоны. Полученный зашифрованный хеш-код называется *цифровой подписью* или *сигнатурой* зоны. Цифровые подписи обычно добавляются к данным, подлинность которых они удостоверяют в виде записей RRSIG в подписанном зонном файле.

Для проверки сигнатуры ее нужно дешифровать открытым ключом подписавшей стороны, "прогнать" данные через тот же алгоритм хеширования и сравнить вычисленный хеш-код с расшифрованным. В случае совпадения подписавшее лицо считается аутентифицированным, а данные — целостными.

В технологии DNSSEC каждая зона имеет открытые и секретные ключи. Фактически она имеет два набора ключей: пара ключей для подписи зоны и пара ключей для подписи ключа. Секретным ключом подписи зоны подписывается каждый набор ресурсов (т.е. каждый набор записей одного типа, относящихся к одному узлу). Открытый ключ подписи зоны используется для проверки сигнатур и включается в зонную базу данных в виде записи DNSKEY.

Родительские зоны содержат записи DS, представляющие собой хешированный вариант записей DNSKEY для самоподписывающихся ключей подписи ключей (self-signed key-signing keys). Сервер имен проверяет аутентичность записи DNSKEY дочерней зоны, сравнивая ее с подписью родительской зоны. Для проверки аутентичности ключа родительской зоны сервер имен проверяет родительскую зону родительской зоны и так далее, пока не вернется в корневую зону. Открытый ключ корневой зоны должен быть открыто опубликован и включен в файлы "подсказок" корневой зоны.

В соответствии со спецификациями DNSSEC, если зона имеет несколько ключей, каждый из них должен применяться при проверке данных. Это требование объясняется тем, что ключи могут быть изменены без прерывания работы службы DNS. Если рекурсивный сервер имен, использующий технологию DNSSEC, посылает запрос неподписанной зоне, то поступающий неподписанный ответ считается корректным. Проблема возникает лишь тогда, когда срок действия подписи истек или родительская и дочерняя зоны не согласовали текущий ключ DNSKEY дочерней зоны.

Прежде чем погрузиться в механизм генерирования ключей и подписания зон, необходимо иметь краткое представление о реальном состоянии технологии DNSSEC и ее влиянии на системных администраторов. Ее можно развернуть уже сейчас, но остается нерешенным множество проблем. К преимуществам технологии DNSSEC можно отнести следующее.

- Текущие версии программного обеспечения серверов DNS (как **named**, так и **nsd/unbound**) уже готовы. Существуют инструменты для подписания зон и верификации подписей.
- Совсем скоро будут созданы подписанные зоны. В начале 2010 года домены `.gov`, `.org` и некоторые домены ccTLD (преимущественно в Европе) уже были подписаны. (Швеция первой подписала домен TLD.) Корневой домен был подписан в 2010 году, а другие домены gTLD были подписаны позднее. Правительство США потребовало, чтобы все сайты в домене `.gov` также были подписаны.
- Стандарты IETF кажутся функциональными и работоспособными.

Однако остается две проблемы: распределение ключей и размер пакетов.

Пока корневой домен и домены TLD не подписаны, цепочка доверия часто обрывается. Сайты, требующие, чтобы их зоны были подписаны, должны найти другие способы опубликовать свои ключи. Схема динамической проверки (lookaside validation scheme), описанная Сэмом Уейлером (Sam Weiler) из компании Sparta в документах RFC4431 и RFC5074, предоставляет удобный способ решения этой проблемы, основанный на привлечении к проверке ключей сайтов посторонних организаций, таких как ISC. Это удобное временное решение, которое также имеет изъян. Организация ICS используется для запуска критически важных серверов (в частности, она контролирует работу корневого сервера F), но иногда происходят сбои. Кроме того, существует проблема, связанная с возможным нарушением конфиденциальности со стороны посредника, знающего все сайты, которые посещают пользователи. (Разумеется, это относится ко всем операторам корневых серверов.) Другой паллиатив заключается в использовании так называемых промежуточных репозитариев точек доверия (ITAR — Interim Trust Anchor Repository). Этот механизм описан на сайте `itar.iana.org`.

Надежные ключи имеют большую длину, и некоторые сайты предпочитают распространять некоторые из них. Это приводит к увеличению размера пакетов. Спецификации DNSSEC требуют использования технологии EDNS0, представляющей собой расширенный протокол DNS, поддерживающий пакеты UDP, размер которых превышает 512 байт. Однако их поддерживают не все реализации; эти реализации не могут использовать технологию DNSSEC.

Даже на серверах, поддерживающих технологию EDNS0, предельный размер пакета, передаваемого через линию связи между двумя серверами, может оказаться меньше, чем размер огромного подписанного пакета, оснащенного ключами. Если пакет слишком велик, то на уровне IP теоретически его можно разбить на фрагменты, но проблема все равно остается. Некоторые реализации протокола TC/IP фрагментируют пакеты TCP, но не UDP. Некоторые брандмауэры не могут сохранить состояние, чтобы затем правильно восстановить пакеты UDP. Кроме того, некоторые брандмауэры передают на порт 53 UDP-пакеты, размер которых превышает 512 байт. Ой! Если ответ UDP искажен или не проходит через линию связи, то клиент переключается на протокол TCP, имеющий свои собственные проблемы, связанные с производительностью.

Укажем еще несколько проблем.

- Россия отказывается использовать алгоритм RSA, а ведь он является единственным алгоритмом, который необходим для реализации серверов имен, поддерживающих протокол DNSSEC. Россия стандартизировала алгоритм GOST, представляющий собой алгоритм симметричного шифрования, схема которого напоминает алгоритм DES.[31]
- Китай и некоторые другие страны имеют свои собственные корневой, .com- и .net-серверы, что приводит к расщеплению дерева имен DNS. Как технология DNSSEC будет работать с расщепленным деревом имен?
- Стандарт RFC5011, предложенный для автоматического обновления точек доверия DNSSEC, резко увеличивает затраты на обработку ключей за счет добавления ключей к пользовательским наборам записей о ресурсах DNSKEY. Это расширение может исчерпать память и кажется плохой идеей. Кроме того, иногда при использовании схемы RFC5011 ключи сайта оказываются аннулированными, даже если эти ключи все еще необходимы для верификации кешированных данных.
- Дистрибутивные пакеты системы Linux поставляются вместе со списком ключей, необходимых для начала работы. Это кажется нам неудачной идеей, поскольку эти ключи со временем неизбежно устареют и станут неправильными. Ваша система DNS будет медленно деградировать, а вы не будете знать почему. (Списки ключей сами по себе не всегда плохи. Например, их можно использовать на веб-сайтах RIPE и IANA для перекрестной проверки ключей, полученных по системе DNS, пока корневой домен и домены TLDs не будут подписаны.)
- Иногда нам необходим хорошо известный узел (аналог www для веб-серверов), который можно было бы использовать для публикации открытых ключей в ожидании, пока будет подписана вершина дерева DNS — ключ.*имя-домена* или что-то в этом роде.

Системные администраторы должны начать думать о подписании своих доменов и установке одного или двух новых серверов. Мы рекомендуем разворачивать протокол DNSSEC уже сейчас, но сначала тщательно осуществить этот процесс на тестовой сети и лишь затем переходить к масштабу предприятия.

Развертывание протокола DNSSEC можно выполнить в виде двух независимых этапов.

- Подпишите ваши зоны и предоставляйте подписанные данные клиентам, поддерживающим протокол DNSSEC.
- Проверяйте ответы на запросы ваших пользователей.

Испытайте удобный инструмент SecSpider, разработанный в университете UCLA (University of California, Los Angeles) и размещенный на сайте `secspider.cs.ucla.edu`. Он зондирует вашу установку протокола DNSSEC с нескольких узлов Земного шара, чтобы убедиться, что ваши ключи доступны и крупные пакеты, содержащие эти ключи, могут достичь этих узлов. (Именно благодаря инструменту SecSpider была выявлена проблема, связанная с предельно допустимым размером пакета, передаваемым по линии связи при использовании протокола DNSSEC.) Программа SecSpider также

[31] Протокол GOST является надежным (насколько нам известно) и использует намного более короткую длину ключа, чем другие алгоритмы. Поскольку он является алгоритмом симметричного шифрования и не относится к системам с открытым ключом, он может заменить протокол TSIG, но не может использоваться для реализации технологии DNSSEC. Предложения разрешить использование алгоритма GOST блокируются организацией IETF.

идентифицирует неправильные конфигурации протокола DNSSEC. Это может оказаться полезным для сбора данных студентами, пишущими дипломные работы. Вы можете также получить копии открытых ключей других подписанных зон с веб-сайта SecSpider (`secspider.cs.ucla.edu/trust-anchors.conf`).

Центр DNS-OARC (DNS Operations, Analysis, and Research Center) реализовал тестовый сервер для проверки размеров ответов. С помощью утилиты **dig** на этот сервер можно послать запрос, чтобы выяснить, насколько большой ответный пакет UDP может пройти по линии связи между этим сервером и вашим сайтом. Рассмотрим следующий пример.

```
$ dig +short rs.dns-oarc.net txt
rst.x1014.rs.dns-oarc.net.
rst.x1202.x1014.rs.dns-oarc.net.
rst.x.1382.x1202.x1014.rs.dns-oarc.net.
"63.231.83.113 DNS reply size limit is at least 1382 bytes"
"63.231.83.113 sent EDNS buffer size 4096"
```

Этот пример свидетельствует о том, что по линии связи могут пройти ответы DNS, размер которых не превышает 1382 байт, несмотря на то что объявленный размер буфера составляет 4096 байт. В данном случае проблема, вероятно, заключается в том, что брандмауэр не допускает фрагментирование пакетов UDP.

Другие общепринятые размеры пакетов: 486, если сервер не поддерживает протокол EDNS0 и ограничивает размер пакетов UDP 512 байт, и 4023 — если можно полностью использовать буфер размером 4096 байт. Если передать команде **dig** аргумент *@server*, то можно увидеть ограничения, наложенные на размер пакетов, которые могут проходить по линии связи между машиной DNS-OARC и данным сервером. Более подробную информацию можно найти на сайте `dns-oarc.net/oarc/services/replysizetest`.

Если вы планируете реализовать протокол DNSSEC, а программа SecSpider или сервер DNS-OARC свидетельствуют о проблемах с размерами пакетов, возможно, настало время поговорить со специалистами, управляющими брандмауэром, и устранить проблемы перед развертыванием протокола DNSSEC.

## Правила протокола DNSSEC

Прежде чем приступать к развертыванию протокола DNSSEC, следует усвоить несколько правил и процедур или хотя бы подумать о них. Рассмотрим примеры.

- Ключи какого размера вы будете использовать? Более длинные ключи являются более надежными, но они увеличивают размеры пакетов.
- Как часто будете изменять ключи, если никаких нарушений правил безопасности не происходит?
- Как будете распространять свои открытые ключи? Как сайты, нуждающиеся в ваших ключах, будут проверять их аутентичность?

Мы предлагаем завести специальный журнал, в который вы будете записывать данные о каждом сгенерированном ключе; об аппаратном и программном обеспечении, используемом для этого; о дескрипторе, присвоенном ключу; о версии программного генератора ключей; использованном алгоритме; длине ключа; и сроке действия подписи. Если впоследствии криптографический алгоритм окажется скомпрометированным, вы сможете проверить свой журнал и выяснить, не подвергаетесь ли вы опасности.

## Записи о ресурсах DNSSEC

Протокол DNSSEC использует шесть типов записей о ресурсах, кратко описанные в разделе 17.8, — DS, DLV, DNSKEY, RRSIG, NSEC и NSEC3. Сначала мы опишем их в целом, а затем очертим их использование в процессе подписания зоны. Каждая из этих записей создается инструментами протокола DNSSEC, а не вводится в зонный файл с помощью текстового редактора.

Запись DS (Designated Signer) возникает только в дочерней зоне и означает, что подзона является безопасной (подписанной). Она также идентифицирует ключ, используемый дочерней зоной для подписания своего собственного набора записей о ресурсах KEY. Запись DS содержит идентификатор ключа (пятизначное число), криптографический алгоритм, тип дайджеста (digest type) и дайджест записи об открытом ключе, разрешенном (или использованном) для подписи записи о ключе дочерней зоны.

Если ваша родительская зона не подписана, вы можете установить точку доверия в организации ISC, используя запись DLV (domain lookaside validation) в том же самом формате. Рассмотрим пример точки доверия и соответствующей записи.[32]

```
example.com.              IN  DS   682  5  1  12898DCF9F7AD20DBCE159E7...
example.com.dlv.isc.org.  IN  DLV  582  5  1  12898DCF9F7AD20DBCE159E7...
```

Изменение существующих ключей в родительской и дочерней зонах является сложной проблемой и требует сотрудничества и взаимодействия между родительской и дочерней зонами. Для решения этой проблемы создаются записи DS, используются отдельные ключи для подписи ключей и зон, а также применяются пары многократных ключей.

Ключи, содержащиеся в записи о ресурсах DNSKEY, могут быть либо ключами для подписания ключа (key-signing key — KSK) или ключами для подписания зоны (zone-signing key — ZSK). Для различия между ними используется новый флаг под названием SEP ("secure entry point" — "точка безопасного входа"). Пятнадцатый бит поля флагов устанавливается равным единице для ключа KSK и 0 — для ключа ZSK. Это соглашение делает поле флагов нечетным числом для ключа KSK и четным для ключей ZSK, если интерпретировать его как десятичное число. В настоящее время эти значения равны 257 и 256 соответственно.

Для обеспечения удобного перехода от одного ключа к другому можно генерировать многократные ключи. Дочерняя зона может изменить свои ключи подписания зоны, не сообщая об этом родительской зоне. Она должна согласовывать с родительской зоной только изменение ключа для подписания ключей. Когда ключ изменяется, и старый, и новый ключи на определенном отрезке времени считаются действующими. Как только срок действия кешированных значений в Интернете истечет, старый ключ будет считаться отмененным.

Запись RRSIG — это подпись набора записей о ресурсах (т.е. набора всех записей одного и того же типа и с одним и тем же именем внутри зоны). Записи RRSIG генерируются программным обеспечением, предназначенным для подписания зоны, и добавляются в подписанную версию зонного файла.

Запись RRSIG содержит много информации.

- Тип записей о ресурсах, входящих в подписываемый набор.
- Алгоритм, используемый для подписания и закодированный небольшим числом.

[32] В этом разделе 64-разрядные хеши и ключи были усечены, чтобы сэкономить пространство и лучше проиллюстрировать структуру записей.

- Количество меток (фрагментов, разделенных точками) в поле имени.
- Значение TTL для подписываемого набора записей.
- Срок действия подписи (в формате *ггггммддччсссс*).
- Время подписания набора записей (также в формате *ггггммддччсссс*).
- Идентификатор ключа (пятизначный номер).
- Имя подписавшего (имя домена).
- Сама цифровая подпись (64-разрядная).

Рассмотрим пример.

```
RRSIG   NS 5 2 57600 20090919182841 (
        20090820182841 23301 example.com.
        pMKZ76waPVTbIguEQNUojNVlVewHau4p...==)
```

При подписании зоны также генерируются записи NSEC или NSEC3. В отличие от подписанных наборов записей, они сертифицируют интервалы *между* именами наборов записей и тем самым позволяют подписывать ответ типа "нет такого домена" или "нет такого набора записей о ресурсах". Например, сервер может ответить на запрос записей A с именем `bork.atrust.com`, послав запись NSEC, подтверждающую отсутствие любых записей A между именами `bork.atrust.com` и `borrelia.atrust.com`.

К сожалению, включение в записи NSEC конечных точек позволяет обойти зону и выяснить все ее действующие узлы. В версии NSEC3 этот недостаток был устранен путем включения в запись хешированных имен конечных точек, а не самих имен. Правда, это было сделано за счет более интенсивных вычислений: чем выше безопасность, тем ниже производительность. В настоящее время используются как записи NSEC, так и записи NSEC3, и вы должны будете сделать выбор между ними при генерировании своих ключей и подписании своих зон.

Если защита от блуждания по зоне не является критически важной для вашей организации, мы рекомендуем пока использовать запись NSEC. Только последние версии серверов BIND (9.6 и далее) и NSD (3.1 и далее) понимают записи NSEC3.

## Настройка протокола DNSSEC

Поскольку сервер NSD является единственным авторитетным сервером имен, только он должен осуществлять обработку подписанных данных для клиентов, поддерживающих протокол DNSSEC. При этом нет необходимости явно включать протокол DNSSEC. Если зона подписана, сервер NSD использует протокол DNSSEC автоматически.

Сервер BIND является более сложным. Текущие версии BIND не содержат в дистрибутивном наборе пакет OpenSSL, поэтому, если вы хотите использовать протокол DNSSEC, то должны получить либо заранее настроенный пакет, включающий поддержку протокола DNSSEC, либо библиотеки SSL непосредственно с сайта `openssl.org`. Если вы используете вторую возможность, то нужно перекомпилировать сервер BIND, чтобы включить криптографическую поддержку (с помощью опции **`--with-openssl`** команды **`./configure`**). Если вы не сделаете этого, то команда **`dnssec-keygen`** сообщит об ошибке. Однако она по-прежнему будет генерировать ключи TSIG, поскольку для этого не требуется пакет OpenSSL. Если версия пакета OpenSSL слишком стара и уязвима для известных угроз, то сервер BIND откроет красивую страницу с предупреждением.

Использование подписанных зон связано с двумя разными рабочими потоками: один из них создает ключи и подписывает зоны, а второй обслуживает содержимое этих подписанных зон. Эти функции не обязательно реализовывать на одном и том же

компьютере. На самом деле лучше изолировать закрытые ключи и процесс подписания зоны, сильно загружающий центральный процессор, на компьютере, который не доступен из Интернета. (Разумеется, компьютер, обрабатывающий данные, должен быть виден из Интернета.)

На первом этапе настройки протокола DNSSEC следует организовать зонные файлы так, чтобы все файлы данных для зоны находились в одном каталоге. Это необходимо для правильной работы инструментов, управляющих зонами по протоколу DNSSEC.

Затем следует включить протокол DNSSEC на своих серверах с помощью опиций файла **named.conf**.

```
options {
    dnssec-enable yes;
}
```

(для авторитетного сервера) и

```
options {
    dnssec-enable yes;
    dns-validation yes;
}
```

(для рекурсивных серверов). Опция `dnssec-enable` приказывает вашему авторитетному серверу включать подписи наборов записей DNSSEC в свои ответы на запросы, поступающие от серверов имен, работающих по протоколу DNSSEC. Опция `dnssec-validation` заставляет демона **named** проверять легитимность подписей, которые он получает в ответ от других серверов.

## Генерирование пар ключей

Необходимо сгенерировать две пары ключей для каждой зоны, которую хотите подписать, — для подписания зоны (ZSK) и для подписания ключей (KSK). Каждая пара состоит из открытого и закрытого ключей. Закрытый ключ KSK подписывает ключ ZSK и создает безопасную точку входа для зоны. Закрытый ключ ZSK подписывает записи о ресурсах зоны. Открытые ключи затем публикуются, чтобы позволить другим сайтам проверить ваши подписи.

Команды сервера BIND

```
$ dnssec-keygen -a RSASHA1 -b 1024 -n ZONE example.com
Kexample.com.+005+23301
$ dnssec-keygen -a RSASHA1 -b 2048 -n ZONE -f KSK example.com
Kexample.com.+005+00682
```

и команды сервера NSD

```
$ ldns-keygen -a RSASHA1 -b 1024 example.com
Kexample.com.+005+23301
$ ldns-keygen -a RSASHA1 -b 2048 -k example.com
Kexample.com.+005+00682
```

генерируют для сайта `Kexample.com` пару 1024-битовых ключей ZSK, использующую алгоритмы RSA и SHA-1, а также соответствующую пару 2048-битовых ключей KSK.[33] Серьезная проблема, связанная с ограничением на предельный размер пакета UDP, вынуждает отдавать предпочтение коротким ключам для подписания зоны и часто их менять. Для повышения уровня безопасности можно использовать более длинные ключи

[33] 2048 бит — это, конечно, излишество; многие сайты используют 1500 бит и меньше.

для подписания ключей. Ключи генерируются недолго: одна-две минуты для коротких ключей и полчаса и более — для длинных ключей на медленных устаревших ноутбуках.

Оба генератора ключей печатают в стандартном потоке вывода базовое имя файла для сгенерированного ключа. В нашем примере **example.com** — это имя ключа, **005** — идентификатор набора алгоритмов RSA/SHA-1, а **23301** и **00682** — хешированные значения, которые называются идентификаторами ключей, слепками ключей или дескрипторами ключей.[34] При каждом запуске генератор ключей в пакете BIND создает два файла (**.key** и **.private**), а генератор ключей на сервере NDS — три файла (**.key**, **.private** и **.ds**).

```
Kexample.com+005+23301.key          # Открытый ключ для подписи зоны
Kexample.com+005+23301.private      # Закрытый ключ для подписи зоны
Kexample.com+005+23301.ds           # Запись DS для ключа ZSK (только NSD)

Kexample.com+005+00682.key          # Открытый ключ для подписи ключа
Kexample.com+005+00682.private      # Закрытый ключ для подписи ключа
Kexample.com+005+00682.ds           # Запись DS для ключа KSK (только NSD)
```

Существует несколько алгоритмов шифрования, каждый из которых имеет свой диапазон длин ключей. Для того чтобы увидеть список доступных алгоритмов, можно выполнить команду **dnssek-keygen** без аргументов или команду **ldns-keygen -a list**. Как сервер BIND, так и сервер NSD могут использовать ключи, сгенерированные другим программным обеспечением.

В зависимости от версии вашего программного обеспечения, имена некоторых доступных алгоритмов могут содержать имя NSEC3 в качестве префикса или окончания. Если вы хотите использовать записи NSEC3, а не NSEC для подписания отрицательных ответов, должны сгенерировать NSEC3-совместимые ключи одним из NSEC3-совместимых алгоритмов; подробности можно найти на справочных страницах, посвященных командам **ldns-signzone** или **dnssec-keygen**.

Каждый файл **.key** содержит по одной записи о ресурсах DNSSEC для сайта `example.com`. Например, ниже приведен открытый ключ для подписания зоны, усеченный по ширине страницы. Его можно назвать ключом ZSK, потому что поле файлов равно 256, а не 257, как для ключей KSK.

```
example.com   IN  DNSKEY  256  3  5  AwEAAex7tHe60w5va8sPpnRe4RX8MgI...
```

Эти открытые ключи должны быть вставлены в зонные файлы с помощью директивы `$INCLUDE` или другим способом либо в конец, либо сразу после записи SOA. Для того чтобы скопировать ключи в зонный файл, можно добавить их с помощью команды **cat**[35] и вставить с помощью текстового редактора.

Файлы **.ds**, созданные генератором ключей **ldns-keygen** на сервере NSD, содержат записи DS. Если протокол DNSSEC развернут полностью, то файл, соответствующий ключам KSK, должен храниться в родительской зоне. Запись DS может генерироваться на основе записи о ресурсах DNSKEY, соответствующей ключам KSK. Некоторые подписанные зоны требуют наличия именно таких записей, вместо или в дополнение к записи DNSKEY для ключей KSK. Запись DS может выглядеть следующим образом.

---

[34] Для того чтобы сравнивать процессы на серверах BIND и NSD было проще, мы поступили некорректно и сделали слепки ключей одинаковыми. В реальном мире каждый ключ должен иметь отдельный слепок.

[35] Используйте команды наподобие ***cat Kexmaple.com+*.key>>zonefile***. Операция >> означает добавление ключа в файл ***zonefile***,а не его замену, как операция >.

```
example.com   3600  IN DS  23301  1  1  5bd844108f8d8fea341b3bc2f2135e...
example.com   3600  IN DS  00682  1  1  0dbf80886b7168633ff8273255de09...
```

В идеале, закрытая часть любой пары ключей должна храниться в компьютере, отключенном от сети, или хотя бы в компьютере, недоступном из Интернета. Для динамически обновляемых зон это требование практически невыполнимо, а для ключей, предназначенных для подписания зон, еще и непрактично. Тем не менее для ключей, предназначенных для подписания других ключей, это требование абсолютно разумно, поскольку эти ключи имеют долгий срок действия. Подумайте о скрытом главном сервере, недоступном извне для ключей ZSK. Распечатайте закрытый ключ KSK или напишите его на флешке, а затем спрячьте в сейфе, пока он не потребуется в следующий раз.

"Заперев" новые закрытые ключи, целесообразно записать информацию о новых ключах в системный файл ключей. При этом не обязательно записывать туда сами ключи. Достаточно записать их идентификаторы, алгоритмы, дату, назначение и т.д.

По умолчанию срок действия подписи ограничен одним месяцем для записей RRSIG (ZSK-подписи наборов записей о ресурсах) и тремя месяцами для записей DNSSIG (KSK-подписи ключей ZSK). В настоящее время рекомендуется использовать 1024-битовые ключи ZSK от трех месяцев до одного года, а 1280-битовые ключи KSK — от одного до двух лет.[36] Поскольку рекомендуемые сроки действия ключей дольше, чем сроки, установленные для них по умолчанию, эти параметры необходимо указывать явно при подписании или периодическом переподписании зон, даже если сами ключи не изменились.

## Подписание зоны

Итак, теперь, когда у вас есть ключи, вы можете подписывать зоны с помощью команд **dnssec-signzone** (сервер BIND) или **ldns-signzone** (сервер NDS), добавляющих записи RRSIG и NSEC или NSEC3 в каждый набор записей о ресурсах. Эти команды считывают оригинальный зонный файл и создают отдельную подписанную копию с именем *зонный_файл*.**signed**.

Синтаксис команды **dnssec-signzone** для сервера BIND имеет следующий вид.

```
dnssec-signzone [-o имя_зоны] [-N increment]
   [-k KSK-файл] зонный_файл [ZSK-файл]
```

Здесь параметр *имя_зоны* по умолчанию равен параметру *зонный_файл*, а поля ключей по умолчанию совпадают с именами файлов, созданных командой **dnssec-keygen**, как описано выше.

Если вы называете свои зонные файлы по именам зон и сохраняете имена оригинальных файлов ключей, то команда упрощается.

```
dnssec-signzone [-N increment] зонный_файл
```

Флаг `-N increment` автоматически увеличивает порядковый номер в записи SOA, так что забыть его невозможно. Кроме того, можно указать значение **unixtime**, чтобы установить порядковый номер равным текущему времени UNIX (количеству секунд, прошедших с 1 января 1970 года), или значение **keep**, чтобы предотвратить изменение оригинального порядкового номера командой **dnssec-keygen**. Порядковый номер увеличивается в файле подписанной зоны, но не в оригинальном файле зоны.

Рассмотрим пример, в котором используются сгенерированные ранее ключи.

[36] Рекомендации разных организаций, касающиеся длин криптографических ключей, приведены на веб-сайте `keylength.com`.

```
$ sudo dnssec-signzone -o example.com -increment
  -k Kexample.com+005+00682 example.com Kexample.com+005+23301
```

Подписанный файл упорядочивается в алфавитном порядке и содержит записи DNSKEY, добавленные вручную, а также записи RRSIG и NSEC, сгенерированные в ходе подписания. Порядковый номер зоны увеличен.

Если вы сгенерировали свои ключи с помощью алгоритма NSEC3RSASHA1, то должны подписать зону так, как показано выше, но указав флаг **-3** *salt*.

Перечислим другие полезные опции команды **dnssec-signzone**.

- **-g** — генерирование записи (записей) DS, которая должна быть включена в родительскую зону;
- **-l** — генерирование записи (записей) DLV, которая используется, если родительская зона не подписана;
- **-s** *начальный_момент* — установка момента времени, с которого подписи считаются действительными;
- **-e** *конечный_момент* — установка момента времени, после которого подписи считаются недействительными;
- **-t** — вывод статистических показателей.

Данные о сроках действия подписей можно выразить в виде абсолютного времени в формате *ггггммддччммсс* или относительного времени, считая от текущего момента, в формате +*N*, где *N* — количество секунд. По умолчанию период действия подписи изменяется от одного часа в прошлое до 30 дней в будущее. Рассмотрим пример, в котором мы указываем, что подписи должны быть действительными до окончания календарного 2010 года.

```
$ dnssec-signzone -N increment -e 20101231235959 example.com
```

На сервере NSD синтаксис подписания зоны выглядит следующим образом.

**ldns-signzone** [**-o** *имя_зоны*] *имя_зоны ключ* [*ключ ...*]

Вы можете просто указать оба ключа и разрешить команде **ldns-signzone** самой выяснить, какой из них является ключом KSK, а какой — ключом ZDK. Рассмотрим пример.

```
$ sudo ldns-signzone example.com Kexample.com.+005+00682
  Kexample.com.+005+23301
```

Как и с командой **dnssec-signzone**, вы можете установить предельный срок действия подписей, используя флаг **-e** *ггггммдд*. Для генерирования записей NSEC3 вместо записей NSEC для подписания пробелов, используется флаг **-n -s** *salt*.

Размеры файлов подписанных зон от четырех до десяти раз больше, чем размеры файлов исходных зон, и все попытки навести логический порядок напрасны. Строка наподобие

```
mail-relay                  A    63.173.189.2
```

превращается в несколько следующих строк.

```
mail-relay.example.com.    57600  A     63.173.189.2
                           57600  RRSIG A  5  3  57600  20090722234636 (
                                  20090622234636 23301 example.com.
                                  Y7s9jDWYuuXvozeU7zGRdFCl+rzU8cLiwoev
                                  OI2TGfL1bhsRgJfkpEYFVRUB7kKVRNguEYwk
                                  d21RSkDJ9QzRQ+w==)
```

```
3600  NSEC  mail-relay2.example.com. A RRSIG NSEC
3600  RRSIG NSEC 5 3 3600 20090722234636 (
            20090722234636 23301 example.com.
            WNsN84hF0notymRxZRIZypqWzLIPBZAUJ77R
            HP0hLfBDoqmZYw== )
```

С практической точки зрения файл подписанной зоны больше нельзя назвать понятным для человека и его невозможно отредактировать вручную из-за записей RRSIG и NSEC или NSEC3. Этот файл не содержит ни одного фрагмента, который пользователь мог бы изменить вручную!

За исключением записей DNSSEC, каждый набор записей о ресурсах (ресурсных записей с одинаковым типом и именем) получает одну подпись от ключа ZSK. Записи о ресурсах DNSSEC подписываются как ключом ZSK, так и ключом KSK, поэтому они содержат две записи RRSIG. Тем не менее 64-разрядное представление подписи заканчивается несколькими знаками равенства, потому что длина записи должны быть кратной четырем.

Для того чтобы последующие примеры стали яснее, предположим, что зонный файл называется так же, как и зона, причем зонные файлы и ключевые файлы располагаются в одном и том же каталоге. В реальной жизни действительно целесообразно задавать файлы ключей явно, особенно если вы часто меняете ключи и должны быть уверены, что команды используют правильные ключи.

Как только ваша зона подписана, остается лишь указать ваш сервер имен в подписанных версиях зонных файлов. Если вы используете сервер BIND, поищите инструкцию `zone`, соответствующую каждой зоне в файле **`named.conf`**, и измените параметр `file` с `example.com` на `example.com.signed`. Для сервера NSD соответствующий файл конфигурации называется **`nsd.conf`**, и вы должны найти в нем строки `zonefiles`.

В заключение перезапустите демон сервера имен, заставив его прочитать снова свой файл конфигурации. Для сервера BIND следует выполнить команды **`sudo rndc reconfig`** и **`sudo rndc flush`**, а для сервера NSD — команды **`sudo nsdc rebuild`** и **`sudo nsdc restart`**.

Теперь мы обслуживаем подписанную зону DNSSEC! Для того чтобы внести изменения, вы можете отредактировать либо исходный неподписанный зонный файл, либо подписанный зонный файл, а затем переподписать зону. Редактирование подписанной зоны иногда оказывается чрезвычайно сложной задачей, но это проще, чем повторное подписание всей зоны. Удалите записи RRSIG, соответствующие всем изменяемым записям. Для того чтобы избежать путаницы между версиями, вероятно, следует внести идентичные изменения в неподписанную зону.

При передаче подписанной зоны в качестве аргумента команде **`dnssec-signzone`** или **`ldns-signzone`** все неподписанные записи подписываются, а подписи всех записей, срок действия которых подходит к концу, обновляются. Выражение "срок действия подходит к концу" означает, что прошло три четверти периода действия подписи. Повторное подписание, как правило, приводит к изменениям, поэтому следует изменить порядковый номер зоны вручную или автоматически с помощью сервера BIND, используя раздел **`-N increment`** в командной строке **`dnssec-signzone`**.

Это все, что касается локальной части конфигурации протокола DNSSEC. Осталось только решить трудную проблему: соединить наш безопасный "DNS-островок" с другими надежными и подписанными частями иерархии DNS. Мы должны либо передать наши записи DS в подписанную родительскую зону, либо использовать динамическую проверку доменов. Решение этих задач описывается в следующем разделе.

## Цепочка доверия в протоколе DNSSEC

Продолжим наш пример, связанный с настройкой протокола DNSSEC. Сайт `example.com` теперь подписан, и его серверы имен поддерживают протокол DNSSEC. Это значит, что при отправке запросов они используют протокол EDNS0, расширенный протокол DNS, а в DNS-заголовке пакета устанавливают опцию, включающую протокол DNS. Отвечая на запросы, которые приходят с такой комбинацией установленных битов, они включают в свой ответ подписанные данные.

Клиент, получающий подписанные запросы, может оценить корректность ответа, проверив его подпись с помощью соответствующего открытого ключа. Однако он получает свой ключ из своей собственной зонной записи DNSKEY, что, если вдуматься, довольно подозрительно. Что может помешать постороннему лицу предоставить ложные записи и ложный публичный ключ, чтобы пройти проверку?

Есть несколько вариантов ответа на этот вопрос. Ваш сайт должен реализовать хотя бы один из них. В противном случае вся ваша работа по настройке протокола DNSSEC пойдет насмарку.

Каноническое решение заключается в том, чтобы передать вашей родительской зоне запись DS, которую она должна включить в свой зонный файл. Запись DS, приходящая из родительской зоны, сертифицируется родительским закрытым ключом. Если клиент доверяет своей родительской зоне, он должен верить в то, что запись DS родительской зоны правильно отражает открытый ключ вашей зоны.

В свою очередь, родительская зона сертифицируется своей родительской зоной и так вплоть до корня. После развертывания протокола DNSSEC в полном масштабе единственным ключом, который вы должны будете знать заранее, будет открытый ключ, используемый для подписи корня. Этот ключ может находиться в файле подсказок корня и использоваться во всем процессе функционирования системы DNS.

Если вам повезло и ваша родительская зона подписана, просто передайте ее администратору свою запись DS и запись DNSKEY, предназначенную для подписи ключей и использованную для подписи записи DS.[37] Флаг **`-g`** команды **`dnssec-signzone`** в пакете BIND генерирует файлы **`dsset-domain`** и **`keyset-domain`**, которые можно безопасно передать родительской зоне, чтобы она добавила их в свой зонный файл. Аналогично команда **`ldns-keygen`** генерирует требуемую DS-запись в файле **`.ds`** и запись DNSKEY в файле **`.key`** во время генерирования ключей. Помните, что вы должны опубликовать свою запись DNSKEY в своей зоне до того, как родительская зона инсталлирует соответствующую запись DS.

Если ваша родительская зона не подписана, необходимо как-то иначе доказать внешнему миру, что открытый ключ, опубликованный в вашей системе DNS, действительно принадлежит вам. Это можно сделать тремя разными способами.

- Опубликуйте свой открытый ключ с помощью одного или нескольких репозиториев доверительных точек (trusted anchor repository — TAR); например, с помощью репозитория SecSpider. Для того чтобы использовать ключи TAR на своих серверах, получите список ключей от репозиториев SecSpider, ITAR (поддерживается организацией IANA и содержит только домены TLD) или RIPE-NCC (содержит только свои собственные зоны, в основном европейские домены TLD и реверсные зоны). Поместите эти ключи в раздел `trusted-keys` в конфигурационном файле своего сервера имен.

[37] Как узнать, подписана ли родительская зона? Выполните команду ***`dig + dnssec`*** или ***`drill -D`***.

- Используйте сервер динамической проверки доменов, например сервер, поддерживаемый консорциумом ISC (Internet Systems Consortium). По существу, этот сервер делает сайт `isc.org` вашим приемным DNS-родителем. Это легко и просто; см. `isc.org/ops/dlv`. Существуют и другие серверы DLV, но поскольку использовать можно только один из них, а сервер консорциума ISC очень хорошо продуман и превосходно работает, предпочесть следует именно его.
- Используйте демон Vantages (`vantage-points.org`), который взаимодействует со своими копиями, принадлежащими друзьям, которым вы доверяете, формируя тем самым социальную сеть демонов. Эти демоны могут получать ключи от других сайтов в Интернете совершенно независимо друг от друга и сравнивать их, чтобы определить, являются ли они аутентичными.

Решение, связанное с использованием серверов DLV, мы опишем более подробно в следующем разделе. Однако помните, что все три способа, перечисленные выше, являются временными решениями, которые предназначены для того, чтобы облегчить развертывание протокола DNSSEC. Если ваша родительская зона уже подписана, о них можно даже не упоминать. Просто передайте вашей родительской зоне свою запись DS.

## Сервер DLV: динамическая проверка доменов

Кеширующий сервер, поддерживающий протокол DNSSEC и получающий подписанный ответ на запрос, сначала проверяет, совпадает ли подпись записи с открытым ключом домена, заданным его записью DNSSEC. Затем клиент пытается проверить сам ключ, просматривая родительскую зону в поисках записи DS. Если запись DS недоступна, клиент ищет запись DLV в исходном домене; эта запись переадресовывает его на сайт `dlv.isc.org` или другой сервер DLV, который действует как родитель зоны. Как только клиент получит запись DLV от сайта `dlv.isc.org`, он сможет проверить всю цепочку доверия. Следовательно, настройка службы DLV для вашей зоны сводится к генерированию правильных записей DLV и размещению их в правильных местах.

Запись DLV на самом деле представляет собой замаскированную запись DS. Типы этих записей отличаются, но тела совпадают. Поле *имя* в записи также изменяется, чтобы поместить запись в зоне провайдера службы DLV. Например, имя `example.com` может стать `example.com.dlv.isc.org`.

В пакете BIND команда **`dnssec-signzone -l`** (строчная буква L) генерирует запись DLV.

```
$ sudo dnssec-signzone -l dlv.isc.org example.com
```

Эта команда повторно подписывает зону и создает файл с именем **`dlvset-example.com`**, содержащий запись DLV, предназначенную для зоны провайдера DLV.

Пользователи серверов NSD/Unbound должны генерировать записи DLV самостоятельно. Скопируйте файл **`.ds`**, созданный во время генерирования ключа KSK и изменения типа записи с DS на DLV. Затем измените поле *имя*. Например, измените запись

```
example.com              3600  IN  DS   25069  1  1  0dbf80886b716863de09...
```

на

```
example.com.dlv.isc.org  3600  IN  DLV  25069  1  1  0dbf80886b716863de09...
```

Внесенные изменения выделены полужирным шрифтом.

Собрав запись DLV и файлы ключей, использованных для подписания вашей зоны, зайдите на веб-страницу `dlv.isc.org` и выполните инструкции, благодаря которым сервер `isc.org` станет вашим DLV-сервером. Консорциум ISC заставит вас проходить

несколько этапов проверки, чтобы убедиться, что вы действительно владеете своим доменом, имеете право им управлять и безопасно предоставили его открытый ключ. Однако это не сложная процедура.

Консорциум ISC вставит несколько строк в раздел `trusted-keys` вашего файла **named.conf**.

```
trusted-keys {
  dlv.isc.com 257 3 5 "хешированная информация о ключе";
  dlv.isc.com 257 3 5 "хешированная информация о другом ключе";
  ...
}
```

Пользователи сервера BIND тоже добавляют одну строку в раздел `option` файла **named.conf**.

```
dnssec-lookaside "." trusted-anchor "dlv.isc.org"
```

Пользователи сервера NSD должны добавить запись DLV в зону и заново подписать зону. Для того чтобы подключить проверку с помощью записи DLV и сервера **unbound**, получите KSK-запись DNSKEY с веб-сайта `dlv.isc.org` или сайта SecSpider и проверьте их подписи. (Программа SecSpider проверяет согласованность ключей, полученных из разных мест.) Не следует просто применять команду **dig** к ключу. Это небезопасно, пока протокол DNSSEC не развернут в полном масштабе. Поместите ключ в файл, находящийся в рабочем каталоге сервера **unbound**, например в каталоге **dlv.isc.org.key**, и добавьте строку

```
dlv-anchor-file: "dlv.isc.org.key"
```

в раздел `server` файла **unbound.conf**.

## Смена ключей DNSSEC

Для протокола DNSSEC смена ключей всегда была трудной задачей. Его исходные спецификации были, по существу, изменены, для того чтобы решить проблемы, связанные с обеспечением взаимодействия между родительской и дочерней зонами для создания, изменения или удаления ключей. Новые спецификации называются DNSSEC-bis.

Смена ключей ZSK представляет собой относительно несложную задачу и не предусматривает наличия родительской зоны или какой-либо точки доверия. Единственным "скользким" местом является расписание. Ключи имеют определенный срок действия, поэтому их замену необходимо производить до истечения этого срока. Однако ключи имеют период TTL, определенный в зонном файле. Для иллюстрации предположим, что период TTL равен одному дню и что ключи на следующей неделе еще будут действовать. Придется выполнить следующие действия.

- Сгенерировать новый ключ ZSK.
- Включить его в зонный файл.
- Подписать или заново подписать зону с помощью ключа KSK или *старого* ключа ZSK.
- Попросить сервер имен перезагрузить зону; теперь в ней будет действовать новый ключ.
- Подождать 24 часа (период TTL); теперь все могут использовать как старый ключ, так и новый.
- Подписать зону снова с помощью ключа KSK и *нового* ключа ZSK.

- Попросить сервер имен перезагрузить зону.
- Подождать 24 часа; теперь все имеют новую зону.
- Удалить старый ключ ZSK, например, при следующем изменении зоны.

Эта схема называется предварительной публикацией (prepublishing). Совершенно очевидно, что до того, пока все будут использовать новый ключ, приходится запускать этот процесс как минимум дважды после истечения двух периодов TTL. Эти периоды ожидания гарантируют, что любой сайт с кешированными значениями всегда будет иметь кешированный ключ, соответствующий этим кешированным данным.

Еще одной переменной, которая влияет на этот процесс, является время, которое потребуется вашему самому слабому подчиненному серверу для обновления своей копии в вашей зоне после получения уведомления от главного сервера. По этой причине не следует ждать до последней минуты, чтобы начать процесс смены ключей или повторного подписания зон, срок действия подписей которых истек. Просроченные подписи недействительны, поэтому сайты, проверяющие подписи DNSSEC, не смогут выполнить поиск для вашего домена.

Механизм смены ключей KSK называется двойным подписанием (double signing) и также довольно прост. Однако вам придется переслать вашу новую запись DS родительской зоне или послать запись DLV своему "суррогатному родителю". Убедитесь, что родительская зона или репозиторий точек доверия вас признали, прежде чем переключиться на новый ключ. Процесс смены ключей KSK состоит из следующих этапов.

- Создать новый ключ KSK.
- Включить его в зонный файл.
- Подписать зону и новым, и старым ключами KSK и ZSK.
- Попросить сервер имен перезагрузить зону.
- Подождать 24 часа (период TTL); теперь у всех есть новый ключ.
- Уведомить все точки доверия о новом значении KSK.
- После подтверждения удалить старую запись KSK из зоны.
- Заново подписать зону новыми ключами KSK и ZSK.

## Инструменты DNSSEC

Кроме комплектов инструментов, входящих в дистрибутивные наборы пакетов BIND и NSD/Unbound, есть еще четыре комплекта инструментов для развертывания протокола DNSSEC и тестирования существующих настроек: **ldns**, Sparta, RIPE и Vantages. По крайней мере, еще два комплекта инструментов находятся на этапе разработки: OpenDNSSEC (`opendnssec.org`) и DNSSHIM. Инструмент OpenDNSSEC должен устранить всю путаницу и сложность, автоматически возникающую при использовании протокола DNSSEC, что звучит обнадеживающе. Инструмент DNSSHIM — это реализация авторитетного сервера DNS с автоматической конфигурацией подчиненных серверов и программами для работы по протоколу DNSSEC, написанными на языках Java и Python.

### *Инструменты ldns, nlnetlabs.nl/projects/ldns*

По словам сотрудников компании NLnet Labs, **ldns** — это библиотека программ для разработки инструментов DNS, включающая примеры, демонстрирующие использование этой библиотеки. Эти инструменты перечислены ниже вместе с кратким описани-

ем каждого из них. Все они находятся в каталоге **examples**, за исключением утилиты **drill**, имеющей свой собственный каталог в дистрибутивном пакете. Команды сопровождаются справочными страницами. Файл **README**, относящийся к верхнему уровню иерархии каталогов, содержит очень краткие инструкции по инсталляции.

- **ldns-keygen** генерирует пары ключей TSIG и DNSSEC.
- **ldns-signzone** подписывает зонный файл с записью NSEC или NSEC3.
- **ldns-verify-zone** проверяет состояние записей RRSIG, NSEC или NSEC3.
- **ldns-key2ds** преобразовывает запись DNSSEC в запись DS.
- **ldns-rrsig** распечатывает в удобном для чтения виде даты истечения сроков действия ключей из записей RRSIG.
- **ldns-nsec3-hash** распечатывает запись NSEC для заданного имени в хешированном виде.
- **ldns-revoke** устанавливает флаг отмены для ключа RR в протоколе DNSSEC (см. документ RFC54011).
- **ldns-chaos** показывает идентификатор сервера имен, хранящийся в классе CHAOS.
- **ldns-keyfetcher** извлекает открытые ключи DNSSEC для зон.
- **ldns-read-zone** читает зону и распечатывает ее в разных форматах.
- **ldns-update** посылает пакет динамического обновления.
- **ldns-walk** совершает обход зоны, используя записи NSEC протокола DNSSEC.
- **ldns-zsplit** разбивает зону на фрагменты, чтобы подписывать их параллельно.
- **ldns-zcat** объединяет зонные файлы, разбитые утилитой **ldns-zsplit** на фрагменты.
- **ldns-compare-zones** демонстрирует разницу между двумя зонными файлами.
- **ldns-notify** проверяет обновления подчиненных серверов зон.
- **ldns-dpa** анализирует пакеты DNS с помощью файлов слежения **tcpdump**.

Многие из этих инструментов очень просты и выполняют только одну рутинную операцию для системы DNS. Они были написаны в качестве примеров использования библиотеки **ldns** и демонстрируют, насколько простым становится код, когда библиотека берет всю тяжелую работу на себя.

### *Инструменты Sparta, dnssec-tools.org*

Комплект инструментов Sparta создан на основе инструментов сервера BIND, предназначенных для поддержки протокола DNSSEC, и включает в себя следующие команды.

- **zonesigner** генерирует ключи и подписи зон.
- **donuts** анализирует зонные файлы в поисках ошибок и несоответствий.
- **donutsd** выполняет команду **donuts** через определенные интервалы времени и предупреждает о проблемах.
- **rollerd**, **rollctl** и **rollinit** осуществляют автоматическую смену ключей, используя схему предварительной публикации для ключей ZSK и метод двойного подписания для ключей KSK.
- **trustman** управляет точками доверия и включает реализацию смены ключей RFC5011.

- **dnspktflow** отслеживает поток пакетов DNS в последовательности запросов и ответов, перехваченных командой **tcpdump**, и создает диаграмму.
- **mapper** отображает зонные файлы, демонстрируя защищенные и незащищенные фрагменты.
- **validate** — инструмент для проверки подписи из командной строки.

Веб-сайт содержит хорошую документацию и учебные пособия для всех этих инструментов. Исходный код доступен для загрузки и защищен лицензией BSD.

Компания Sparta поддерживает библиотеки DNSSEC, написанные на языке Perl и распространяемые в сети CPAN (Comprehensive Perl Archive Network). Кроме того, она распространяет "заплатки" для нескольких популярных пакетов программного обеспечения (включая Firefox, Thunderbild, Postfix, **sendmail**, **libSPF** и OpenSSH), чтобы обеспечить их более полную совместимость с протоколом DNSSEC.

### *Инструменты RIPE, ripe.net*

Инструменты компании RIPE функционируют как внешний компонент инструментов пакета BIND, предназначенных для поддержки протокола DNSSEC, и основное внимание уделяют вопросам управления. Их подробные сообщения, при выполнении и упаковке многих аргументов и команд, имеют более понятную форму.

### *Инструменты Vantages, vantage-points.org*

Vantages — это каркас для распределенного мониторинга, база которого находится в университете штата Колорадо (Colorado State University). В настоящее время он сосредоточен на операционных вопросах, связанных с протоколом DNSSEC. Инструменты Vantages могут помочь при развертывании протокола DNSSEC.

Главным продуктом проекта является инструмент **vantaged**, демон, собирающий записи DNSSEC и сравнивающий их значения с аналогичными записями, полученными от других демонов **vantaged**, находящихся в Интернете. Если множество демонов **vantaged** дают один и тот же ответ, то ключ считается правильным. Для того чтобы достичь этого результата обманным путем, злоумышленник должен был бы заставить все сайты, выполняющие программное обеспечение Vantage, дать неправильный ответ. Демон **vantaged** собирает ключи от источников DNS, HTT и HTTPS и классифицирует их по четырем состояниям: подтвержденный, временный, неизвестный и конфликтный. Он добавляет подтвержденные ключи в раздел `trusted-keys` в файле **named.conf**.

Инструменты Vantages имеют несколько дополнительных функций.

- **d-sync** — отслеживает согласованность ключей в записях DS между родительской и дочерней зонами, что особенно полезно при смене ключей.
- **dnsfunnel** определяет максимальную пропускную способность линии связи между вами и любым другим сайтом. Он напоминает инструмент **traceroute**.
- **dnskey-grab** получает ключи DNSKEY для зоны от ее авторитетных серверов.

## Отладка протокола DNSSEC

Протокол DNSSEC был разработан для того, чтобы обеспечить взаимодействие между подписанными и неподписанными зонами, а также между серверами имен, поддерживающими протокол DNSSEC и игнорирующими его. Следовательно, возможно постепенное развертывание протокола, что часто и происходит, хотя и не всегда.

DNSSEC — это распределенная система с многочисленными изменчивыми частями. Все проблемы порождают авторитетные серверы, распознаватели клиентов и линии связей между ними. Проблема, которая кажется локальной, может отразиться на удаленном пользователе, поэтому такие инструменты, как SecSpider и Vantages, отслеживающие распределенное состояние системы, могут оказаться очень полезными. Эти инструменты, а также утилиты, перечисленные выше, и регистрационные файлы вашего сервера имен являются основными орудиями отладки.

Сначала убедитесь, что вы записали категорию регистрации по протоколу DNSSEC из файла **named.conf** в файл на локальном компьютере. Полезно отделить все сообщения, связанные с протоколом DNSSEC, чтобы не записывать в этот файл никаких других категорий регистрации. Рассмотрим пример регистрационной спецификации для демона **named**.

```
channel dnssec-log {
   file "/var/log/named/dnssec.log" versions 4 size 10m ;
   print-time yes ;
   print-category yes ;
   print-severity yes ;
   severity debug 3;
}
category dnasec { dnssec-log;}
```

В пакете BIND следует установить уровень отладки равным 3 или выше, чтобы увидеть этапы проверки, выполняемые рекурсивным сервером BIND при попытках проверить подпись. Этому уровню регистрации соответствует две страницы регистрационной информации для одной проверяемой подписи. Если вы отслеживаете работу загруженного сервера, то регистрационная информация от нескольких запросов будет перемежаться другими данными. Разбираться в этой путанице — очень сложная и утомительная задача.

Для серверов NSD и Unbound следует установить уровень детальности сообщений в их конфигурационных файлах выше, чем заданный по умолчанию (0 и 1 соответственно). Для сервера Unbound достаточно настроить уровень детальности сообщений "на лету" с помощью команды **verbosity** *уровень* в команде **unbound-control**. Подобно серверу BIND, уровень отладки на сервере Unbound должен быть установлен равным 3, чтобы продемонстрировать этапы проверки подписи.

Если все работает хорошо, установите параметр `val-log-level` на сервере Unbound равным `1`, чтобы распечатывать сообщения об ошибках для каждой неправильной подписи в виде одной строки. Этот уровень детальности поможет вам найти сайты, вызывающие проблемы. Затем вы можете отследить причины проблем для каждой подписи с помощью либо команды **drill**, либо команды **unbound-host -v -d** (иди даже **-dddd**, чтобы получить больше отладочной информации). При этом командам **drill** и **unbound-host** необходимо передать релевантные открытые ключи.

Команда **drill** имеет два особенно полезных флага: **-T** — для отслеживания цепочки доверия от корня до указанного узла и **-S** — для отслеживания подписей от указанного узла обратно к корню. Рассмотрим практический пример вывода, полученного от команды **drill -S**, позаимствованный из документа DNSSEC HOWTO компании NLnet Labs.

```
$ drill -S -k ksk.keyfile example.net SOA
DNSSEC Trust tree:
example.net (SOA)
|---example.net. (DNSKEY keytag: 17000)
    |---example.net. (DNSKEY keytag: 49656)
```

```
|---example.net. (DS keytag: 49656)
    |---net. (DNSKEY KEYTAG: 62972)
       . |---net. (DNSKEY KEYTAG: 13467)
         |---net. (DS KEYTAG: 13467)
             |---. (DNSKEY KEYTAG: 63380)
                 |---net. (DNSKEY KEYTAG: 63276) ;; Chase successful
```

Если сервер имен, выполняющий проверку, не может проверить подпись, он возвращает сигнал SERVFAIL. Проблема может заключаться в неправильной конфигурации одной из зон, входящих в цепочку доверия, в фальшивых данных, поступивших от злоумышленника, или в настройке самого рекурсивного сервера, выполняющего проверку. Попробуйте выполнить команду `drill` и отследить подписи вдоль цепочки доверия, чтобы обнаружить источник проблемы. Если все подписи окажутся верифицированными, то попытайтесь послать запрос проблемному сайту с помощью команды `dig`, а затем `dig +cd`. (Флаг `cd` отключает проверку подписей.) Примените их к каждой зоне, входящей в цепочку доверия, чтобы выяснить, в чем проблема. Вы можете перемещаться по цепочке доверия как вверх, так и вниз. Чаще всего причиной ошибки становятся устаревшие точки доверия или просроченные подписи.

## 17.14. Microsoft и DNS

Многие годы консорциум ISC и группа разработчиков сервера BIND стремились обеспечить взаимодействие с инструментами DNS, разработанными компанией Microsoft и ее службой Active Directory. Компанию Microsoft обвиняли в преднамеренном отклонении от стандартов и отказе от документирования используемых ею расширений протоколов. Однако оказалось, что компания Microsoft на самом деле не пыталась избежать совместимости; ее специалисты просто были несколько некомпетентными и работали с неправильным программным обеспечением (их собственным кодировщиком ASN.1 и синтаксическим анализатором), которое искажало пакеты настолько, что сервер BIND просто не мог в них разобраться. Теперь все в порядке. Ошибки исправлены, и сервер BIND совместно с компанией Microsoft следуют протоколу IETF и могут взаимодействовать. Это хорошие новости.

Плохая новость заключается в том, что служба Active Directory тесно интегрирована с протоколами Kerberos и LDAP и следует своими собственными извилистыми путями (которые похожи друг на друга!) Замена любого из фрагментов, например центра распределения ключей Kerberos, на совместимую реализацию с открытым исходным кодом обречена на неудачу. Сервер BIND может выполнять аутентификацию с помощью службы Active Directory, используя протокол GSS-TSIG, но авторизация остается практически невозможной, потому что служба Active Directory хранит всю информацию в базе данных LDAP.

Советы, касающиеся обеспечения взаимодействия со службой Active Directory, можно найти в главе 30.

## 17.15. Тестирование и отладка

Серверы BIND и NSD/Unbound предоставляют три основных инструмента для отладки: журнальная регистрация, управляющая программа и инструмент запросов их командной строки. Наиболее глубоко разработанными, но и наиболее сложными являются инструменты сервера BIND.

# Журнальная регистрация в пакете BIND

🕮 Система Syslog описывается в главе 11.

Возможности подсистемы журнальной регистрации демона **named** действительно впечатляют. Сервер BIND использует систему Syslog для записи сообщений об ошибках и аномалиях. В новейших версиях концепция журнальной регистрации обобщена: добавлен еще один уровень переадресации и поддерживается направление сообщений непосредственно в файлы. Прежде чем переходить к деталям, приведем мини-словарь терминов, связанных с журнальной регистрацией в пакете BIND (табл. 17.13).

**Таблица 17.13. Лексикон пакета BIND**

| Термин | Что означает |
|---|---|
| Канал | Место, куда направляются сообщения, — система Syslog, файл или устройство **/dev/null** [a] |
| Категория | Класс сообщений, генерируемых демоном **named**, например сообщения о динамических обновлениях или об ответах на запросы |
| Модуль | Имя исходного модуля, который сгенерировал сообщение |
| Средство | Название средства системы Syslog; за DNS не закреплено собственное средство, но можно выбрать любое стандартное |
| Важность | Степень важности сообщения об ошибке |

[a] **/dev/null/** — псевдоустройство, отбрасывающее все входящую информацию.

Подсистема журнальной регистрации конфигурируется при помощи инструкции `logging` в файле **named.conf**. Сначала определяются каналы — возможные пункты доставки сообщений. Затем для различных категорий сообщений задаются каналы, куда они будут поступать.

При формировании сообщения ему назначаются категория, модуль и уровень важности. После этого оно рассылается по всем связанным с категорией и модулем каналам. В каждом канале имеется фильтр, определяющий, сообщения какого уровня важности можно пропускать. Каналы, ведущие в систему Syslog, подвергаются дополнительной фильтрации в соответствии с правилами, установленными в файле **/etc/syslog.conf**.

Общий вид инструкции `logging` таков.

```
logging {
     определение_канала
     определение_канала
  ...
     category имя_категории {
        имя_канала
        имя_канала
     ...
     };
};
```

## Каналы

Аргумент *определение_канала* выглядит по-разному, в зависимости от того, ведет ли он к файлу или к системе Syslog. Для каждого канала необходимо выбрать либо тип `file`, либо тип `syslog`; совмещать их канал не может.

```
channel имя_канала {

    file путь [versions число_версий | unlimited] [size размер];
    syslog средство;
    severity важность;

    print-category yes | no;
    print-severity yes | no;
    print-time yes | no;
};
```

Для файлового канала аргумент *число_версий* сообщает о том, сколько резервных копий файла нужно хранить. Аргумент *размер* задает предельно допустимый размер файла (например, `2048`, `100k`, `20m`, `15g`, `unlimited`, `default`), по достижении которого произойдет автоматическая ротация. К примеру, если файловый канал называется **`mylog`**, то в процессе ротации появятся версии **`mylog.0`**, **`mylog.1`** и т.д.

Для каналов системы Syslog задается имя средства, указываемое при регистрации сообщений. Это может быть любое стандартное средство, но на практике единственным разумным выбором являются средства `daemon` и `local0`—`local7`.

🕮 Список средств системы Syslog приводился в главе 11.

Остальные разделы в определении канала являются необязательными. Аргумент *важность* может принимать следующие значения (в порядке уменьшения приоритета): `critical`, `error`, `warning`, `notice`, `info` и `debug` (здесь может добавляться номер уровня, например `severity debug 3`). Значение `dynamic` соответствует текущему уровню отладки сервера.

Параметры `print` устанавливают или отменяют вывод различных префиксов сообщений. Система Syslog вставляет перед каждым сообщением метку времени, а также имя узла, но не уровень важности или категорию. Существует также параметр, позволяющий отображать имя исходного файла (модуля), сгенерировавшего сообщение. Параметр `print-time` рекомендуется включать только для файловых каналов, поскольку в Syslog произойдет ненужное дублирование информации.

В табл. 17.14 перечислены четыре канала, которые определены по умолчанию. В большинстве случаев создавать новые каналы не требуется.

**Таблица 17.14. Стандартные каналы журнальной регистрации пакета BIND**

| Имя канала | Назначение |
|---|---|
| `default_syslog` | Направляет сообщения уровня `info` и выше в систему Syslog от имени средства `daemon` |
| `default_debug` | Направляет сообщения в файл **named.run**; уровень важности устанавливается равным `dynamic` |
| `default_stderr` | Направляет сообщения в стандартный канал ошибок демона **named** с уровнем важности `info` |
| `null` | Отбрасывает все сообщения |

## *Категории*

В момент создания программы категории определяются программистом. Они организовывают сообщения по темам или функциональным возможностям, а не по важности. В табл. 17.15 приведен текущий список категорий сообщений.

**Таблица 17.15. Категории сообщений пакета BIND**

| Категория | Что охватывает |
|---|---|
| `client` | Клиентские запросы |
| `config` | Ошибки анализа и обработки конфигурационного файла |
| `database` | Сообщения об операциях с базой данных |
| `default` | Категории, для которых не был явно назначен канал |
| `delegation-only` | Запросы, посланные в NXDOMAIN зонами, выполняющими только делегирование |
| `dispatch` | Диспетчеризация входящих пакетов среди модулей сервера |
| `dnssec` | Сообщения протокола DNSSEC |
| `edns-disabled` | Информация о серверах, вышедших из строя |
| `general` | Неклассифицированные сообщения |
| `lame-servers` | Сообщения о серверах, которые, как предполагается, обслуживают зону, но на самом деле это не так[a] |
| `network` | Сетевые операции |
| `notify` | Сообщения об изменениях зон |
| `queries` | Короткое сообщение для каждого (!) запроса, принимаемого сервером |
| `resolver` | Операции преобразования имен, например рекурсивный поиск, выполняемый от имени клиента |
| `security` | Принятые или непринятые запросы |
| `unmatched` | Запросы, которые демон named не может классифицировать (неправильный класс, нет представления) |
| `update` | Сообщения о динамических обновлениях |
| `update-security` | Одобрение или отклонение запросов на обновление |
| `xfer-in` | Сообщения о зонных передачах, принимаемых сервером |
| `xfer-out` | Сообщения о зонных передачах, отправляемых сервером |

[a] Это может быть как родительская, так и дочерняя зона.

## *Журнал запросов*

Стандартный вид инструкции `logging` таков.

```
logging {
   category default { default_syslog; default_debug; };
};
```

После внесения существенных изменений в систему BIND необходимо просматривать журнальные файлы; возможно, стоит также повышать уровень отладки. После того как демон **named** вернется в стабильное состояние, настройте систему так, чтобы регистрировались только важные сообщения.

Журнал запросов — источник весьма полезной информации. На его основании можно проверить, работают ли директивы `allow`, узнать, кто посылает вам запросы, идентифицировать неправильно сконфигурированных клиентов и т.д.

Для того чтобы включить режим регистрации запросов, назначьте категории `queries` какой-нибудь канал. Регистрация в системе Syslog менее эффективна, чем непосредственно в файле, поэтому при регистрации каждого запроса создайте файловый канал, связанный с локальным диском. Зарезервируйте для журнала достаточно дискового про-

странства и будьте готовы отключить регистрацию, когда накопится достаточный объем данных (команда **rndc querylog** включает и отключает регистрацию запросов).

Иногда отладка представления является довольно сложной задачей, но, к счастью, представление, соответствующее конкретному запросу, можно занести в журнал вместе с запросом.

Ниже перечислены наиболее часто регистрируемые сообщения.

- *Неправильно сконфигурированный сервер* ("Lame server resolving xxx"). Если подобное сообщение поступает от одной из внутренних зон, значит, в конфигурации системы имеется ошибка. Если же в сообщении говорится о какой-то зоне в Интернете, то можно не беспокоиться: это "чужая" ошибка. Сообщения второго вида вполне можно отбрасывать, направляя их в канал `null`.
- *Отклонение запроса* ("...query (cache) xxx denied"). Причиной этого сообщения может быть неправильная конфигурация удаленного сайта, нарушение правил или ситуация, в которой некто делегировал вам зону, но вы ее не конфигурировали.
- *Просроченное время при распознавании: отключение EDNS* (too many timeouts resolving xxx: disabling EDNS). Это сообщение может возникнуть вследствие отказа брандмауэра, не пропускающего пакеты UDP, размер которых превышает 512 байт, и не допускающего фрагментирование. Кроме того, оно может свидетельствовать о проблемах на конкретном узле. Следует убедиться, что проблема не связана с вашим брандмауэром, и рассмотреть возможность переадресации этих сообщений в нулевой канал.
- *Неожиданное распознавание RCODE (SERVFAIL)* (unexpected RCODE (SEVFAIL) resolving xxx). Это сообщение может быть признаком атаки или, что более вероятно, сигналом о том, что кто-то постоянно посылает запросы к неправильно сконфигурированной зоне.
- *Неправильная отсылка* ("Bad referral"). Это сообщение свидетельствует о неправильном взаимодействии серверов имен зоны.
- *Неавторитетен* ("Not authoritative for"). Подчиненный сервер не может получить авторитетные данные о зоне. Возможно, у него хранится неправильный адрес главного сервера или же тот не смог загрузить зону.
- *Зона отклонена* ("Zone rejected"). Демон **named** отказался загружать зонный файл, поскольку тот содержит ошибки.
- *Не найдены записи NS* ("No NS RR for"). В зонном файле после записи SOA не найдены записи NS. Возможно, они отсутствуют либо не начинаются с табуляции или какого-нибудь другого пробельного символа. Во втором случае они не присоединяются к зоне и, следовательно, интерпретируются неправильно.
- *Не задано стандартное значение TTL* ("No default TTL set"). Желательно задавать стандартное значение TTL посредством директивы `$TTL`, располагаемой в начале зонного файла. Ошибка свидетельствует о том, что такая директива отсутствует. В версии BIND 9 наличие этой директивы обязательно.
- *Нет корневых серверов имен для класса* ("No root nameservers for class"). Демону **named** не удается найти корневые серверы имен. Следует проверить файл корневых "подсказок" и подключение сервера к Интернету.
- *Адрес уже используется* ("Address already in use"). Порт, который нужен для работы демона **named**, занят другим процессом, возможно, другой копией демона. Если

демон отсутствует в списке выполняющихся процессов, то, очевидно, он перед этим аварийно завершил свою работу и оставил открытым управляющий сокет утилиты **rndc**, который нужно найти и удалить. Хороший способ устранить проблему — остановить процесс с помощью утилиты **rndc** и перезапустить процесс **named**.

- % **sudo rdnc stop**
- % **sudo /usr/sbin/named ...**
- *Обновление запрещено* ("Denied update from..."). Имела место попытка динамического обновления зоны, которая была отвергнута вследствие установки `allow-update` или `update-policy` для зоны в файле **named.conf**. Это очень распространенное сообщение об ошибке, которая часто вызывается неправильной конфигурацией системы Windows.

### Пример конфигурации журнала запросов в пакете BIND

Приведенный ниже фрагмент взят из файла **named.conf** одного из загруженных серверов имен TLD и представляет собой полную конфигурацию журнала запросов.

```
logging
   channel default_log { # Default channel, to a file
      file "log/named.log" versions 3 size 10m;
      print-time yes;
      print-category yes;
      print-severity yes;
      severity info;
   };
  channel xfer-log { # Zone transfers channel, to a file
      file "log/xfer.log" versions 3 size 10m;
      print-time yes;
      print-category yes;
      print-severity yes;
      severity info;
   };
  channel dnssec-log { # DNSSEC channel, to a file
      file "log/dnssec.log" versions 3 size 10m;
      severity debug 1;
      print-severity yes;
      print-time yes;
   };
   category default { default_log; defaula_debug; }
   category dnssec { dnssec-log; }
   category xfer-in { xfer-log; }

   category xfer-out { xfer-log; }
   category notify { xfer-log; }
};
```

### Уровни отладки в пакете BIND

Уровни отладки демона **named** обозначаются целыми числами от 0 до 100. Чем выше число, тем больше текста содержит выходная информация. Уровень 0 выключает отладку. Уровни 1 и 2 отлично подходят для отладки конфигурационного файла и базы данных. Уровни выше 4 предназначены для разработчиков кода демона.

Отладку можно включить из командной строки, запуская демон **named** с флагом **-d**. Например, команда

```
# sudo named -d2
```

запускает демон **named** на уровне отладки 2. По умолчанию отладочная информация записывается в файл **named.run**, находящийся в каталоге запуска демона. Этот файл растет очень быстро, поэтому во время отладки будьте внимательны.

Отладку можно также включать в процессе работы демона **named** с помощью команды **rndc trace**, которая увеличивает уровень отладки на единицу. Команда **rndc notrace** выключает режим отладки. Кроме того, можно создать канал регистрации сообщений, в определение которого входит строка следующего вида.

```
severity debug 3
```

Эта строка обеспечивает направление в канал всех отладочных сообщений вплоть до уровня 3. Другие директивы в определении канала указывают на то, куда в конечном итоге попадут сообщения. Чем выше уровень важности, тем больше информации регистрируется.

Просматривая журнальные файлы и отладочные сообщения, можно заметить, как часто делаются ошибки в конфигурации DNS. Забыв поставить точку в конце имени, можно спровоцировать угрожающий рост трафика DNS. Помните: точка нужна в конце каждого полностью определенного имени домена.

## Журнальная регистрация в пакетах NSD и Unbound

Журнальная регистрация в пакетах NSD и Unbound проста по сравнению с пакетом BIND. В соответствии с файлом **doc.README**, "пакет NSD не выполняет никакой журнальной регистрации". На самом деле это означает, что пакет NSD не выполняет никакой регистрации трафика DNS и мониторинга, тем не менее, он регистрирует важные события, происходящие с программным обеспечением, в системе Syslog.

По умолчанию журнальные сообщения выводятся в стандартный поток сообщений об ошибках и в систему Syslog с помощью вспомогательного демона. Однако если атрибут `logfile` в инструкции `server` установлен равным **unbound.conf** или **nsd.conf**, то сообщения выводятся в указанный файл.

Объем регистрируемых данных (помимо ошибок, которые учитываются всегда) управляется уровнем детализации, который можно задать в файле конфигурации. При работе с демоном **unbound** уровень детализации можно задать в командной строке. Этот параметр изменяется от 0 до 5; по умолчанию он равен 0 для демона **nsd** и 1 — для демона **unbound**.

Иногда трудно понять, какой уровень детализации следует выбрать для демона **nsd**. Они классифицируются примерно так.

- Уровень 3 — "ошибка"
- Уровень 4 — "предупреждение"
- Уровень 5 — "замечание"
- Уровень 6 — "информация"

Описание уровней детализации для демона **unbound** приведено на справочной странице файла конфигурации.

- Уровень 1 — не регистрируется никакая информация, кроме ошибок
- Уровень 2 — операционная информация

- Уровень 3 — детальная операционная информация
- Уровень 4 — информация о запросах
- Уровень 5 — информация об алгоритмах
- Уровень 6 — информация о кеше и идентификации клиента

Для того чтобы включить режим отладки, можно вызвать демона **nsd** с параметром **-d**. В этом режиме демон **nsd** получает высокий приоритет, не разветвляется и не прекращает работу. Это эквивалентно установке параметра `debug-mode: yes` в разделе `server` в файле **nds.conf**. Если перекомпилировать демона **nsd** с параметром DEBUG, соответствующим конкретному уровню отладки, то можно получить еще больше отладочной информации, но она по большей части требуется только разработчикам.

Демон **unbound** тоже имеет флаг **-d**, чтобы включить режим отладки. Уровнем детализации можно управлять с помощью параметра **-v**; чтобы получить еще больше информации, попробуйте комбинацию **-v -v**. Регистрация отладочной информации задается отдельно от детализации регистрационной информации в файле конфигурации. Можно установить дополнительные уровни регистрации, чтобы облегчить процесс проверки подписи DNSSEC.

## Управляющие программы сервера имен

Все три сервера имен поставляются с управляющими программами: программа **nsdc** управляет демоном **nsd**, программа **rndc** — демоном **named**, а программа **unbound-control** — демоном **unbound**. Программа **nsdc** работает только на локальном компьютере, а программы **rndc** и **unbound-control** могут работать в Интернете, если их соответствующим образом настроить.

Программа **rndc** использует сетевой сокет для взаимодействия с демоном **named** и систему аутентификации TSIG для безопасности; программа **unbound-control** использует протокол SSL/TLS, а программа **nsdc** — сигналы.

### *Использование программы rndc в пакете BIND*

Опции программы **rndc** перечислены в табл. 17.16. Если напечатать имя программы **rndc** без аргументов, то она выведет на экран список доступных команд и их краткое описание. В предыдущих версиях программы **rndc** использовались сигналы, как и в программе **nsdc**. Однако количество доступных команд "перевалило" за 25, поэтому разработчики пакета BIND отказались от использования сигналов. Команды, приводящие к созданию файлов, размещают их в каталоге, заданном в файле **named.conf** как начальный каталог демона **named**.

**Таблица 17.16. Команды программы rndc** [a]

| Команда | Назначение |
|---|---|
| **dumpdb** | Записывает образ базы данных DNS в файл **named_dump.db** |
| **flush** [представление] | Очищает все кеши или кеши, заданные представлением |
| **flushname** имя [представление] | Удаляет имя из кеша сервера |
| **freeze** зона [класс [представление]] | Приостанавливает обновления динамической зоны |
| **thaw** зона [класс [представление]] | Возобновляет обновления динамической зоны |
| **halt** | Останавливает демон **named** без обработки отложенных обновлений |

*Окончание табл. 17.16*

| Команда | Назначение |
|---|---|
| **querylog** | Переключает режим трассировки входящих запросов |
| **notify** зона [класс [представление]] | Повторно посылает зоне уведомления |
| **notrace** | Отключает режим отладки |
| **reconfig** | Повторно загружает конфигурационный файл и все новые зоны |
| **recursing** | Записывает текущие рекурсивные запросы в файл named.recursing |
| **refresh** зона [класс [представление]] | Поддерживает расписания для зоны |
| **reload** | Повторно загружает файл **named.conf** и зонные файлы |
| **reload** зона [класс [представление]] | Повторно загружает только указанную зону или представление |
| **restart** [6] | Повторно запускает сервер |
| **retransfer** зона [класс [представление]] | Повторно копирует данные для зоны из главного сервера |
| **stats** | Записывает статистическую информацию в файл **named.stats** |
| **status** | Отображает на экране текущий статус выполнения демона **named** |
| **stop** | Сохраняет отложенные обновления и останавливает демон **named** |
| **trace** | Увеличивает уровень отладки на единицу |
| **trace** приращение | Увеличивает уровень отладки на приращение |
| **validation** новое_состояние | Включает/отключает проверку DNS на лету |

[a] Аргумент *класс* означает то же, что и в случае записей о ресурсах; для Интернета он обычно равен IN.

[б] Эта команда еще не реализована в пакете BIND 9 (9.7.0), но разработчики обещают скоро ее реализовать (должно быть, это трудно).

Команда **rndc reload** заставляет демон **named** заново прочитать конфигурационный файл и повторно загрузить зонные файлы. Команду **reload** *зона* удобно применять на интенсивно эксплуатируемом сервере, когда изменению подвергается только одна зона, а остальные трогать не нужно. Кроме того, можно задать параметры *класс* и *представление*, чтобы загрузить только указанное представление зонных данных.

Обратите внимание на то, что команды **rndc reload** недостаточно, чтобы добавить совершенно новую зону; для этого требуется, чтобы демон **named** прочитал файл **named.conf** и новый зонный файл. Для новых зон следует использовать команду **rdnc reconfig**, которая заново прочитывает файл конфигурации и загружает любую новую зону, не трогая существующие.

Команда **rdns freeze** останавливает динамические обновления и согласовывает журнал динамических обновлений с зонными файлами. После замораживания зоны зонные данные можно редактировать вручную. Поскольку зона заморожена, динамическое обновление происходить не будет. Завершив редактирование, выполните команду **rdnc thaw** *зона*, чтобы принимать динамические обновления снова.

Команда **rndc dumpdb** заставляет демон **named** записать образ своей базы данных в файл **named_dump.db**. Этот файл очень большой и содержит не только локальные данные, но и данные, накопленные в кеше сервера имен.

Версии демона **named** и программы **rndc** должны совпадать, иначе будет выдано сообщение о несовпадении версий протокола. Обычно они устанавливаются на один и тот

же компьютер, и расхождение версий может вызвать проблемы, если демон **named** попытается управлять другим компьютером.

### Использование программы nsdc в пакете NSD

Управляющая программа **nsdc** пакета NSD представляет собой оболочку, использующую сигналы для управления поведением демона **nsd** и утилиты **zonec**, которая прилагается к предкомпилятору зоны. Поскольку программа **nsdc** должна выполняться на той же машине, что и демон **nsd**, ключи не нужны.

Как следует из табл. 17.17, программа **nsdc** имеет меньше команд, чем программа **rndc**. При выполнении программа **nsdc** считывает конфигурационный файл **nsd.config** и использует утилиту **nsd-checkconf** для проверки синтаксических ошибок.

**Таблица 17.17. Команды программы nsdc**

| Команда | Функция |
|---|---|
| **start** | Запускает сервер nsd |
| **stop** | Останавливает сервер nsd (посылает сигнал SIGTERM) |
| **reload** | Повторно загружает базу данных скомпилированной зоны |
| **rebuild** | Повторно создает зонную базу данных с помощью утилиты zonec |
| **restart** | Повторно запускает сервер nsd, т.е. останавливает его, а затем запускает |
| **running** | Проверяет, запущен ли сервер nsd; если все в порядке, никакой информации не выводит |
| **update** | Пытается обновить все подчиненные зоны |
| **notify** | Посылает уведомления всем подчиненным серверам |
| **patch** | Объединяет изменения, связанные с передачей зоны (в базе данных), в зонном файле (текстовом) |

### Использование программы unbound-control

Программа **unbound-control** посылает сообщения серверу имен **unbound** посредством протокола безопасности транспортного уровня TLS (бывшего SSL), который использует ключ и сертификат для каждого конечного пользователя. Эти ключи конфигурируются в разделе `remote-control` файла конфигурации **unbound.conf**. Для того чтобы управлять поведением сервера, можно использовать более 20 команд. Вместо перечисления всех возможных команд, мы отправляем читателей на справочную страницу программы **unbound-control**, на которой она подробно описана.

Среди этих 20 команд есть обычные команды для запуска, остановки и повторного чтения, но некоторые опции осуществляют очень тонкое управление кешем и локальными зонными данными, которые должны конфигурироваться. Такие опции, как переадресовка (forwarding), могут конфигурироваться или модифицироваться на лету.

## Сбор статистических данных

Каждый сервер имен ведет статистику запросов с той или иной степенью детализации. Некоторые опции можно конфигурировать, например можно указать, какие данные регистрировать, где их записывать, как часто обновлять и т.д. Новый статистический канал в пакете BIND является более гибким механизмом. Сервер NSD позиционируется как надежный и быстрый сервер, статистические данные для которого должна собирать другая программа, пока сервер NSD выполняет свою основную работу. Сервер **unbound** занимает промежуточную позицию.

Серверы BIND и **unbound** могут посылать свои статистические данные другим программам для презентации и визуализации. Сервер BIND использует инструкцию `statistics-channels` и язык XML. Сервер **unbound** применяет надстройки для каталога **contrib**, обеспечивающие связь с системами мониторинга Munin или Cacti.

Демон **named** накапливает итоговую информацию, которую он может сохранять в файле **named.stats** в рабочем каталоге демона **named** в ожидании команды от программы **rndc**.

```
$ sudo rndc stats
```

Рассмотрим небольшой фрагмент вывода с сервера в домене `vix.com`. Большая часть информации была отброшена; в файле показаны 20 групп данных, мы приводим лишь две из них.

```
+++ Statistics Dump +++ (1248150900)
++ Incoming Queries ++
2650862 A
9105 NS
404378 SOA
85744 PTR
246258 MX
3208092 TXT
...
++ Name Server Statistics ++
10028960 IPv4 requests received
388015 IPv6 requests received
5890639 requests with EDNS(0) received
92403 TCP requests received
4363730 queries resulted in successful answer
766435 queries resulted in nxrrset
599672 queries resulted in NXDOMAIN
...
```

Эти статистические данные демонстрируют соотношение успешных и безуспешных проверок и классифицируют разные виды ошибок. Данный сервер получил 10 млн запросов, большинство из которых имели типы AAAA (адрес по протоколу IPv6) и TXT (возможно, запросы на записи SF или DKIM). Записи свидетельствуют о довольно активной деятельности злоумышленников на этом сервере (например, о запросах на неавторизованную передачу зоны). Вероятно, необычно большое количество запросов типа AAAA и TXT является частью этой активности — чаще всего запрашивались записи A.

Если демон **named** компилируется с библиотекой XML, то инструкция `statistics-channels` в файле **named.conf** устанавливает статистический работающий в реальном времени канал, который пользователь может просматривать с помощью веб-браузера.

## Отладка с помощью команды dig

Команды **nslookup**, **dig**, **host** и **drill** позволяют в режиме командной строки посылать запросы базе данных DNS. Первые три команды входят в дистрибутив BIND, а команда **drill** — в дистрибутив Unbound/**ldns**. Команды **nslookup** и **host** простые и выдают ясные результаты, но для того чтобы выяснить все детали, необходимы команды **dig** и **drill**. Команда **drill** лучше прослеживает цепочки подписей по протоколу DNSSEC. Имя команды **drill** обыгрывает имя **dig** (domain information groper — сред-

ство сбора информации о доменах)[38], подсказывая, что благодаря этой команде можно получить еще больше информации от системы DNS, чем с помощью команды **dig**.

По умолчанию команды **dig** и **drill** посылают запросы серверам имен, сконфигурированным в файле **/etc/resolv.conf**. Аргумент *@сервер_имен* адресует команду конкретному серверу имен. Способность посылать запросы конкретному серверу позволяет гарантировать, что любые изменения, вносимые в зону, будут распространены среди подчиненных серверов и во внешнем мире. Это свойство особенно полезно, если вы используете представления (расщепляете DNS) и должны проверять, правильно ли они сконфигурированы.

Если указать тип записи, то команды **dig** и **drill** будут выполнять запрос только к этим записям. Псевдотип **ANY** действует немного неожиданно: вместо того чтобы вернуть все данные, ассоциированные с указанным именем, он возвращает все *кешированные* данные, связанные с ним. Итак, чтобы получить все записи, необходимо задать команду **dig** *домен* **NS**, за которой следует выражение **dig @nsl.***домен*.*домен* **ANY**. (Авторитетные данные в этом контексте рассматриваются как кешированные.)

Команда **dig** имеет более 50 опций, а команда **drill** — около половины этого количества. Получив флаг **-h**, эти команды выдают список всех своих опций. (Для того чтобы упорядочить вывод, можно использовать параметр **less**.)Для обеих команд опция **-x** изменяет порядок следования байтов в IP-адресе на противоположный и выполняет обратный запрос. Флаги **+trace** команды **dig** и **-T** команды **drill** показывают итеративные шаги в процессе распознавания, начиная от корня и вниз по дереву иерархии.

Мы не приводим примеры вывода с помощью команд **dig** и **drill**, поскольку мы использовали их во всей главе при описании разных аспектов системы DNS.

Если ответ является авторитетным (т.е. приходит непосредственно от главного или подчиненного сервера данной зоны), то команды **dig** и **drill** во флагах вывода используют символы `aa`. Код `ad` означает, что ответ является аутентичным в смысле протокола DNSSEC. Тестируя новую конфигурацию, следует убедиться, что вам доступны данные как локальных, так и удаленных узлов.

## Некорректное делегирование

Подавая заявку на получение доменного имени, вы просите, чтобы основному серверу имен и администратору DNS была выделена (делегирована) часть дерева имен. Если не поддерживать работу домена или изменить адреса серверов имен, не обновив связующие записи родительского домена, будет иметь место так называемое *некорректное делегирование* (lame delegation).

Последствия некорректного делегирования могут оказаться весьма негативными. Если хотя бы один из ваших серверов окажется некорректным, то вся ваша система DNS снизит эффективность работы. Если все серверы имен являются некорректными, то ни один из них не будет доступен. Все запросы будут начинаться с корня, пока ответы будут кешироваться, поэтому некорректные серверы и неисправное программное обеспечение, которое не делает негативного кеширования ошибок SERVFAL, увеличивают нагрузку на каждый узел, находящийся на пути от корня к некорректному домену.

Есть два способа выявить некорректное делегирование — просматривая журнальные файлы и используя инструмент под названием **doc**, которое представляет собой сокращение от словосочетания "domain obscenity control" (проверка корректности домена).

[38] Dig — копать, а drill — бурить, т.е. проникать глубже. — *Примеч. ред.*

Несколько примеров работы утилиты **doc** мы рассмотрим в следующем разделе, а пока приведем несколько записей из журнальных файлов.

На многих сайтах в качестве канала для регистрации указан каталог **/dev/null**, чтобы не беспокоиться о некорректном делегировании. Это допустимо, если ваш домен находится в полном порядке и не является ни источником, ни жертвой некорректного делегирования. Всего лишь один некорректный сервер снижает эффективность работы системы DNS; если все серверы некорректные, то домен практически "взлетает на воздух".

Рассмотрим пример регистрационных записей. Мы сократили вывод, чтобы не загромождать изложение; флаг **+short** позволяет сделать вывод команды **dig** еще более лаконичным.

```
Jul 19 14:37:50 nubark named[757]: lame server resolving 'w3w3.com'(in
  'w3w3.com'?): 216.117.131.52#53
```

Послав с помощью команды **dig** запрос о серверах имен для домена w3w3.com одному из серверов gTLD-домена .com, мы получим следующую информацию.

```
$ dig @e.gtld-servers.net w3w3.com ns
;; ANSWER SECTION:
w3w3.com    172800  IN  NS  ns0.nameservices.net.
w3w3.com    172800  IN  NS  ns1.nameservices.net.
```

Если же теперь опросить каждый из этих серверов по очереди, задав им этот же вопрос, то мы получим ответ от сервера ns0 и не получим ответа от сервера ns1.

```
$ dig @ns0.nameservices.net w3w3.com ns
;; ANSWER SECTION:
w3w3.com    14400  IN  NS  ns0.nameservices.net.
w3w3.com    14400  IN  NS  ns1.nameservices.net.
$ dig @ns1.nameservices.net w3w3.com ns
;; QUESTION SECTION:
;; w3w3.com.    IN  IN

;; AUTHORITY RECORDS:
com.                  244362 IN  NS  M.GTLD-SERVERS.NET.
com.                  244362 IN  NS  I.GTLD-SERVERS.NET.
com.                  244362 IN  NS  E.GTLD-SERVERS.NET.
```

Сервер имен ns1.nameservices.net делегировал ответственность за домен w3w3.com серверам домена .com, но они эту ответственность не приняли. Эта конфигурация является неправильной, и возникло некорректное делегирование. Клиенты, пытающиеся попасть в домен w3w3.com, будут обслуживаться медленно. Если домен w3w3.com оплачивает услуги DNS домену nameservices.net, то он заслуживает компенсации!

Иногда, посылая запрос с помощью команды **dig** на авторитетный сервер в поисках некорректности, можно не получить никакой информации. В этом случае следует попытаться повторить запрос с флагом **+norecurse**, чтобы можно было увидеть точно, что знает искомый сервер.

## Инструменты для проверки корректности системы DNS

Существует несколько инструментов, позволяющих проверить некоторые аспекты системы DNS. В пакете BIND 9 поставляются утилиты **named-checkconf** и **named-checkzone**; они проверяют выполнение основных синтаксических правил (не семантику) в файле **named.conf** и зонных файлах. Пакеты NSD и Unbound содержат аналогичные инструменты под названием **nsd-checkconf** и **unbound-checkconf**.

Оригинальным инструментом для проверки системы DNS является программа **nslint**, написанная Крейгом Лересом (Craig Leres), когда он работал в компании Lawarence Berkeley Labs. Утилита **doc** (domain obscenity control) — программа для проверки корректности домена — проверяет делегирование и находит несоответствия и ошибки в развернутой системе DNS. Ниже она обсуждается более подробно. Инструмент **lamer** (расположенный на том же веб-сайте, что и утилита **doc**) просматривает журнальные файлы и посылает администраторам DNS по электронной почте сообщения о некорректных сайтах, уведомляя их о том, что эти сайты порождают некорректное делегирование, и описывая, как исправить проблему. Утилита **DDT**, написанная Жоржем Фразао (Jorze Frazao) и Артуром Ромао (Artur Romao), выполняет отладку кешированных данных.

Утилита **dnswalk** проходит по дереву делегирования и идентифицирует несоответствия между родительским и дочерним узлами или между прямыми и обратными записями. Она также находит отсутствующие точки, избыточные связующие записи и т.д. Это средство для поддержания общего порядка в системе DNS. Для успешной работы утилита **dnswalk** должны иметь возможность выполнять передачу зоны.

Некоторые средства отладки и управления протоколом DNSSEC перечислены в разделе "Отладка протокола DNSSEC".

Утилита **doc** реализует основной сценарий, написанный на языке C. В настоящее время ее сопровождение осуществляет Бред Ноулз (Brad Knowls). Эту утилиту можно загрузить с его сайта `shub-internet.org/brad/dns` (обратите внимание на то, что следует писать "shub", а не "shrub"). Если вы планируете указать утилиту **doc** в своем разделе PATH или запустить ее с помощью планировщика задач **cron**, то должны отредактировать сценарий и установить переменную `auxd` так, чтобы она указывала на каталог инсталляции.

Утилита **doc** проверяет делегирование, выполняя повторяющиеся вызовы команды **dig**. Она сообщает о несоответствиях, ошибках и других проблемах, связанных с конкретным именем домена. На экран выводится информация о всех выявленных проблемах. Кроме того, эта утилита создает подробный журнальный файл в текущем каталоге.

Для своей работы утилита **doc** использует локальный сервер имен. Если домен использует инструкцию `view` сервера BIND и содержит частные адреса RFC1918 в своем внутреннем представлении, то применение утилиты **doc** к внутреннему представлению сбивает ее с толку и она сообщает о ложных ошибках. Если вы используете представления, то запустите утилиту **doc** извне домена, чтобы она видела то, что видят внешние пользователи.

Ниже приведен отчет утилиты **doc** о некорректном домене `w3w3.com`, который был рассмотрен выше.

```
$ doc w3w3.com
Doc-2.2.3: doc w3w3.com
Doc-2.2.3: Starting test of w3w3.com.  parent is com.
Doc-2.2.3: Test date - Tue Jul 21 21:15:11 MDT 2009
Summary.
   ERRORS found for w3w3.com. (count: 1)
   WARNINGS issued for w3w3.com. (count: 1)
Done testing w3w3.com.  Tue Jul 21 21:15:21 MDT 2009
```

Утилита **doc** выводит в журнальный файл (в данном случае файл **log.w3w3.com**) подробности тестирования и сообщение об ошибках (в сумме более 600 строк).

```
ERROR: no SOA record for w3w3.com. from ns1.nameservices.net.
```

Утилита не помечает домен w3w3.com как "некорректное делегирование", а идентифицирует проблему, которая в связи с этим возникает. Начиная с записей NS в родительской зоне, утилита проверяет, является ли сервер ns1.nameservices.net авторитетным по отношению к домену w3w3.com; в данном случае это не так.

Если вы управляете доменом, который содержит поддомены (или вы не доверяете менеджерам вашего родительского домена), постарайтесь выполнять утилиту **doc** из планировщика заданий **cron** каждую неделю, чтобы проверить, что все делегирования, связанные с вашим доменом, являются корректными.

## Производительность системы

Производительность пакета BIND 9 на многопроцессорной архитектуре оказалась не такой высокой, как надеялись его разработчики, но некоторые корневые серверы используют пакет BIND 9 и успешно обрабатывают десятки тысяч запросов в секунду. Для большинства сайтов этого, вероятно, более чем достаточно. Однако производительность пакетов NSD и Unbound еще выше, особенно в зонах, подписанных по протоколу DNSSEC.

Мы уже несколько раз об этом говорили, но скажем еще раз: присваивайте вашим параметрам TTL разумные значения — недели или дни, а не минуты и секунды. Использование коротких периодов TTL отрицательно сказывается как на вашей работе (поскольку вы постоянно перезаписываете одни и те же записи), так и на работе ваших веб-клиентов (поскольку они постоянно извлекают эти записи). Кроме того, это позволяет потенциальным злоумышленникам искажать содержимое вашего кеша. Некоторые серверы имен преднамеренно игнорируют параметры TTL, которые им не нравятся (т.е. если эти значения неразумно малы или чрезмерно велики).

Подкачка памяти снижает производительность сервера нелинейно, поэтому не скупитесь на память ваших узлов, которые запускают серверы имен. Для того чтобы стабилизировать работу рекурсивных серверов, вам понадобится около недели на копирование памяти (см. раздел 17.6).

Для того чтобы оценить объем памяти, который может понадобиться демону **nsd** для обработки ваших авторитетных данных, заполните веб-форму на веб-странице nlsnetlabs.net/nsd/nsd-memsize.html. Отлично!

Используйте переадресующие серверы (см. раздел 17.9).

Сценарии **init**, запускающие демон **named** на многих системах, создают дополнительные точки входа (например, **reload**), предназначенные для системного администратора. Однако проще и надежнее с точки зрения работы на разных платформах использовать утилиту **rdnc**.

🕮 Более подробно утилита inetd описана в разделе 32.1.

Не используйте программы **inetd** или **xinetd** для управления сервером имен; они перезагружают сервер каждый раз, когда им это требуется, тем самым резко замедляя время ответа и не позволяя создавать хоть сколько-нибудь полезный кеш.

# 17.16. Специфика различных дистрибутивов

В этом разделе описывается программное обеспечение серверов имен для разных платформ. В каждый дистрибутивный пакет серверов имен входит пакет BIND, хотя при инсталлировании операционной системы, а также при отдельной установке сервера имен есть возможность указать, какой именно пакет вы хотите установить.

Операционные системы семейства Linux более оперативно реагируют на появление новых версий пакета BIND, чем ее аналоги из семейства UNIX. Тем не менее служба имен играет чрезвычайно важную роль. По этой причине вполне естественно желать, чтобы на компьютере немедленно устанавливались новейшие системы, а не дожидаться, пока эти пакеты станут общедоступными.

Системы Ubuntu и Red Hat содержат полный комплект NSD/Unbound в виде пакетов; система SUSE хранит в своих официальных репозиториях только пакет Unbound. В настоящее время все эти пакеты слегка устарели, поэтому мы рекомендуем загружать их новейшие версии с сайта `nlnetlabs.nl` и собирать их самостоятельно.

В этом разделе рассмотрим указатели на файлы конфигурации, на которых основана работа любой версии пакета BIND любого поставщика. Кроме того, мы покажем, как объединить пакет BIND с другими источниками административных данных, такими как однородные файлы и файлы службы NIS. Более подробное обсуждение последней темы изложено в главе 19.

🕮 Более подробная информация о сценариях загрузки системы приведена в разделе 3.5.

Мы также рассмотрим указатели на сценарии, которые должны выполняться при загрузке серверов имен. Если сервер имен выходит из строя, то выходит из строя абсолютно все — сеть, электронная почта, веб-сайт. Некоторые организации используют сценарий сохранения работоспособного состояния. К таким сценариям относится **nanny** из дистрибутивного пакета BIND.

## Специфика системы Linux

Пакеты BIND для системы Linux инсталлируют сценарий для пакетов **named**, который запускается с помощью команд **init: /etc/inut.d/bind9** для системы Ubuintu и **/etc/init.d/named** для систем RHEL и SUSE.

Пакеты **named** в системе Linux инсталлируют все программы в своих традиционных каталогах. Детали показаны в табл. 17.18. Система Red Hat имеет дополнительные программы, которые взаимодействуют с инструментом Network Manager с помощью флага **-D**, который задается в командной строке демона **named**. Подробности можно найти в файле doc/README-DBUS в пакете Red Hat BIND.

**Таблица 17.18. Файлы пакета BIND в Linux**

| Файл | Каталог | Описание |
|---|---|---|
| **resolv.conf** | **/etc** | Библиотечный файл конфигурации распознавателя |
| **named, iwres** | **/usr/sbin** | Демон сервера имен |
| **lwresd** | **/usr/sbin** | Облегченный распознаватель |
| **named.conf** | **/etc** | Конфигурационный файл сервера named (RHEL и SUSE) |
| **named.conf** | **/etc/bind** | Конфигурационный файл сервера имен (Ubuntu) |
| **named-checkconf** | **/usr/sbin** | Проверяет синтаксис файла конфигурации |
| **namedGetFrowarders** | **/usr/sbin** | Предназначен для инструмента Network Manager (только в Red Hat) |
| **namedSetFrowarders** | **/usr/sbin** | Предназначен для инструмента Network Manager (только в Red Hat) |
| **nsswitch.conf** | **/etc** | Файл переключения службы |

Система Linux использует файл переключения **/etc/nsswitch.conf** для того, чтобы указать, как выполняется отображение IP-адреса в имя компьютера и в каком порядке

следует использовать систему DNS — первой, последней или вообще не использовать. Если файла переключения нет, то по умолчанию устанавливается следующее поведение.

```
hosts: dns [!UNAVAIL=return] files
```

Раздел `!UNAVAIL` означает, что если служба DNS доступна, но имя в ней не найдено, следует прекратить попытки поиска и не продолжать просмотр следующей записи (в данном случае файла **/etc/hosts**). Если сервер имен не был запущен (это бывает во время загрузки системы), то процесс поиска *рекомендуется* согласовывать с файлом **hosts**.

Все рассмотренные нами дистрибутивные пакеты содержат в файле **nsswitch.conf** следующую запись, определяющую поведение по умолчанию.

```
hosts: files dns
```

Не существует "наилучшего" способа конфигурации поиска — все зависит от того, как управляется ваш сайт. В общем, мы предпочитаем хранить как можно больше информации об узле в системе DNS, а не в файлах системы NIS или в обычных файлах, но мы также пытаемся сохранить возможность вернуться при необходимости назад к статическим файлам **hosts** в процессе загрузки системы.

Система Ubuntu является наиболее современной версией системы Linux, поскольку ее дистрибутивные пакеты появились лишь совсем недавно. Ее программы и файлы имеют корневого владельца и группового владельца "bind", которые выдают разрешения на получение доступа, когда демон **named** вызывается как пользователь сервера BIND, а не как корень.

Несколько полезных файлов находится в каталоге **/etc/bind**. Помимо них, в пакет входят файлы **named.conf** и зонные файлы, содержащие корневые подсказки, информацию о локальном узле, адресах широковещательной рассылки и пространстве частных адресов. Поставляемый файл **named.conf** содержит файлы **named.conf.options** и **named.conf.local**. Он задает каталог пакета BIND по умолчанию — **/var/cache/bind**; при поставке этот каталог уже существует, но остается пустым.

Размещение конфигурации в каталоге **/etc**, а зонной информации — в каталоге **/var** объясняется тем, что если ваш сервер является подчиненным по отношению к другим сайтам, то вы не можете управлять размерами своих зонных файлов, которые записывает демон **named**. Для того чтобы предотвратить потенциальное заполнение корневой части, следует сохранить файлы в каталоге **/var**. Зоны, для которых вы являетесь главным сервером, могут находиться вместе с файлами конфигурации (и задаваться абсолютными путями в файле **named.cond**) либо в каталоге **/var/cache/bind**.

Если вы хотите запустить только кеширующий сервер, то модифицировать файл **named.conf** не следует. Все зоны, по отношению к которым вы являетесь авторитетным, желательно добавлять в файл **named.conf.local**.

Стандартные файлы, предоставляемые системой Ubuntu, используют новые функциональные возможности пакета BIND, помогающие серверам правильно работать в системе DNS. Например, они конфигурируют домены `.com` и `.net` как зоны, выполняющие только делегирование, чтобы опечатки пользователей не генерировали доход от рекламы для компании VerySign благодаря их инструменту Site Finder. Если вы не используете пространство частных адресов (RFC1918) изначально, то пустые зонные файлы RFC1918 предотвращают утечку этих адресов из локальной сети. Вперед, Ubuntu!

Каталог **/usr/share/doc/bind9** содержит несколько полезных ссылок. Проверьте файл **README.Debian** (даже в системе Ubuntu), чтобы понять стратегию конфигурирования пакета BIND.

Инсталляция системы SUSE сопровождается сообщениями о выполняемых действиях и генерирует хорошо документированную инсталляцию сервера имен. По умолчанию демон **named** выполняется в виртуальной среде **chroot** в каталоге **/var/lib/named** как пользователь и группа "named". Инсталлятор создает виртуальный каталог **chroot** и заполняет его всеми файлами, необходимыми для выполнения демона **named**, даже такими как сокет UNIX-домена для системы **syslog**. Дополнительные файлы конфигурации (не совпадающие с файлом **named.conf**) и зонные файлы в каталоге **/etc/named.d** копируются в виртуальный каталог при запуске демона **named**. Если вы не хотите выполнять демон **named** в виртуальной среде, модифицируйте строку

```
NAMED_RUN_CHROOTED="yes"
```

в каталоге **/etc/sysconfig/named**. Это все, что следует изменить; сценарии запуска в каталоге **/etc/init.d** используют эту информацию и могут запускать демон **named** в любом режиме.

Система SUSE предоставляет стандартный файл **/etc/named.conf**, содержащий описание множества полезных опций. Файл **/etc/named.conf** является нечитабельным, в отличие от других систем. Стандартный файл импортирует файл с именем **named.conf.include**, который затем импортирует файл **rdnc-access.conf** из каталога **/etc/named**, который можно прочитать. Не совсем ясно, почему система SUSE таким образом беспокоится о безопасности. Программа **rdnc** заранее конфигурируется так, чтобы принимать управляющие команды только от локального узла.

Для того чтобы запустить только кеширующий сервер, файл **named.conf** системы SUSE можно использовать как есть. Если же вы хотите обслуживать свои зоны, то поместите зонные файлы в каталог **/etc/named.d**, а список зонных имен — в файл **/etc/named.conf.include**.

Документация к пакету ISC BIND хранится в каталоге **/usr/share/doc/packages/bind9**.

При инсталляции системы RHEL пакет BIND помещает бинарные файлы в каталог **/usr/sbin**, справочные страницы — в каталог **/usr/share/man**, добавляет пользователя и группу с именем "named" и создает каталоги для зонных файлов. Пользователь "named" имеет доступ к файлам данных с помощью разрешений группы.

Если не внести изменений в файл **/etc/sysconfig/named**, то файл **named.conf** записывается в каталог **/etc** (как указал Пол Викси (Paul Vixie)), а зонные файлы — в каталог **/var/named**. Стандартные файлы конфигурации не поставляются, но в пакете **bindconf** они есть.

## Специфика системы Solaris

Система Solaris 10 поставляется вместе с пакетом BIND 9.3.4-P1, выпущенным в конце 2007 года; он уже устарел, потому следует подумать о его модификации. Система OpenSolaris является более современной и поставляется вместе с пакетом BIND 9.6.1-P1 (2009). Система Solaris всегда называла свои сетевые программы как **in.***имя_программы*, и демон **named** — не исключение, поэтому его можно не сразу найти, если забыть, что на самом деле он называется **in.named**, а не **named**. К счастью, это уже не так, и теперь система Solaris называет демон просто **named**, а имя **in.named** осталось просто как ссылка.

Аналогично системе Linux система Solaris использует файл заказа на обслуживание **/etc/nsswitch.conf**, с помощью которого она указывает, как должны взаимодействовать серверы BIND, NIS, NIS+ (только в системе Solaris 10) и файл **/est/hosts**. Замена строки `hosts` в этом файле на строку

```
hosts:  files dns
```

заставляет распознаватель имен сначала проверять файл **/etc/hosts** и только потом систему DNS. Короткая инструкция `NOTFOUND=return` может модифицировать любую запись. Размещение имен важных серверов и маршрутизаторов в файле **/etc/hosts** облегчает решение причинно-следственной проблемы, которая иногда возникает на этапе загрузки системы, пока не станет доступной служба имен. Служба имен в системе Solaris начинается со службы SMF `svc:/network/dns/server::default`. Опции SMF задают аргументы командной строки.

Имена файлов пакета BIND и их расположение в системе Solaris приведены в табл. 17.19.

**Таблица 17.19. Файлы пакета BIND 9 в системе Solaris**

| Файл | Каталог | Описание |
|---|---|---|
| **resolv.conf** | **/etc** | Библиотечный файл конфигурации распознавателя |
| **named** | **/usr/sbin** | Демон сервера имен |
| **named-checkconf** | **/usr/sbin** | Проверяет синтаксис файла конфигурации |
| **named-checkzone** | **/usr/sbin** | Проверяет синтаксис зонного файла |
| **named-conf** | **/etc** | Файл конфигурации для сервера имен |
| **nsswitch.conf** | **/etc** | Файл переключения службы |

## Специфика системы HP-UX

Система HP-UX содержит устаревший пакет BIND 9.3.2, выпущенный в конце 2005 года. Лучше обновите! Система HP-UX поставляется со стандартными файлами **nsswitch.conf** для разных комбинаций баз данных и служб. Один из них содержит список умолчаний для системы HP. Скопируйте файл, который кажется вам подходящим, в файл **/etc/nsswitch.conf**.

Рекомендации по умолчанию для системы HP:

```
hosts:    dns [NOTFOUND=return] nis [NOFOUND=return] files
```

Однако мы рекомендуем такую настройку, которая предотвращает проблемы при загрузке системы.

```
hosts:    files [NOTFOUND=continue] dns
```

Важно иметь возможность конфигурировать сеть в последовательности загрузки без поиска имен в службах NIS или DNS, поскольку эти службы не могут работать, пока система не загрузится.

Дистрибутивный пакет системы HP-UX содержит хорошо документированные стандартные файлы практически на все случаи жизни. Они находятся в каталоге **/usr /newconfig**, но, увы, среди них нет ни одного, который относился бы к службе имен. Однако существует набор устаревших команд, связанных со службой имен, — **hosts_ to_named**, которая преобразует файл **hosts** в зонный файл, и **sig_named**, которая посылает управляющие сигналы выполняемому процессу **named**. В настоящее время файл

`/etc/hosts` содержит как имя локального узла, так и имя самого узла, поэтому программа для преобразования не нужна. Начиная с версии BIND 9, для управления процессом **named** должна использоваться программа **rdnc**, а не сигналы.

Имена файлов пакета BIND и их расположение в системе HP-UX приведены в табл. 17.20.

**Таблица 17.20. Файлы пакета BIND в системе HP-UX**

| Файл | Каталог | Описание |
|---|---|---|
| resolv.conf | /etc | Библиотечный файл конфигурации распознавателя |
| named | /usr/sbin | Демон сервера имен |
| lwresd | /usr/sbin | Облегченный распознаватель |
| named-checkconf | /usr/sbin | Проверяет синтаксис файла конфигурации |
| named-checkzone | /usr/sbin | Проверяет синтаксис зонного файла |
| named-conf | /etc/namedb | Файл конфигурации для сервера имен |
| nsswitch.conf | /etc | Файл переключения службы |

## Специфика системы AIX


Система AIX содержит пакеты BIND 8 и 9, которым соответствуют бинарные файлы **named8** и **named9** соответственно. При поставке файл **named** представляет собой ссылку на файл **named8**, которая больше не поддерживается консорциумом ISC. Они поставляют версии 8.3.3+ и 9..2.1, выпущенные еще в январе 2004 года. Очевидно, что причиной отсутствия обновлений является лень! Система AIX не содержит команд **named-checkconf** и **named-checkzones**, возможно, потому что они входят в пакет BIND начиная с 2004 года. Сценарий запуска службы имен в системе AIX находится в каталоге **/etc/rc/tcpip**.

Мы полагаем, что такое расположение файлов неразумно, потому что операционные системы во многом уже стандартизировали расположение файлов с учетом пакета BIND, а в системе AIX это еще не сделано. Имена файлов и их расположение в системе AIX приведены в табл. 17.21.

**Таблица 17.21. Файлы пакета BIND в системе AIX**

| Файл | Каталог | Описание |
|---|---|---|
| resolv.conf | /etc | Библиотечный файл конфигурации распознавателя |
| named8 | /usr/sbin | Демон сервера имен BIND 8 |
| named9 | /usr/sbin | Облегченный распознаватель BIND 9 |
| named.conf | /etc | Файл конфигурации для сервера имен |
| netsvc.conf | /etc | Файл переключения служб |
| irc-conf | /etc | Другой файл переключения служб |
| NSORDER | окружение | Переменная переключения окружения |

Обычно в поставку системы AIX входит три механизма, реализующие концепцию "переключения службы", с помощью которых можно указать, к каким каталогам должны обращаться службы. Переменная среды NSORDER замещает информацию, заданную в файлах **/etc/netsvc.conf** и **/etc/irs.conf**.

Эти механизмы не только размещаются в разных местах, но и имеют разный синтаксис. Например, для того чтобы выбрать поиск имен узлов в системе DNS в файле **netsvc.conf**, можно использовать переменную bind, но то же самое можно сделать и с помощью файла **irs.conf**, в котором хранится значение dns. Этот синтаксис позволяет определить, что делать, если предпочитаемая служба не находит ответа, но он отличается от синтаксиса файла **nsswitch.conf** в других системах. Детали механизма переключения служб приведены в разделе 19.5.

# 17.17. Рекомендуемая литература

Система доменных имен и пакет BIND описаны во многих источниках, включая документацию, входящую в состав пакета, отдельные главы в книгах об Интернете, целую книгу серии "In a Nutshell" издательства O'Reilly, а также многочисленные ресурсы в глобальной сети. Серверы NSD и Unbound являются относительно новыми, поэтому внешних источников, посвященных их описанию, относительно немного, но мы нашли одну книгу, которая охватывает сразу несколько веб-документов.

## Списки рассылки и новостные группы

Ниже перечислены списки рассылки, посвященные пакету BIND.

- bind-announce — чтобы подписаться, отправьте сообщение по адресу bind-announce-request@isc.org
- namedroppers — чтобы подписаться, отправьте сообщение по адресу namedroppers-request@internic.net
- bindusers — чтобы подписаться, отправьте сообщение по адресу bind-users-request@isc.org
- bind9-workers — чтобы подписаться, отправьте сообщение по адресу bind9-workers-request@isc.org (для разработчиков)

Сообщения об ошибках следует слать по адресу bind9-bugs@isc.org.
Ниже перечислены списки рассылки, посвященные пакету NSD/Unbound.

- nsd-users — чтобы подписаться, зайдите на веб-сайт nlnetlabs.nl/projects/nsd
- unbound-users — чтобы подписаться, зайдите на веб-сайт unbound.net
- ldns-users — чтобы подписаться, зайдите на веб-сайт nlnetlabs.nl/projects/ldns
- drill — чтобы подписаться, зайдите на веб-сайт nlnetlabs.nl/projects/drill

И вот список рассылки, посвященный службе DNS и функционированию чрезвычайно важных сайтов (регистраторов, корневых серверов, серверов TLD и т.д.).

- dns-operations — чтобы подписаться, зайдите на веб-сайт lists.dns-oarc.net

## Книги и другая документация

- The Nominum BIND Development Team. *BINDv9 Administrator Reference Manual.*
- Документ включен в дистрибутив BIND (**doc/arm**), а также доступен на веб-узле www.isc.org. В нем в общих чертах описаны принципы администрирования пакета BIND 9.

- Reed, Jeremy C., editor. *BIND 9 DNS Administration Reference Book*. redwood City, CA: Reed Media Services, 2007.
- Albitz, Paul and Cricket Liu. *DNS and BIND, 4th Edition*. Sebastopol, CA: O'Reilly, 2001.

  Это очень популярная книга о пакете BIND, содержащая описание версий BIND 8 и BIND 9. Полнота изложения впечатляет.
- Liu, Cricket. *DNS & BIND Cookbook. Sebastobol*, CA: O'Reillymedia, 2002.

  Это упрощенная версия книги о системе DNS, выпущенной издательством O'Reilly, содержащая четкие и ясные инструкции, а также примеры разных операций на серверах имен. Она несколько устарела, но все еще полезна.
- Atchison, Ron. *Pro DNS and BIND*. Berkeley, CA: Apress, 2005.

  Это новая книга о системе DNS, содержащая очень хороший раздел о протоколе DNSSEC с примерами и описание стратегии развертывания. Мы нашли несколько опечаток, но, к счастью, автор поддерживает веб-сайт, посвященный ошибкам. По глубине материала и организации эта книга выгодно отличается от книги *DNS and BIND*, но администратору DNS лучше иметь обе эти книги!
- Mens, Jan-Piet. *Alternative DNS Servers: Choice and Deployment, and Optional SQL/LDAP Back-Ends*. Camridge, England: UIT Cambridge Ltd., 2009.

  Эта книга описывает около 10 разных реализаций серверов имен, включая NSD/Unbound. В ней изложены разные вопросы хранения данных о зоне, приведены прекрасные диаграммы и дано много полезной информации.

## Ресурсы в Интернете

Каталог ресурсов DNS (`dns.net/dnsrd`) — это полезная коллекция ресурсов и ссылок на них, поддерживаемая Андрашем Саламоном (Andras Salamon).

Веб-сайты `isc.org`, `dns-oarc.net`, `ripe.net`, `nlnetlabs.nl`, `f.root-servers.org` и `k.root-servers.org` содержат много информации о системе DNS, исследованиях, результатах измерений, презентациях и др.

В поисковой системе Google индексированный список ресурсов DNS находится по следующему адресу:

```
http://directory.google.com/Top/Computers/Internet/Protocols/DNS
```

Подробная информация о протоколе DNS, записях о ресурсах и других аспектах DNS приведена на сайте `iana.org/assignments/dns-parameters`. Этот документ содержит информацию о том, в каком документе RFC описан тот или иной факт, касающийся работы системы DNS.

Справочник *DNSSEC HOWTO*, написанный Олафом Колкманом (Olaf Kolkman), — это 70-страничный документ, в котором изложены все аспекты развертывания и отладки протокола DNSSEC. Его можно загрузить с сайта `nlnetlabs.nl/dnssec_howto/dnssec_howto.pdf`.

## Документы RFC

Документы RFC, в которых описывается система доменных имен, доступны на веб-узле `www.rfc-editor.org`. Мы использовали только самые важные из них, хотя в настоящее время есть около 100 документов и около 50 черновиков, опубликованных в Интернете, поэтому рекомендуем читателям зайти на сайт `www.rfc-editor.org` и изучить

весь архив. Для того чтобы увидеть все документы, касающиеся текущей версии BIND, найдите каталоги `doc/rfc` и `doc/draft`.

Исходные спецификации 1987 года:

- RFC1034 — *Domain Names: Concepts and Facilities* (Доменные имена: концепции и базовые возможности).
- RFC1035 — *Domain Names: Implementation and Specification* (Доменные имена: реализация и спецификация).

## 17.18. Упражнения

17.1. Поясните назначение следующих записей о ресурсах: SOA, PTR, A, MX и CNAME.

17.2. Что такое связующие записи и зачем они нужны? Найдите с помощью команд **`dig`** или **`drill`** записи, связывающие вашу локальную и родительскую зоны.

17.3. В чем смысл отрицательного кеширования? Почему оно так важно?

17.4. Создайте записи SPF для своего сайта, чтобы контролировать спам.

17.5. ★ Какие действия необходимо предпринять для создания домена второго уровня? Назовите технические и организационные мероприятия.

17.6. ★ В чем разница между авторитетным и неавторитетным ответом на DNS-запрос? Как убедиться в том, что ответ действительно авторитетен?

17.7. ★ Какой компьютер является вашим локальным сервером имен? Какие действия придется ему предпринимать для распознавания имени `www.admin.com` (предполагается, что ни в одном из кешей DNS информации об этом домене нет)?

17.8. ★ Объясните, какое влияние на DNS оказывает 512-байтовое ограничение на размер пакета, налагаемое протоколом UDP? Какие существуют решения потенциальных проблем?

17.9. ★ Изучите 512-битовый русский алгоритм GOST или 256-битовый алгоритм NIST P-256 ECDSA и оцените их влияние на 512-байтовое ограничение размера пакетов UDP. Могут ли они обеспечить требуемый размер пакета? Насколько длинными являются ключи и подписи?

17.10. ★ Создайте записи SSHFP для своего сайта и обновите свою команду **`ssh`** для их использования.

17.11. ★ Используйте сервер для оценки размера пакетов ISC DNS-OARC из разных точек своего сайта, чтобы определить, какая локальная конфигурация может запретить развертывание протокола DNSSEC. Какой размер вы видите? Изменяется ли он при изменении вашего расположения? Какой размер вы видите из главного узла? Соберите такие же данные с помощью инструментов SecPider или **`dnsfunnel`**. Согласуются ли ваши данные? Если нет, то какой инструмент дает более точную информацию?

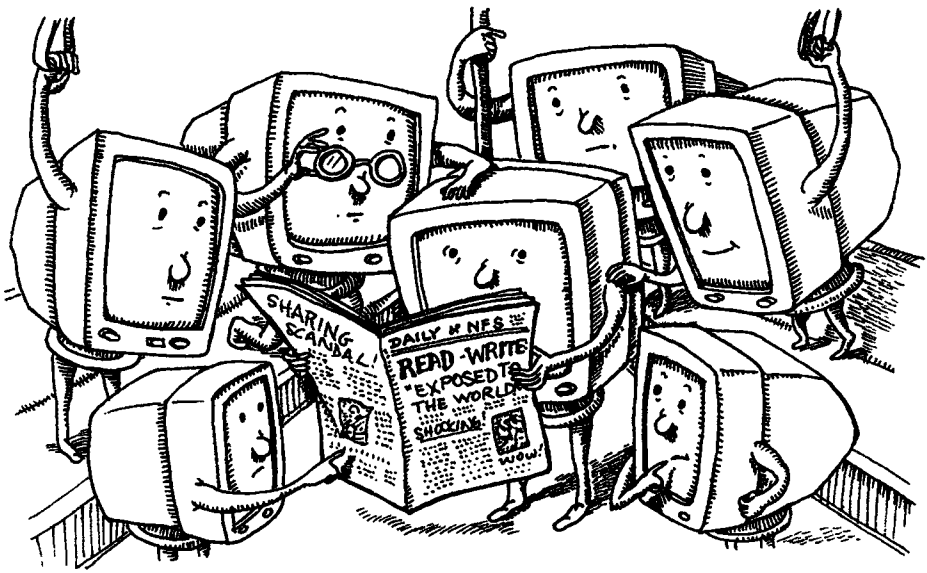
17.12. ★ Используя инструменты DNSSEC или библиотеки, создайте сценарий, определяющий синхронизацию между защищенным сайтом и записями DS на его родительском сервере, а также срок действия подписи. Ежедневно выполняйте этот сценарий с помощью планировщика заданий **`cron`**.

17.13. ★ Создайте записи DKIM для своего домена и настройте почтовые серверы и клиенты на работу с ним.

17.14. ★ Создайте поддомен вашего сайта. Добавьте реальный сайт со множеством имен и адресов, затем защитите его с помощью протокола DNSSEC и соедините с надежной сетью в Интернете, создав запись DLV на сайте консорциума ICS. Включите регистрацию и несколько дней наблюдайте за регистрационными записями. Задокументируйте все свои действия и проблемы

# Глава 18

# Сетевой протокол Network File System




Сетевой протокол Network File System, или NFS, позволяет пользователям совместно работать с файлами, расположенными на разных компьютерах. Протокол NFS почти прозрачен для пользователей, т.е. при сбое сервера, поддерживающего протокол NFS, информация не пропадает. Клиенты просто ждут, когда сервер вновь начнет функционировать, а затем продолжают работать так, будто ничего не произошло.

Протокол NFS разработала компания Sun Microsystems в 1984 году. Первоначально протокол NFS был реализован как суррогат файловой системы для бездисковых клиентов, однако предложенный протокол оказался столь удачным, что со временем стал универсальным решением проблемы совместного использования файлов. Все дистрибутивные пакеты систем UNIX и Linux содержат версию протокола NFS; многие из них используют лицензию компании Sun. В настоящее время протокол NFS открыто описан в документах RFC (см. RFC1094, RFC1983 и особенно RFC 3530).

## 18.1. Введение в протокол NFS

Совместное использование файлов в сети выглядит простой задачей, но на самом деле она представляет собой сложную проблему, имеющую множество вариантов решения и нюансов. В качестве свидетельства сложности этой задачи напомним, что многочисленные ошибки протокола NFS проявились в необычных ситуациях только спустя четверть века его использования. Современные администраторы могут быть уверены в том, что большинство протоколов совместного использования файлов (NFS и CIFS) не

портят данные и не вызывают ярость пользователей, но для того, чтобы этого достичь, пришлось немало потрудиться.

## Проблемы, связанные с состоянием

При разработке файловой системы необходимо решить, какая ее часть будет отслеживать файлы, открываемые каждым клиентом. Информация об этих файлах называется "состоянием" (state). Сервер, не делающий записей о статусе файлов и клиентов, называется сервером без сохранения состояния (stateless), а сервер, который решает эту задачу, — сервером с сохранением состояния (stateful). Многие годы использовались оба этих подхода, причем каждый из них имеет как преимущества, так и недостатки.

Серверы с сохранением состояния отслеживают все открытые файлы в сети. Этот режим работы вызывает множество проблем (больше, чем можно было бы ожидать) и затрудняет восстановление в случае краха. Когда сервер возвращается из небытия, клиент и сервер должны заново согласовать, какое состояние следует считать последним перед крахом. Серверы без сохранения состояния позволяют клиентам лучше контролировать файлы и облегчают управление файлами, открытыми в режиме чтения/записи.

На сервере без сохранения состояния каждый запрос не зависит от предшествующих запросов. Если сервер или клиент терпит крах, то никаких потерь для процесса это не влечет. В этом случае сервер безболезненно терпит крах и перезагружается, поскольку никакого контекста нет. Однако в этом случае сервер не может знать, какие клиенты открыли файлы для записи, поэтому не способен управлять параллельной работой.

## Проблемы производительности

Пользовательский интерфейс сетевых файловых систем не должен отличаться от пользовательского интерфейса локальных файловых систем. К сожалению, глобальные сети имеют большое время ожидания, что приводит к неправильному выполнению операций и сужению полосы пропускания. Все это в итоге приводит к снижению производительности работы с большими файлами. Большинство протоколов файловых служб, включая NFS, реализуют методы минимизации потерь производительности как в локальных, так и глобальных сетях.

Большинство протоколов пытается минимизировать количество запросов в сети. Например, предварительное кеширование предусматривает загрузку фрагментов файла в буфер локальной памяти, чтобы избежать задержки при считывании нового раздела файла. Часть полосы пропускания используется для того, чтобы избежать обмена информацией с сервером в ходе общего цикла сканирования сети (full round trip exchange). Кроме того, некоторые системы кешируют записи в памяти и посылают их обновления в пакетах, уменьшая тем самым задержку, вызванную необходимостью выполнить операции записи на сервере. Эти пакетные операции обычно называют объединением запросов (request coalescing).

## Безопасность

Любая служба, предоставляющая удобный доступ к файлам в сети, является источником серьезных угроз для безопасности. Локальные файловые системы реализуют сложные алгоритмы управления доступом, наделяя пользователей разными правами. В сети эти проблемы многократно усложняются, поскольку существуют требования, предъявляемые к быстродействию машин, а также имеются различия между их конфигура-

циями, ошибки в программном обеспечении файловых систем и нерешенные вопросы, связанные с протоколами совместного использования файлов.

Развитие служб каталогов и централизованной аутентификации повысило безопасность сетевых файловых систем. По существу, ни один клиент не может аутентифицировать себя самостоятельно, поэтому необходима доверенная централизованная система, верифицирующая личности и предоставляющая им доступ к файлам. Сложная организация этих служб замедляет их адаптацию, но в настоящее время централизованное управление доступом в той или иной степени реализовано в большинстве организаций.

# 18.2. Серверная часть NFS

Новейшая версия протокола NFS является платформенно-независимой, обеспечивает высокую производительность в глобальных сетях, таких как Интернет, а также гарантирует высокую безопасность. Большинство его реализаций также содержат диагностические инструменты для решения проблем, связанных с настройкой и производительностью. Клиентское и серверное программное обеспечение расположено в ядре. Однако эти части протокола NFS не требуют настройки и прозрачны для администратора.

## Версии и история протокола

Первая открытая версия протокола NFS, появившаяся в 1989 году, имела номер 2. Клиент второй версии NFS не мог считать операцию записи завершенной, не получив подтверждения от сервера. Для того чтобы избежать сбоя, сервер должен был записывать каждый модифицированный блок на диск, прежде чем ответить. Такое ограничение существенно замедляло запись, поскольку во многих случаях модифицированные блоки вполне можно было хранить лишь в буферном кеше.

В версии 3, появившейся в начале 1990-х годов, этот недостаток был устранен за счет схемы согласования, которая позволяла выполнять запись асинхронно. Были улучшены также другие аспекты протокола, связанные с производительностью и обработкой больших файлов, вследствие чего версия NFS 3 стала работать гораздо быстрее, чем NFS 2. По этой причине все организации должны использовать версии NFS 3 или 4.

Версия NFS 4 является результатом крупной переработки протокола; она содержит много усовершенствований и новых функциональных возможностей.

- Совместимость и взаимодействие со всеми брандмауэрами и устройствами NAT.
- Интегрирование в основном протоколе NFS протоколов блокировки и монтирования.
- Операции с сохранением состояния.
- Высокая интегрированная безопасность.
- Поддержка репликации и миграции.
- Поддержка клиентов как UNIX, так и Windows.
- Списки управления доступом (ACL).
- Поддержка файлов в кодировке Unicode.
- Хорошая производительность даже при узкой полосе пропускания.

Несмотря на то что во многих аспектах версия 4 представляет собой шаг вперед, изменения, внесенные в протокол, не изменили существенно процесс конфигурирования и администрирования протокола NSF.

Разные версии протокола не совместимы друг с другом, но серверы, поддерживающие протокол NFS (включая все рассмотренные нами операционные системы), обычно реализуют три из них. На практике все клиенты и серверы протокола NFS могут взаимодействовать с помощью одной из этих версий. Если и клиентская, и серверная стороны поддерживают протокол NFS 4, следует использовать именно эту версию.

## Транспортные протоколы

В версии NFS 2 применялся протокол UDP, поскольку он обеспечивал наилучшую производительность в локальных сетях 80-х годов. Несмотря на то что протокол NFS сам выполняет сборку пакетов и осуществляет контроль ошибок, ни в UDP, ни в NFS не реализованы алгоритмы управления перегрузкой, которые необходимы для достижения нормальной производительности в крупных IP-сетях.

С целью решения этих проблем в большинстве UNIX-систем теперь разрешено использовать в качестве транспортного протокола в версии NFS 3 как UDP, так и TCP, а в версии NFS 4 — только TCP.[1] Первоначально эта возможность предназначалась для обеспечения работы NFS в сетях с маршрутизаторами и в Интернете. По мере удешевления памяти и роста производительности центральных процессоров и сетевых контроллеров, первоначальное преимущество протокола UDP над протоколом TCP улетучилось.

## Состояние

Прежде чем начать работать с файловой системой NFS, клиент должен ее смонтировать, как если бы это была файловая система, находящаяся на локальном диске. Однако версии NFS 2 и 3 не сохраняют состояние, поэтому сервер не "знает", какие клиенты смонтировали ту или иную файловую систему. Вместо этого сервер вручает клиенту секретный ключ по факту успешного завершения операции монтирования. Этот ключ идентифицирует каталог монтирования на сервере NFS и дает клиенту возможность получить доступ к содержимому каталога. Ключ сохраняется при перезагрузке, чтобы сервер после сбоя смог вернуться в предыдущее состояние. Клиент может просто подождать, пока сервер перезагрузится, и повторить свой запрос.

С другой стороны, версия NFSv4 представляет собой протокол с сохранением состояния: и клиент, и сервер хранят информацию об открытых файлах и блокировках. При сбое сервера клиент облегчает процесс восстановления, посылая на сервер информацию о состоянии, предшествующем сбою. Восстанавливающийся сервер ожидает в течение заданного периода отсрочки, пока бывшие клиенты отчитаются о своем предыдущем состоянии, прежде чем приступить к выполнению новых операций и блокировок. Управления ключами, которое существовало в версиях V2 и V3, в версии NSFv4 больше нет.

## Экспорт файловой системы

Говорят, что сервер "экспортирует" файловую систему, если он делает ее доступной для использования другими компьютерами. По определению, все серверы экспортируют хотя бы один каталог. В версиях V2 и V3 каждый экспорт рассматривается как независимая сущность. В версии V4 каждый сервер экспортирует одну иерархическую псевдофайловую систему, содержащую все свои экспортируемые каталоги. По существу, псев-

[1] С формальной точки зрения, можно использовать любой транспортный протокол, реализующий алгоритм управления перегрузкой, но в настоящее время единственным разумным выбором является только протокол TCP.

дофайловая система — это пространство имен файловой системы сервера, из которой удалено все, что не подлежит экспорту.

В качестве примера рассмотрим следующий список каталогов, в котором экспортируемые каталоги выделены полужирным шрифтом.

**/www/domain1**
**/www/domain2**
/www/domain3
**/www/logs/httpd**
/var/spool

В версии NFS 3 каждый экспортируемый каталог должен конфигурироваться отдельно. Клиентские системы должны выполнить три разных запроса на монтирование, чтобы получить доступ ко всему экспорту сервера.

В то же время в версии NFS 4 псевдофайловая система соединяет разрозненные части структуры каталогов и создает единое представление для клиентов NFS. Вместо запроса на монтирование каждого из каталогов **/www/domain1**, **/www/domain2** и **/var/logs/httpd**, клиент может просто смонтировать серверный псевдокорневой каталог и просмотреть всю иерархию.

Каталоги **/www/domain1** и **/var/spool**, не подлежащие экспорту, не появляются в этой иерархии. Кроме того, отдельные файлы, содержащиеся в каталогах **/**, **/var**, **/www** и **/var/logs**, не видны клиенту, потому что псевдофайловая часть иерархии состоит только из каталогов. Таким образом, при использовании версии NFSv4 клиент видит экспортируемую файловую систему в следующем виде.

```
/
   /www
     /www.domain1
     /www.domain2
   /var
     /var/logs
       /var/logs/httpd
```

Сервер задает корень экспортируемой файловой системы в файле конфигурации под названием **exports**.

## Блокировка файлов

Блокировка файлов (реализуемая системными вызовами **flock**, **lockf** или **fcntl**) в течение долгого времени была слабым звеном в системах UNIX. В локальных файловых системах этот механизм был воплощен далеко не идеально. В ранних версиях протокола NFS серверы не сохраняли состояние: им не известно, какие компьютеры работают с конкретным файлом. Однако эта информация необходима для использования блокировок. Как быть?

Традиционное решение заключается в реализации механизма файловых блокировок отдельно от протокола NFS. В большинстве систем этой цели служат два демона: **lockd** и **statd**. К сожалению, по ряду причин они работают не оптимально, и блокировка файлов в протоколе NFS оставалась его слабым местом.

В версии NFSv4 демоны **lockd** и **statd** больше не используются для блокировки в основном протоколе (следовательно, серверы сохраняют состояние). Это изменение значительно усложнило протокол, но устранило множество проблем, характерных для более ранних версий протокола NFS. К сожалению, по отдельности демоны **lockd** и **statd** все еще нужны для поддержки клиентов, использующих версии V2 и V3. Все рас-

сматриваемые нами примеры операционных систем содержат ранние версии протокола NFS, поэтому отдельные демоны по-прежнему запускаются по умолчанию.

## Вопросы безопасности

Во многих аспектах версии V2 и V3 протокола NFS были типичными наследниками всех недостатков, касающихся вопросов безопасности в системах UNIX и Linux. Этот протокол изначально не предполагал никаких мер безопасности, и благодаря этому он был удобным. Версия NFSv4 устранила дефекты системы безопасности, которые были присущи предыдущим версиям, обеспечив сильную поддержку службам безопасности и внедрив более совершенную идентификацию пользователей.

Все версии протокола NFS не зависят от механизма, обеспечивающего безопасность работы в сети, и большинство серверов поддерживают разные режимы аутентификации. Некоторые из этих режимов перечислены ниже.

- AUTH_NONE — без аутентификации.
- AUTH_SYS — способ управления доступом для пользователей и групп, напоминающий стиль системы UNIX.
- RPCSEC_GSS — строгий режим аутентификации, предполагающий целостность и закрытость в дополнение к аутентификации.

Традиционно в большинстве организаций применяется режим AUTH_SYS, который использует идентификаторы групп и пользователей в системе UNIX. В этой схеме при запросе к серверу клиент просто посылает локальный идентификатор пользователя и группы. Сервер сравнивает значения этих идентификаторов со значениями из своего файла **/etc/passwd** [2] и определяет, следует ли предоставлять доступ данному пользователю. Таким образом, если два пользователя имеют одинаковые идентификаторы на двух разных клиентах, то они получат доступ ко всем файлам друг друга. Более того, пользователи, имеющие права администратора системы, могут применить команду **su** к любому идентификатору по своему усмотрению; после этого сервер предоставит им доступ к соответствующим файлам.

Согласование файла **passwd** со всеми системами играет очень существенную роль в средах, использующих режим AUTH_SYS. Однако даже это — всего лишь иллюзия безопасности; любой мошеннический узел (или Windows-машина) может "аутентифицировать" своих пользователей по своему желанию и тем самым разрушить безопасность протокола NFS.

Для того чтобы предотвратить эти проблемы, большинство организаций использует более надежный механизм идентификации, такой как протокол Kerberos в сочетании со слоем NFS RPSSEC_GSS. Эта конфигурация требует от клиента и сервера совместного участия в работе механизма Kerberos. Механизм Kerberos аутентифицирует клиентов централизованно, тем самым предотвращая возможность самоидентификации, описанной выше. Кроме того, протокол Kerberos обеспечивает сильное шифрование и гарантирует целостность файлов, передаваемых по сети. Все системы, поддерживающие протокол NFS версии 4, должны реализовать режим RPCSEC_GSS, в то же время в версии 3 это требование не является строгим.

🕮 Более подробная информация о протоколе Kerberos изложена в разделе 22.10.

Доступ к томам системы NFS обеспечивает файл **/etc/exports**, в котором перечислены имена узлов (или IP-адреса), которые имеют право на доступ к совместно ис-

[2] Или его эквивалента в виде сетевой базы данных, такой как NIS или LDAP.

пользуемым файловым системам сервера. К сожалению, это слишком слабый механизм безопасности, поскольку сервер доверяет клиентам, называющим свой идентификатор. Клиента легко заставить солгать и указать неверный идентификатор и IP-адрес, поэтому этот механизм не слишком надежен. Тем не менее файловые системы следует экспортировать только клиентам, заслуживающим доверия, и всегда следует проверять, не экспортировали ли вы случайно файловые системы всему миру.

Протокол NFSv4 использует в качестве транспортного протокола только TCP и обычно осуществляет связь через порт 2049. Поскольку протокол NFSv4 не использует другие порты, открыть доступ через брандмауэр так же просто, как открыть протокол TC через порт 2049. Как и в случае со списком всех конфигураций доступа, кроме порта, следует идентифицировать адреса источника и пункта назначения. Если ваша организация не планирует предоставлять услуги протокола NFS узлам, расположенным в Интернете, заблокируйте доступ через брандмауэр или используйте фильтр локальных пакетов.

🕮 Более подробная информация о брандмауэрах приведена в разделе 22.11.

Мы не рекомендуем использовать для организации файловой службы в глобальных сетях протоколы NFSv2 и NFSv3, поскольку в протоколах RPC много ошибок и нет сильных механизмов безопасности. Администраторы серверов, работающих по протоколу NFS 3, должны блокировать доступ к TCP- и UDP-портам 2049, а также к порту 111 сервера **portmap**.

## Идентифицирующее отображение в версии 4

Как указывалось в главе 7, операционная система UNIX идентифицирует пользователей с помощью набора индивидуальных и групповых идентификаторов, записанных в локальном файле **passwd** или административной базе данных. С другой стороны, версия 4 протокола NFS представляет пользователей и группы в виде строковых идентификаторов, имеющих вид `пользователь@nfs-домен` или `группа@nfs-домен`. И клиенты, и серверы, работающие по протоколу NFS, запускают демона идентифицирующего отображения (identity mapping daemon), который отображает значения идентификаторов UNIX в строки.

Когда клиент, использующий версию 4, выполняет операцию, возвращающую идентификаторы, например команду **ls -l**, возвращающую листинг файла, демон идентифицирующего отображения использует свой локальный файл **passwd** для превращения пользовательских и групповых идентификаторов каждого файлового объекта в строку, например `ben@atrust.com`. Затем отображение идентификаторов клиентов выполняет обратное действие, превращая строку `ben@atrust.com` в локальные пользовательские и групповые идентификаторы, которые могут совпадать, а могут и не совпадать с идентификаторами сервера. Если строковое значение не соответствует ни одной локальной сущности, используется учетная запись анонимного пользователя.

В этот момент вызов удаленной файловой системы (**stat**) завершается и возвращает вызвавшей ее команде (в данном случае команде **ls**) значения пользовательского и группового идентификаторов. Однако, поскольку команда **ls** была выполнена с флагом **-l**, она должна вывести на экран текстовые имена, а не числа. Итак, команда **ls**, в свою очередь, переводит идентификаторы обратно в текстовые имена, используя библиотечные утилиты **getpwuid** и **getgrgid**. Эти утилиты еще раз проверяют файл **passwd** или эквивалентную ему базу данных. Какая долгая и запутанная процедура!

К сожалению, идентифицирующее отображение используется только при извлечении и записи атрибутов файлов, как правило, используемых монопольно. *Идентифицирующее*

*отображение не играет никакой роли в процессе аутентификации и управлении доступом.* Оно лишь переводит идентификаторы в форму, принятую в протоколе RPC. Следовательно, согласованность файлов **passwd** по-прежнему важна для пользователей, работающих в режиме "безопасности" AUTH_SYS.

В системах, не синхронизирующих файлы **passwd**, вследствие неоднозначности идентификаторов и аутентификации возникает побочный эффект. Идентифицирующее отображение может выполнять отображение точнее, чем базовый протокол NFS, вызывая противоречие между прямым доступом к файлам и доступом, на самом деле предоставляемым сервером, работающим по протоколу NFS. В качестве примера рассмотрим следующие команды клиента протокола NFSv4.

```
[ben@nfs-client]$ id ben
uid=1000(ben) gid=1000(ben) groupd=1000(ben)

[ben@nfs-client]$ id john
uid=1010(john) gid=1010(john) groupd=1010(john)

[ben@nfs-client]$ ls -ld ben
drwxr-xr-x 2 john root      4096  May 2007  16:42   ben

[ben@nfs-client]$ touch ben/file
[ben@nfs-client]$ ls -l ben/file
-rw-rw-r--  1 john  nfsnobody 0    May 2007  17:07   ben/file
```

Мы видим, что пользователь ben имеет идентификатор 1000, а john — 1010. Рабочий каталог **ben**, экспортированный по протоколу NFS, имеет разрешение 775 и принадлежит пользователю john. Однако пользователь ben может создать файл в этом каталоге, даже если вывод команды **ls -l** означает, что у него нет прав на запись.

На сервере пользователь john имеет идентификатор 1000. Поскольку на клиентском компьютере идентификатор пользователя john равен 1010, идентифицирующее отображение выполняет преобразование идентификатора, как описано выше, и в результате пользователь john оказывается владельцем каталога. Однако демон идентифицирующего отображения не участвует в управлении доступом. Для создания файла непосредственно на сервер посылается идентификатор пользователя ben, равный 1000, который интерпретируется как идентификатор пользователя john.

Как узнать, какие операции используют идентифицирующее отображение, а какие нет? Это просто: как только пользовательский или групповой идентификатор появляется в *интерфейсе файловой системы* (например, в сочетании с командой **stat** или **chown**), он отображается. Если пользовательские или групповые идентификаторы *неявно* используются для управления доступом, они направляются через специальную систему аутентификации.

К сожалению для администраторов, демоны идентифицирующего отображения не стандартизованы, поэтому их процессы конфигурирования на разных система могут быть разными. Специфика каждой системы описывается в разделе 18.5.

## Учетные записи root и nobody

В общем случае пользователи должны иметь одинаковые привилегии в любой системе, к которой они получают доступ. Однако традиционно системные администраторы стараются ограничивать возможности неконтролируемого применения прав привилегированного пользователя в файловых системах, смонтированных посредством NFS. По умолчанию сервер NFS перехватывает входящие запросы, посылаемые от имени пользо-

вателя с идентификатором 0, и "делает вид", будто они поступают от другого пользователя. Этот прием называется "поражением в правах" ("squashing root"). Таким образом, учетная запись **root** не отменяется, но уравнивается в правах с обычными учетными записями.

Для этого специально определена фиктивная учетная запись **nobody**, чтобы под нее "маскировался" пользователь **root**, работающий на сервере NFS. Традиционно ее идентификатор равен 65534 (обратный код числа -2)[3]. Значения UID и GID для пользователя **root** в файле **exports** можно изменить. С помощью опции `all_squash` можно заставить систему преобразовать все клиентские идентификаторы пользователей в одинаковый серверный идентификатор. Если идентификатор равен -1, то в системах Solaris и HP-UX доступ вообще запрещается.

Все эти предосторожности преследуют благородную цель, однако конечный результат далек от идеала. Даже будучи клиентом NFS, пользователь **root** может с помощью команды **su** "принять облик" любого пользователя, так что файлы никогда не бывают полностью защищены. Единственный эффект от смены идентификаторов заключается в том, что предотвращается доступ к файлам, которые принадлежат пользователю **root** и недоступны для чтения или записи остальным пользователям.

## Производительность версии 4

Протокол NFSv4 был разработан для того, чтобы обеспечить высокую производительность в глобальных сетях. Большинство глобальных сетей имеет более долгое время ожидания и меньшую пропускную способность, чем локальные сети. Протокол NFS должен был решить эти проблемы с помощью следующих усовершенствований.

- Процедура протокола RPC под названием COMPOUND включает несколько файловых операций в один запрос, сокращая время ожидания при выполнении многочисленных сетевых запросов.
- Механизм делегирования позволяет кеширование файлов на клиентской стороне. Клиенты могут сохранять контроль над локальными файлами, включая их открытие для записи.

Эти функциональные возможности являются частью ядра протокола NFS и не требуют специального внимания со стороны системного администратора.

## Дисковые квоты

Доступ к информации о дисковых квотах на удаленном компьютере осуществляет серверный демон **rquotad**, также работающий отдельно от NFS. Сервер NFS будет поддерживать дисковые квоты, но пользователи не смогут ничего о них узнать, если на удаленном компьютере не работает демон **rquotad**.

Мы считаем механизм дисковых квот морально устаревшим, поэтому не будем рассказывать об этом демоне. Однако некоторые организации по-прежнему поддерживают его работу, чтобы пользователи не захватили все доступное пространство на диске. Если вы поддерживаете работу сети в одной из таких организаций, то обратитесь к справочной странице, посвященной квотам. Больше мы к этому вопросу возвращаться не будем.

[3] На сервере NFS в Red Hat стандартное значение UID равно -2, тогда как в файле ***passwd*** учетной записи ***nobody*** присвоен идентификатор 99. Можно оставить все как есть, добавить в файл ***passwd*** запись с идентификатором -2 или задать параметры `anonuid` и `anongid` равными 99.

# 18.3. Серверная часть протокола NFS

Говорят, что сервер NFS "экспортирует" каталог, когда он делает этот каталог доступным для использования другими компьютерами. Системы Solaris и HP-UX вместо слова "экспорт" используют словосочетание "совместное использование". Для того чтобы не создавать путаницы, мы будем на протяжении этой главы использовать термин "экспорт".

В версии NFSv3 процесс, используемый клиентами для монтирования файловой системы (т.е. для того, чтобы узнать секретный ключ), отделен от процесса, используемого для доступа к файлам. Эти операции используют отдельные протоколы, а запросы обрабатываются разными демонами: **mountd** — для запросов на монтирование и **nfsd** — для реальной файловой службы. В некоторых системах эти демоны называются **rpc.nfsd** и **rpc.mountd**, поскольку они основаны на протоколе RPC (а значит, для их запуска нужен сервер **postmap**). В этой главе для простоты мы будем пропускать префикс **rpc**.

Версия NFSv4 не использует демон **mountd** вообще. Однако, если не все ваши клиенты используют версию NFSv4, демон **mountd** следует выполнять.

На сервере NFS демоны **mountd** и **nfsd** должны запускаться при загрузке системы и должны работать на протяжении всего времени ее функционирования. Сценарии загрузки системы обычно автоматически запускают этих демонов, если существует какая-либо конфигурация экспорта. Имена сценариев запуска сервера NFS для каждой из рассматриваемых нами платформ перечислены в табл. 18.1.

**Таблица 18.1. Сценарии запуска сервера NFS**

| Система | Путь к сценарию |
|---|---|
| Ubuntu | **/etc/init.d/nfs-kernel-server** |
| | **/etc/init.d/nfs-common** |
| SUSE | **/etc/init.d/nfsservera** |
| Red Hat | **/etc/rc.d/init.d/nfs** |
| Solaris | **/etc/init.d/nfs.server** |
| HP-UX | **/sbin/init.d/nfs.server** |
| AIX | **/etc/rc.nfs** |

[a] **/etc/init.d/nfs** монтирует клиентскую файловую систему NFS.

Протокол NFS использует единую базу данных для управления доступом, которая определяет, какие файловые системы должны быть экспортированы и какие клиенты должны их монтировать. Оперативная копия этой базы данных обычно хранится в файле **xtab** (**sharetab** — в системах Solaris и HP-UX), а также во внутренних таблицах ядра. Поскольку файлы **xtab** и **sharetab** не предназначены для чтения людьми, для добавления и модификации записей в них используются вспомогательные команды **exports** или **share**. Для удаления записей из таблицы экспорта используются команды **exportfs -u** или **unshare**.

Монтирование бинарного файла вручную — неблагодарная задача, поэтому в большинстве систем предполагается, что пользователь монтирует текстовый файл, а не перечисляет экспортируемые системные каталоги и их установки доступа. Система может прочитать этот текстовый файл при загрузке и автоматически создать файл **xtab** или **shatretab**.

В большинстве систем каталог **/etc/exports** является каноническим списком, доступным для чтения экспортируемых каталогов. Его содержимое можно прочитать с помощью команды **exportfs -a**. В системах Solaris и HP-UX каноническим списком

является **/etc/dfs/dfstab**, который представляет сценарий, составленный из команд **share**. (Команда **shareall** сканирует в файле **dfstab** команды, связанные с протоколом NFS, и выполняет их. Поскольку протокол NFS — единственная "родная" система совместного использования файлов, подчиняющаяся этим правилам, команда **shareall** эквивалентна команде **sh /etc/dfs/dfstab**.)

Информация, приведенная в предыдущих разделах, подытожена в табл. 18.2. В ней указано, какой файл следует отредактировать, если вы хотите экспортировать новую файловую систему, и что делать, если вы хотите ввести в действие внесенные вами исправления.

**Таблица 18.2. Где задаются экспортируемые каталоги**

| Система | Путь к сценарию | |
|---|---|---|
| Linux | **/etc/exports** | Выполнить команду **/usr/sbin/exportfs -a** |
| Solaris | **/etc/dfs/dfstab** | Выполнить команду **shareall** |
| HP-UX | **/etc/dfs/dfstab** | Выполнить команду **shareall** |
| AIX | **/etc/exports** | Выполнить команду **/usr/sbin/exportfs -a** |

Протокол NFS работает с логическим уровнем файловой системы. Любой каталог можно экспортировать; он не обязан быть точкой монтирования или корнем физической файловой системы. Однако с точки зрения безопасности протокол NFS различает границы между файловыми системами и требует, чтобы каждое устройство было смонтировано отдельно. Например, на компьютере, имеющем отдельный раздел **/users**, можно было бы экспортировать корневой каталог без экспорта каталога **/users**. [4]

Клиентам обычно разрешается монтировать подкаталоги экспортируемых каталогов, если это необходимо, хотя протокол этого не требует. Например, если сервер экспортирует каталог **/chimchim/users**, то клиент может смонтировать только каталог **/chinchim/users/joe** и игнорировать остальную часть каталога **users**.

Большинство версий системы UNIX не позволяет экспортировать подкаталоги с разными опциями, но на практике в системе Linux это возможно.

## Команда share и файл dfstab (Solaris, HP-UX)

Solaris

Сценарий **/etc/dfs/dfstab** однократно выполняет команду **share** для каждой экспортируемой файловой системы. Например, на сервере, использующем каталог **/home** совместно с узлами monk и leopard (причем узел monk имеет права привилегированного пользователя) и каталог **/usr/share/man** — совместно с узлами ross и harp, файл **/etc/dfs/dfstab** может содержать следующие команды.

```
share -F -nfs -o rw=monk.atrust.com:leopard.atrust.com,root=monk.atrust.com
   /home
share -F -nfs -0 rw=ross.atrust.com:harp.atrust.com /usr/share/man
```

После редактирования файла **/etc/dfs/dfstab** следует обязательно выполнить команду **shareall**, чтобы ввести в действие сделанные изменения. Поскольку команда **shareall** просто выполняет команды, записанные в файле **dfstab**, она не выводит из совместного использования удаленные вами файловые системы. Для того чтобы удалить файловую систему из совместного использования, следует выполнить команду **unshare** */путь/к/файлам*. Наиболее полезные опции команды **share** перечислены в табл. 18.3.

[4] Разумеется, никогда не следует экспортировать корневой каталог.

**Таблица 18.3. Опции команды share (Solaris, HP-UX)**

| Опция | Описание |
|---|---|
| `ro` | Экспортирует файловую систему только для чтения всему миру (не рекомендуется) |
| `ro = список` | Экспортирует файловую систему только для чтения, открывая доступ только для пользователей, перечисленных в списке |
| `rw` | Экспортирует файловую систему для чтения и записи всему миру (не рекомендуется) |
| `rw = список` | Экспортирует файловую систему для чтения и записи, открывая доступ только для пользователей, перечисленных в списке |
| `root = список` | Перечисляет узлы, которым разрешен доступ к данной файловой системе с правами привилегированного пользователя; в противном случае привилегированный доступ клиента означает доступ "анонимного" пользователя (как правило, с идентификатором -2) |
| `anon = идентификатор` | Задает идентификатор, в который должен отображаться корень; по умолчанию считается анонимным |
| `nosub` | Запрещает клиентам монтировать подкаталоги в экспортированном каталоге |
| `nosuid` | Запрещает создавать файлы setuid и setgid в системе NFS |

Если в качестве параметра команды **share** указывается список, то он должен состоять из групп элементов, разделенных двоеточиями. В табл. 18.4 перечислены все способы, которыми можно задать узлы или группы узлов.

**Таблица 18.4. Спецификации клиента для команды share**

| Тип | Синтаксис | Описание |
|---|---|---|
| Имя узла | *имя_узла* | Отдельные узлы (должны быть полностью указаны) |
| Сетевая группа | *имя_группы* | Сетевые группы NIS (используется редко) |
| Домены DNS | *.домен.com* | Любой узел внутри домена |
| IP-сети | *@имя_сети* | Имена сетей, как указано в файле **/etc/networks** [a] |

[a] Допускаются также спецификации в стиле CIDR, например **@128.138.92.128/25**.

Замечание в табл. 18.4, касающееся имен узлов, можно назвать излишним: отдельные имена узлов *должны* быть указаны полностью или они будут проигнорированы.

Для того чтобы явно закрыть доступ к элементу, достаточно поставить перед его именем косую черту. Список просматривается слева направо во время каждого сканирования в поисках соответствующего элемента, поэтому символы отрицания должны предшествовать более общим элементам, которые они модифицируют. Например, строка

```
share -F -nfs -0 -rw=-@192.168.10.0/25:.booklab.atrust.com /users
```

экспортирует каталог **/users**, допускающий чтение и запись, во все узлы домена DNS book.atrust.com, за исключением узлов сети 192.168.10. В этой команде флаг **-F** означает, что команда **share** должна использовать файловую систему **nfs**, а не системы, перечисленные в файле **/etc/dfs/fstypes**.

Некоторым клиентам можно экспортировать каталог только для чтения, а другим — для чтения и записи. Для этого данных клиентов следует включить в опции **ro=** и **rw=** соответственно.

Справочная страница команды **share** содержит только несколько основных параметров протокола NFS. Полный список этих параметров можно найти на справочной странице команды **share_nfs**.

## Команда exports и файл exports (Linux, AIX)

Файл **exports** содержит список экспортированных каталогов в левом столбце и список связанных с ними параметров в правом. Например, строка в файле **exports** в системе AIX выглядит следующим образом.

```
/home    -vers=4,sec-sys,access=harp.atrust.com
```

Это позволяет монтировать каталог **/home** на машине harp.atrust.com с помощью протокола NFS и механизма аутентификации системы UNIX (sec-sys).

Файловые системы, перечисленные в файле **exports** без указания узлов, монтируются на *всех* компьютерах, что является значительной брешью в системе безопасности.

Параметры и синтаксические конструкции, используемые в файле **exports**, изменяются от системы к системе, хотя между ними существует определенная тематическая схожесть. В следующем разделе описываются общие форматы для систем AIX и Linux. Как всегда, точную информацию следует искать в справочной системе.

## Файл exports в системе AIX

Формат файла **exports** в системе AIX можно назвать самым классическим из всех рассматриваемых в книге систем. Допустимые параметры перечислены в табл. 18.5.

**Таблица 18.5. Параметры файла exports в системе AIX**

| Параметр | Описание |
|---|---|
| access=*список* | Перечисляет узлы, которые могут монтировать файловую систему |
| ro | Экспортирует файловую систему всем только для чтения; ни один клиент не может ничего записывать в эту файловую систему |
| rw | Экспортирует файловую систему всем для чтения и записи (по умолчанию) |
| rw=*список* | Экспортирует файловую систему, в основном, для чтения. Список перечисляет узлы, которым разрешается монтировать файловую систему для записи; все остальные клиенты должны только читать эту систему |
| root=*список* | Узлы, указанные в списке, могут монтировать файловую систему как привилегированный пользователь. Без этой опции привилегированный доступ клиента эквивалентен доступу анонимного пользователя |
| vers=*n* | Экспортирует каталог клиентам с помощью версии n. Корректными значениями считаются 2, 3 и 4 |
| sec=*режим* | Задает список методов обеспечения безопасности для экспортируемого каталога. Допускаются значения sys (аутентификация в системе UNIX), dh (DES), krb5 (аутентификация по протоколу Kerberos), krb5i (аутентификация и защита целостности по протоколу Kerberos), krb5p (аутентификация, также защита целостности и конфиденциальности по протоколу Kerberos) и none (анонимный доступ; не рекомендуется) |
| anon=*n* | Задает идентификатор, в который отображаются привилегированные пользователи. По умолчанию значение равно -2 (аноним). Задав значение -1, можно вообще запретить привилегированный доступ |

В табл. 18.5 *список* состоит из имен узлов и сетевых групп, разделенных двоеточиями. Параметры в табл. 18.5 похожи на параметры команды **share**. Однако между ними есть небольшая разница. Например, параметр

```
rw=leopard.atrust.com:ross.atrust.com
```

в команде **share** означает экспорт каталога для чтения и записи с предоставлением доступа только перечисленным узлам. В системе AIX этот параметр позволяет всем монтировать данный каталог только для чтения. Ясно! В системе AIX вы должны использовать параметр `access`, чтобы ограничить доступ к каталогу заданным списком клиентов.

```
rw,access=leopard.atrust.com:ross.atrust.com
```

По умолчанию каталог экспортируется для чтения и записи, поэтому параметр `rw` можно не указывать. Тем не менее его указание ошибкой не является.

Каждая строка в файле **exports** в системе AIX должна состоять из пути к каталогу, разделителя и дефиса, за которым следует список параметров, разделенных запятыми. Например, приведенная ниже команда открывает узлу `leopard.atrust.com` доступ к каталогу **/home** в режиме безопасности AUTH_SEC и в рамках четвертой версии протокола NFS.

```
/home -vers=4,sec=sys,rw,access=leopard.atrust.com
```

Помните, что после изменения файла **/etc/exports** следует выполнить команду **exportfs**.

## Файл exports в системе Linux

Как и в системе AIX, в системе Linux файл **/etc/exports** содержит перечисление файловых систем, экспортируемых в рамках протокола NFS, а также клиентов, которые могут иметь к ним доступ. Список клиентов отделен от списка файловых систем пробелами, и после имени каждого клиента в скобках задается список параметров, разделенных запятыми. Длинные строки можно разделять на несколько с помощью обратной косой черты.

Вот как выглядит этот формат.

```
/home            harp(rw,no_root_squash) monk(rw)
/usr/share/man   *.atrust.com(ro)
```

Этот формат не позволяет задать сразу несколько клиентов с одинаковыми наборами параметров, хотя некоторые спецификации клиентов можно распространить на несколько узлов. В табл. 18.6 приведены четыре типа спецификаций, которые могут появиться в файле **exports**.

**Таблица 18.6. Спецификации клиента в файле /etc/exports в системе Linux**

| Тип | Синтаксис | Описание |
|---|---|---|
| Имя узла | *имя_узла* | Отдельные узлы |
| Сетевая группа | *@имя_группы* | Сетевые группы NIS (используется редко) |
| Шаблонный символ | ** и ?* | Полностью определенные имена доменов * с шаблонными символами, причем символ "*" не может заменять точку |
| IP-сети | *ip_адрес/маска* | Имена сетей, как указано в файле /etc/networks [a] |

[a] Fully qualified domain names (FQDN).

В табл.18.7 перечислены основные параметры экспорта, используемые в системе Linux.

В системе Linux сервер NFS имеет необычные функциональные возможности, позволяющие экспортировать подкаталоги экспортируемых каталогов с разными параметрами. Для подкаталогов, которые вы предпочитаете исключить из общего использования, следует использовать параметр `noaccess`.

**Таблица 18.7. Основные параметры экспорта в системе Linux**

| Опция | Описание |
|---|---|
| `ro` | Экспорт только для чтения |
| `rw` | Экспорт для чтения и записи (по умолчанию) |
| `rw = список` | Экспорт преимущественно для чтения; в списке перечисляются узлы, которым разрешено монтировать файловую систему для записи; остальные должны монтировать ее только для чтения |
| `root_squash` | Отображает ("поражает в правах") идентификаторы UID 0 и GID 0 в значения, заданные параметрами `anonuid` и `anongid`. [a] Этот параметр задается по умолчанию |
| `no_root_squash` | Разрешает привилегированный доступ для всех. Опасно! |
| `all_squash` | Отображает все идентификаторы UID и GID в значения, установленные для анонимных пользователей. Целесообразно для персональных компьютеров и отдельных узлов, не вызывающих доверия |
| `anonuid=xxx` | Задает идентификатор UID для удаленных привилегированных пользователей, которых следует лишить привилегий |
| `anongid=xxx` | Задает идентификатор GID для удаленных привилегированных пользователей, которых следует лишить привилегий |
| `secure` | Допускает удаленный доступ только через привилегированный порт |
| `insecure` | Допускает удаленный доступ через любой порт |
| `noaccess` | Блокирует доступ к данному каталогу и подкаталогам (используется при вложенном экспорте) |
| `wdelay` | Откладывает запись в ожидании объединенных обновлений |
| `no_wdelay` | Немедленно записывает данные на диск |
| `async` | Заставляет сервер отвечать на запросы до того, как запись на диск будет выполнена в действительности |
| `nohide` | Выявляет файловые системы, смонтированные в дереве экспортированных файлов |
| `hide` | Противоположен по смыслу параметру `nohide` |
| `subtree_check` | Проверяет, находится ли запрашиваемый файл в экспортированном дереве |
| `no_subtree_check` | Проверяет лишь, относится ли запрашиваемый файл к экспортированной файловой системе |
| `secure_locks` | Требует авторизации для всех запросов на блокировку |
| `insecure_locks` | Использует менее строгие критерии блокировки (поддерживает старых клиентов) |
| `sec=`*режим* | Задает список методов обеспечения безопасности для экспортируемого каталога. Допускаются значения `sys` (аутентификация в системе UNIX), `dh` (DES), `krb5` (аутентификация по протоколу Kerberos), `krb5i` (аутентификация и защита целостности по протоколу Kerberos), `krb5p` (аутентификация, также защита целостности и конфиденциальности по протоколу Kerberos) и `none` (анонимный доступ; не рекомендуется) |
| `fsid=`*num* | Задает псевдофайловую систему в версии 4 (обычно 0) |

[a] В отличие от большинства операционных систем, Linux разрешает "поражение в правах" не только привилегированных клиентов. Более подробно это изложено в пункте `all_squash`.

Например, конфигурация

```
/home       *.atrust.com(rw)
/home/ben    (noaccess)
```

позволяет узлам в домене atrust.com иметь доступ ко всему содержимому каталога **/home**, за исключением подкаталога **/home/ben**. Отсутствие имени клиента во второй строке означает, что данный параметр относится ко всем узлам; возможно, тем самым можно достичь более высокой степени безопасности.

Параметр subtree_check (по умолчанию) проверяет, принадлежит ли файл, к которому обращается клиент, к экспортированному подкаталогу. Если этот параметр отключить, то будет проверяться только факт, что файл находится в экспортированной файловой системе. Проверка поддерева может вызвать неожиданные проблемы, если запрашиваемый файл был переименован, пока клиент держал его открытым. Если такие ситуации возникают часто, следует выбрать параметр no_subtree_check.

Параметр secure_locks требует авторизации и аутентификации при блокировке файла. Некоторые клиенты системы NFS не посылают мандатов вместе с запросами на блокировку и не работают с параметром secure_locks. В этом смысле существует возможность заблокировать файлы, доступные для всеобщего чтения. Лучше всего заменить этих клиентов теми, кто посылает мандаты. Однако в качестве временной меры можно использовать и параметр insecure_locks.

Демон **mountd** в системе Linux можно запускать без демона **inetd**. Эта конфигурация позволяет обеспечить дополнительный контроль за доступом с помощью сетевого анализатора **tcpd**. Более полная информация по этому вопросу приведена в разделе 22.8.

## Демон nfsd: обслуживание файлов

После того как демон **mountd** убедился в правильности клиентского запроса на монтирование, клиенту разрешается запрашивать различные операции над файлами. Эти запросы обрабатываются на сервере демоном **nfsd**[5]. Он не должен выполняться на компьютере-клиенте NFS. Единственное исключение — когда клиент сам экспортирует файловые системы.

Демон **nfsd** принимает числовой аргумент, определяющий, сколько экземпляров самого себя нужно породить посредством системного вызова **fork**. Правильный выбор числа процессов очень важен, но, к сожалению, это из области черной магии. Если это число будет слишком мало или слишком велико, производительность сервера NFS может снизиться.

Оптимальное количество выполняющихся демонов **nfsd** зависит от операционной системы и используемого аппаратного обеспечения. Если вы заметили, что команда **ps** обычно показывает количество демонов в состоянии D (непрерываемый сон) и при этом наблюдается простой центрального процессора, попробуйте увеличить количество потоков. Если при добавлении демонов **nfsd** происходит увеличение средней загрузки (о чем сообщает команда **uptime**), значит, вы слишком увлеклись и следует вернуться к прежнему состоянию. Кроме того, следует регулярно выполнять команду **nfsstat** и проверять производительность, которая может зависеть от количества выполняемых демонов **nfsd**. Более подробно демон **nfsd** описан в разделе 18.6.

На интенсивно эксплуатируемом сервере NFS версии 2 или 3 с большим числом UDP-клиентов буферы UDP-сокетов могут переполниться, если запросы начнут поступать в то время, когда все демоны **nfsd** уже задействованы. Число переполнений можно узнать с помощью команды **netstat -s**. Запускайте дополнительные демоны до тех пор, пока число переполнений не упадет до нуля, так как переполнение свидетельствует о серьезной нехватке серверных демонов.

[5] Демон ***nfsd*** всего лишь осуществляет не предусматривающий ответа системный вызов, код которого встроен в ядро.

Для изменения числа запускаемых демонов **nfsd** следует отредактировать соответствующий сценарий в файле конфигурации. Расположение этого файла и доступные параметры зависят от системы. Их примеры приведены в табл. 18.8. Внеся изменения в файл конфигурации демона **nfsd**, необходимо заново запустить службу с помощью сценариев, перечисленных в табл. 18.1.

**Таблица 18.8. Как задать количество выполняемых демонов nfsd**

| Система | Файл конфигурации (в каталоге /etc) | Параметр | По умолчанию |
|---|---|---|---|
| Ubuntu | **default/nfs-kernel-server** | RPCNFSDCOUNT | 8 |
| SUSE | **sysconfig/nfs** | USE_KERNEL_NFCD_NUMBER | 4 |
| Red Hat | **sysconfig/nfs** | RPCNFSDCOUNT | 8 |
| Solaris | **default/nfs** | NFSD_SERVER | 16 |
| HP-UX | **default/nfs** | NFSD_SERVER | 16 |
| AIX | Для изменения количества выполняемых демонов nfsd используется программа SMIT или команда chnfs. | | |

# 18.4. Клиентская часть протокола NFS

Процессы монтирования сетевых и локальных файловых систем во многом схожи. Команда **mount** понимает запись вида *имя_узла*:*каталог* как путь к каталогу, расположенному на указанном компьютере. Этому каталогу будет поставлен в соответствие каталог локальной файловой системы. По завершении монтирования доступ к сетевой файловой системе осуществляется традиционными средствами. Таким образом, команда **mount** и ее NFS-расширения — самое важное для системного администратора на NFS-клиенте.

Для того чтобы файловую систему NFS можно было монтировать, ее необходимо соответствующим образом экспортировать (об этом рассказывалось выше). Клиентская команда **showmount** позволяет клиенту проверить, правильно ли сервер экспортирует файловые системы.

```
$ showmount -e monk
Export list for monk:
/home/ben harp.atrust.com
```

Как следует из этого примера, каталог **/home/ben** на сервере **monk** экспортирован в клиентскую систему `harp.atrast.com`. Если сетевая файловая система по какой-то причине не работает, возможно, вы просто забыли выполнить команду **exportfs -a** после редактирования файла **exports**. Проверьте после этого вывод команды **showmount**.

Если каталог был правильно экспортирован на сервере, а команда **showmount** сообщает об ошибке или возвращает пустой список, внимательно проверьте, все ли необходимые процессы запущены на сервере (**portmap**, **mountd**, **nfsd**, **statd** и **lockd**), разрешен ли в файлах **hosts.allow** и **hosts.deny** доступ к этим демонам и, вообще, та ли это клиентская система.

🕮 Более подробная информация о файлах **hosts.*** и системе TCP Wrapper приведена в разделе 22.8.

Информация о пути, отображаемая командой **showmount**, например **/home/ben**, как в предыдущем случае, является корректной только в версиях протокола NFS 2 и 3. Серверы, работающие в рамках протокола NFS 4, экспортируют целостную и единоо-

бразную псевдофайловую систему. Традиционная концепция протокола NFS, относящаяся к разным точкам монтирования, в версии 4 не работает, поэтому команду **showmount** применять нельзя.

К сожалению, достойной альтернативы команде **showmount** в версии NFS 4 нет. На сервере команда **exportfs -v** демонстрирует существующие экспортированные файловые системы, но она работает только в локальном масштабе. Если у вас нет прямого доступа к серверу, смонтируйте корень псевдофайловой системы сервера и пройдите по структуре каталогов вручную, отметив каждую точку монтирования.

Реальное монтирование файловой системы осуществляется примерно такой командой.

```
$ sudo mount -o rw,hard,intr,bg monk:/home/ben /nfs/ben
```

Для того чтобы сделать то же самое в версии 4 под управлением системы Linux, выполните следующую команду.

```
$ sudo mount -t nfs4 -o rw,hard,intr,bg monk:/ nfs/ben
```

В данном случае указанные после опции **-o** флаги говорят о том, что файловая система монтируется в режиме чтения/записи (**rw**), файловые операции разрешается прерывать (**intr**), а повторные попытки монтирования должны выполняться в фоновом режиме (**bg**). Наиболее распространенные флаги приведены в табл. 18.9.

**Таблица 18.9. Флаги монтирования NFS**

| Флаг | Назначение |
|---|---|
| `rw` | Монтирование файловой системы для чтения/записи (она должна экспортироваться сервером в режиме чтения/записи) |
| `ro` | Монтирование файловой системы только для чтения |
| `bg` | Если смонтировать файловую систему не удается (сервер не отвечает), следует перевести операцию в фоновый режим и продолжить обработку других запросов на монтирование |
| `hard` | Если сервер отключился, операции, которые пытаются получить к нему доступ, блокируются до тех пор, пока сервер не включится вновь |
| `soft` | Если сервер отключился, операции, которые пытаются получить к нему доступ, завершаются выдачей сообщения об ошибке. Этот флаг полезно устанавливать для того, чтобы предотвратить зависание процессов в случае неудачного монтирования не очень важных файловых систем |
| `intr` | Позволяет прерывать с клавиатуры заблокированные операции (будут выдаваться сообщения об ошибке) |
| `nointr` | Не позволяет прерывать с клавиатуры заблокированные операции |
| `retrans=`*n* | Указывает, сколько раз нужно повторить запрос, прежде чем будет выдано сообщение об ошибке (для файловых систем, смонтированных с флагом `soft`) |
| `timeo=`*n* | Задает интервал тайм-аута для запросов (в десятых долях секунды) |
| `rsize=`*n* | Задает размер буфера чтения равным *n* байт |
| `wsize=`*n* | Задает размер буфера записи равным *n* байт |
| `sec=`*режим* | Задает режим безопасности |
| `vers=`*n* [a] | Задает версию протокола NFS |
| `proto =` *протокол* | Выбирает транспортный протокол; им должен быть протокол `tcp` для версии NVS 4 |

[a] Несмотря на то что флаг `vers` указан на справочных страницах систем Linux, посвященных команде mount, использование ее результатов является ошибкой. Для того чтобы задать четвертую версию протокола NFS, следует выполнить команду mount -t nfs4.

Файловые системы, смонтированные с флагом `hard` (установка по умолчанию), могут вызывать зависание процессов при отключении сервера. Это особенно неприятно, когда такими процессами оказываются стандартные демоны. Вот почему не рекомендуется обслуживать ключевые системные программы через NFS. В общем случае использование флагов `soft` и `intr` позволяет сократить число проблем, связанных с NFS. Но эти же флаги могут вызывать нежелательные побочные эффекты (например, останов 20-часового процесса моделирования из-за временного сбоя в сети после 18 часов работы)[6]. Исправить некоторые недостатки монтирования позволяют средства автоматического монтирования, например демон **autofs** (описывается далее).

Размеры буферов чтения и записи применимы в отношении обоих протоколов — TCP и UDP, но оптимальные значения различны. В случае TCP буфер должен быть большим, поскольку данные передаются эффективнее. Хорошее значение — 32 Кбайт. В случае UDP, если сервер и клиент находятся в одной сети, оптимальный размер равен 8 Кбайт. Тем не менее по умолчанию принят размер 1 Кбайт, хотя даже на **man**-странице рекомендуется повысить это значение до 8 Кбайт.

Протестировать точку монтирования можно с помощью команды **df**, как если бы это была обычная локальная файловая система.

```
% df /nfs/ben
Filesystem          1k-blocks     Used Available Use%  Mounted on
ltopard:/home/ben   17212156  1694128 14643692   11%  /nfs/ben
```

Разделы NFS демонтируются командой **umount**. Если сетевая файловая система кем-то используется в момент демонтирования, будет выдано сообщение об ошибке.

```
umount: /nfs/ben: device is busy
```

Найдите с помощью команды **lsof** процессы, у которых есть открытые файлы в этой файловой системе, и уничтожьте эти процессы либо смените каталоги в случае интерпретаторов команд. Если ничего не помогает или сервер отключен, воспользуйтесь командой **umount -f** для принудительного демонтирования.

Стоит повторить сноску в табл.18.9: по умолчанию команда **mount** системы Linux, распознав в командной строке синтаксическую конструкцию *имя_узла:каталог*, выбирает тип файловой системы **nfs**, который является корректным только в версиях 2 и 3. Для того чтобы задать четвертую версию протокола NFS, следует выполнить команду **mount -t nfs4**.

## Монтирование файловых систем NFS на этапе начальной загрузки

С помощью команды **mount** можно создавать лишь временные сетевые точки монтирования. Файловые системы, которые являются частью постоянной конфигурации, должны быть перечислены в файле **/etc/fstab** (**/etc/vfstab** в системе Solaris), для того чтобы они автоматически монтировались на этапе начальной загрузки. С другой

[6] Джефф Форис (Jeff Forys), один из наших рецензентов, заметил по этому поводу: "Большинство сетевых файловых систем должно монтироваться с флагами ***hard***, ***intr*** и ***bg***, поскольку они наилучшим образом соответствуют исходной концепции NFS (надежность и отсутствие информации о состоянии). Флаг ***soft*** — это мерзкий сатанинский трюк! Если пользователь захочет прервать операцию монтирования, пусть он сделает это сам. В противном случае следует дождаться включения сервера, и все в конце концов нормализуется без какой бы то ни было потери данных".

стороны, они могут обрабатываться утилитой автоматического монтирования, например демоном **autofs**.

🕮 Более подробная информация о демоне **autofs** приведена в разделе 18.8.

Следующие элементы файла **fstab** предназначены для монтирования файловых систем **/home** и **/usr/local** компьютеров **monk** и **ross**.

```
# filesystem      mountpoint fstype   flags                          dump fsck
monk:/home        /nfs/home  nfs      rw,bg,intr,hard,nodev,nosuid   0    0
ross:/usr/local   /usr/local nfs4     ro,bg,intr,soft,nodev,nosuid   0    0
```

При добавлении элементов в файл **fstab** обязательно создавайте каталоги точек монтирования с помощью команды **mkdir**. Можно сделать так, чтобы изменения вступили в силу немедленно (без перезагрузки). Для этого следует выполнить команду **mount -a -t nfs**, которая смонтирует все файловые системы типа `nfs`, перечисленные в файле **fstab**.

Формат файла **/etc/vfstab** в системе Solaris немного отличается, но параметры остаются прежними. Параметры протокола NFS в основном являются одинаковыми у всех систем.

Для конфигурирования процесса монтирования системы NFS в процессе загрузки операционной системы AIX используется программа SMIT. Не надо редактировать файл **/etc/filesystems** вручную, потому что он может быть записан заново при импорте или экспорте группового тома.

🕮 Более подробная информация о файле **fstab** приведена в разделе 8.9.

Добавляя записи в файл **/fstab/vfstab**, не забудьте создать соответствующие каталоги в точке монтирования с помощью команды **mkdir**. Эти изменения можно ввести в действие немедленно (без перезагрузки), выполнив команду **mount -a -F nfs** в системах Solaris или HP-UX; в системе Linux вместо флага **-a** необходимо использовать флаг **-F**, а версия 4 монтируется с помощью команды **mount -a -t nfs4**. В системе AIX файловые системы NFS можно смонтировать с помощью команды **mount -v nfs -a**.

В поле `flags` файла **/etc/fstab** задаются параметры точек монтирования NFS. Это те же параметры, которые указываются в командной строке **mount**.

## Ограничения на выбор порта

Клиентам NFS разрешается использовать любой TCP- или UDP-порт для подключения к серверу NFS. Однако некоторые серверы могут требовать, чтобы запросы поступали из привилегированного порта (номер которого меньше 1024). Другие серверы позволяют выбирать порт по своему усмотрению. Впрочем, в мире персональных компьютеров и настольных Linux-систем применение привилегированных портов не приводит к реальному повышению безопасности системы.

Большинство клиентов протокола NFS следуют традиционному (и по-прежнему рекомендуемому) подходу: по умолчанию выбирается привилегированный порт, что позволяет избежать ненужных конфликтов. Для того чтобы разрешить запросы на монтирование, поступающие из непривилегированных портов, в системе Linux можно использовать параметр `insecure`.

## 18.5 Идентифицирующее отображение в протоколе NFS 4

В отличие от ранних версий протокола NFS, в которых пользователи идентифицировались просто по идентификаторам UID и GID, в версии 4 используются строки, имеющие вид `пользователь@nfs_домен` и `группа@nfs_домен`. Как на серверной, так и на клиентской стороне демон идентифицирующего отображения выполняет преобразование строковых идентификаторов в значения локальных идентификаторов UID и GID и наоборот. Отображенные значения используются для трансляции файла, содержащего информацию об атрибутах, но не для управления доступом, которое реализуется отдельно.

Все системы, участвующие в работе системы по протоколу NFSv4, должны иметь одинаковые домены NFS. В большинстве случаев в качестве домена NFS целесообразно использовать свой домен DNS. Например, естественно выбрать `atrust.com` в качестве домена NFS для сервера `harp.atrust.com`. Клиенты в поддоменах (например, `booklab.atrust.com`) могут при желании использовать более короткие имена сайта (например, `atrust.com`) для реализации взаимодействия в рамках протокола NFS.

К сожалению для администраторов, стандартной реализации отображения идентификаторов UID в протоколе NFSv4 не существует. В табл. 18.10 перечислены демоны идентифицирующего отображения в каждой из систем и указано местонахождение их конфигурационного файла.

**Таблица 18.10. Демоны идентифицирующего отображения и их конфигурации**

| Система | Демон | Конфигурация |
|---|---|---|
| Linux | /usr/sbin/rpc.idmapd | /etc/idmapid.conf |
| Solaris | /usr/lib/nfs/nfsmapid | /etc/default/nfs [a] |
| HP-UX | /usr/lib/nfs/nfsmapid | /etc/default/nfs [a] |
| AIX | /usr/sbin/nfsgyd | chnfsdom домен |

[a] Домен задается параметром NFSMAID_DOMAIN.

Кроме указания доменов NFS, для реализации идентифицирующего отображения необходима дополнительная помощь системного администратора. Демон запускается в момент загрузки системы в том же самом сценарии, который управляет системой NFS. После внесения изменения в файл конфигурации необходимо вновь запустить демон. Параметры, определяющие, например, детализацию вывода и альтернативное управление анонимной учетной записью, обычно являются доступными; более подробную информацию следует искать на справочной странице, посвященной демону идентифицирующего отображения.

## 18.6. Команда nfsstat: отображение статистики NFS

Команда **`nfsstat`** отображает различные статистические данные, накапливаемые в системе NFS. Команда **`nfsstat -s`** выдает статистику процессов сервера NFS, а команда **`nfsstat -c`** отображает информацию, касающуюся клиентских операций.

```
% nfsstat -c
Client rpc:
```

```
calls   badcalls   retrans    badxid     timeout    wait newscred timers
62235   1595       0          3          1592       0    0        886

Client nfs:
calls   badcalls   nclget     nslsleep
62213   3          62643      0
null    getattr    setattr    readlink   lookup   root   read
0%      34%        0%         21%        30%      0%     2%
write   wrcache    create     remove     rename   link   symlink
3%      0%         0%         0%         0%       0%     0%
mkdir   readdr     rmdir      fsstat
0%      6%         0%         0%
```

Приведенные результаты получены на нормально функционирующем сервере NFS. Если более 3% вызовов терпят неудачу, то это говорит о наличии проблем на NFS-сервере или в сети. Как правило, причину можно выяснить, проверив значение параметра `badxid`. Если его значение близко к нулю, а количество тайм-аутов больше 3%, то пакеты, поступающие на сервер и отправляемые с него, теряются в сети. Решить эту проблему можно, уменьшив значение параметров монтирования `rsize` и `wsize` (размеры блоков для чтения и записи).

Если значение параметра `badxid` такое же большое, как и значение параметра `timeout`, то сервер отвечает, но слишком медленно. В этом случае необходимо либо заменить сервер, либо увеличить параметр `timeo`.

Регулярное выполнение команд **`nfsstat`** и **`netstat`** и анализ выдаваемой ими информации помогут администратору выявлять возникающие в NFS проблемы раньше, чем с ними столкнутся пользователи.

## 18.7. Специализированные файловые серверы NFS

Быстрый и надежный файловый сервер — один из важнейших элементов вычислительной среды. Конечно, дешевле всего организовать файловый сервер на рабочей станции, воспользовавшись имеющимися жесткими дисками. Однако это не оптимальное решение с точки зрения производительности и удобства администрирования.

Уже много лет на рынке предлагаются специализированные файловые серверы NFS. У них есть ряд преимуществ по сравнению с "кустарными" системами.

- Они оптимизированы на выполнение операций с файлами и, как правило, обеспечивают наилучшую производительность при работе с системой NFS.
- По мере увеличения требований к хранилищу файлов можно легко наращивать мощности сервера, даже если придется обслуживать терабайтовые массивы данных и сотни пользователей.
- Они более надежны, чем обычные системы, благодаря усеченному набору программ, применению дополнительных устройств, обеспечивающих избыточность, и зеркальному дублированию информации на жестких дисках.
- Они обычно реализуют доступ к файлам для клиентов как UNIX, так и Windows, а иногда даже содержат встроенные веб-, FTP- и SFTP-серверы.
- Чаще всего они проще в администрировании, чем файловые серверы UNIX.
- Они обладают улучшенными средствами резервного копирования и контроля состояния, по сравнению с обычными UNIX-системами.

На рынке существующих систем лидируют устройства компании Network Appliance, Inc. (`netapp.com`). Диапазон предложений — от малых до очень крупных систем, и цены вполне приемлемы. Компания EMC занимает нишу устройств высшего класса. Она выпускает хорошие продукты, но их цены могут испугать кого угодно. Компания LeftHand Networks, приобретенная корпорацией HP, является еще одним игроком, который выиграл от появления дешевых, высокопроизводительных и удобных устройств для хранения информации.

Еще одним возможным решением проблем, связанных с эффективным управлением памятью, могут стать сети хранения данных (storage area network — SAN). Они отличаются от специализированных серверов тем, что не имеют представления о файловых системах; они просто обслуживают блоки дисков. Следовательно, сеть SAN не обременена затратами, связанными с операционной системой, и способна осуществлять быстрый доступ для записи и чтения, но не может управлять параллельным доступом нескольких клиентов без помощи кластеризованной файловой системы. Более подробная информация о сетях SAN приведена в разделе 8.11.

## 18.8. Автоматическое монтирование

Индивидуальное монтирование файловых систем посредством упоминания их в файлах **`/etc/fstab`** и **`/etc/vfstab`** сопряжено в крупных сетях с рядом проблем.

Во-первых, ведение файла **`fstab`** на нескольких сотнях компьютеров — весьма утомительная задача. Каждый из компьютеров может отличаться от других и требовать особого подхода.

Во-вторых, если файловые системы монтируются с множества компьютеров, то в случае сбоя всего лишь одного из серверов наступает хаос, так как все команды, пытающиеся получить доступ к точкам монтирования этого сервера, зависают.

В-третьих, сбой какого-нибудь важного сервера может нанести немалый ущерб пользователям, сделав недоступными такие важные разделы, как, например, **`/usr/share/man`**. Самый простой выход из этой ситуации — временно смонтировать копию раздела с резервного сервера. Однако система NFS не имеет инструментов для обеспечения работы резервных серверов.

Решить эту проблему можно с помощью демона автоматического монтирования, который подключает файловые системы, когда к ним выполняется обращение, и отключает их, когда надобность в них отпадает. Этот процесс осуществляется незаметно для пользователей и позволяет свести к минимуму число активных точек монтирования. Демону можно также предоставить список реплицированных (идентичных) файловых систем, чтобы сеть могла функционировать в случае отказа основного сервера.

Для реализации фонового подключения и отключения демон связывает драйвер виртуальной файловой системы с каталогами, обозначенными как точки автоматического монтирования. Раньше он делал это, выступая в качестве сервера NFS, но такой подход имел ряд серьезных ограничений, поэтому редко применяется в современных системах. В настоящее время используется драйвер файловой системы **`autofs`**, встроенный в ядро.

Вместо того чтобы зеркально дублировать в сети реальную файловую систему, демон автоматического монтирования воссоздает ее иерархию в соответствии со спецификациями, указанными в файле конфигурации. Когда пользователь ссылается на каталог виртуальной файловой системы демона, последний перехватывает эту ссылку, монтирует реальную файловую систему, к которой обращается пользователь, и передает ссылку дальше. На системах, поддерживающих демон **`autofs`**, файловая система NFS монтиру-

ется в файловой системе **autofs** в обычном для системы UNIX режиме. Другие системы могут потребовать, чтобы монтирование осуществлялось в отдельном каталоге, на который затем устанавливается символическая ссылка.

Идея автоматического монтирования была предложена компанией Sun, которая в настоящее время стала частью корпорации Oracle. Имеющийся в Linux демон **automount** имитирует работу одноименной утилиты компании Sun, хотя реализован независимо и обладает рядом отличительных особенностей. Аналогично система AIX предоставляет свой собственный демон **automount**, который компания IBM называет "инструментом управления для системы AutoFS".

Разные реализации демона **automount** используют три вида файлов конфигурации, которые называются таблицами (maps): таблицы прямых назначений, таблицы косвенных назначений и главные таблицы.[7] Таблицы прямых и косвенных назначений содержат информацию о файловых системах, подлежащих автоматическому монтированию. Главная таблица перечисляет таблицы прямых и косвенных назначений, которые должен учесть демон **automount**. В каждый момент времени может быть активной только одна главная таблица; по умолчанию главная таблица хранится в каталоге **/etc/auto_master** (**/etc/auto.master** — в системе Linux).

В большинстве систем демон **automount** представляет собой отдельную команду, которая считывает свои файлы конфигурации, настраивает все необходимые для демона **autofs** параметры и прекращает работу. Действительные ссылки на файловые системы, подлежащие автоматическому монтированию, обрабатываются с помощью демона **autofs** другим процессом — демоном **automontd**. Этот демон выполняет свою работу незаметно и не требует дополнительной конфигурации.

В системах Linux этот демон называется **automountd**, а функция настройки выполняется в сценарии загрузки **/etc/init.d/autofs**. Подробная информация об этом приведена ниже. Далее мы будем называть команду настройки именем **automount**, а демон автоматического монтирования — **automountd**.

Если вы изменили главную таблицу или одну из таблиц прямых назначений, на которую она ссылается, то необходимо заново запустить демон **automount**, чтобы эти изменения вступили в силу. При использовании параметра **-v** команда **automount** демонстрирует изменения, внесенные в ее конфигурацию.

Демон **automount** имеет также аргумент **-t**, сообщающий, как долго (в секундах) файловая система, монтируемая автоматически, может оставаться неиспользуемой перед тем, как будет размонтирована. По умолчанию этот параметр равен 10 минутам. Поскольку точка монтирования NFS, принадлежащая сбойному серверу, может вызвать зависание программы, целесообразно удалять автоматически смонтированные файловые системы, которые больше не используются. Кроме того, не следует задавать таймаут слишком долгим.[8]

## Таблицы косвенных назначений

Эти таблицы используются для автоматического монтирования нескольких файловых систем в общем каталоге. Однако путь к каталогу задается в главном файле, а не в самой таблице. Например, таблица назначений для файловых систем, монтируемых

[7] Таблицы прямых назначений также управляют базой данных системы NFS или каталогом LDAP, но это сложно.

[8] С другой стороны, монтирование файловой системы занимает определенное время. Система отвечает быстрее и плавнее, если файловые системы не монтируются непрерывно.

в каталоге **/chimchim** (в соответствии с показанным выше главным файлом), может иметь следующий вид.

```
users   harp:/harp/users
devel   -soft harp:/harp/devel
info    -ro harp:/harp/info
```

В первой колонке указывается имя подкаталога, в котором будет смонтирована файловая система. В следующих колонках перечислены опции монтирования и приведен исходный путь к файловой системе. В данном примере (файл **/etc/auto.harp**) демону **automount** сообщается о том, что он может монтировать каталоги **/harp/users**, **/harp/devel** и **/harp/info** с компьютера **harp**, при этом каталог **info** монтируется только для чтения, а каталог **devel** — в режиме **soft**.

В рассматриваемой конфигурации подкаталоги на компьютере **chimchim** и на локальном компьютере будут идентичными, хотя это вовсе не обязательно.

## Таблицы прямых назначений

Рассматриваемые таблицы перечисляют файловые системы, не имеющие общего префикса, такие как **/usr/src** и **/cs/tools**. Таблица прямых назначений (например, **/etc/auto.direct**), указывающая, что обе эти файловые системы должны монтироваться автоматически, может выглядеть следующим образом.

```
/usr/src   harp:/usr/src
/cs/tools  -ro monk:/cs/tools
```

Поскольку у этих файловых систем нет общего родительского каталога, их монтирование должно реализовываться с помощью разных демонов **autofs**. Эта конфигурация требует дополнительных расходов, но ее преимуществом является то, что точка монтирования и структура каталогов всегда остаются доступными таким командам, как **ls**. Применяя команду **ls** к каталогу, содержащему таблицы косвенных назначений, пользователи могут запутаться, потому что демон **automount** не показывает подкаталоги, пока не получит доступ к их содержимому (команда **ls** не заглядывает внутрь каталогов, смонтированных автоматически, поэтому она не может вызвать их монтирование).

## Главные таблицы

Главный файл содержит список таблиц прямых и косвенных назначений, которые должен учитывать демон **automount**. Для каждой таблицы косвенных назначений указывается корневой каталог, используемый для монтирования, определенного в таблице.

Главная таблица, использующая таблицы прямых и косвенных назначений, упомянутых в предыдущих примерах, может выглядеть следующим образом.

```
# Directory     Map
/harp            /etc/auto.harp -proto=tcp
/-               /etc/auto.direct
```

В первом столбце указывается имя локального каталога для таблицы косвенных назначений или специальный токен `/-` для таблицы прямых назначений. Во втором столбце указывается файл, в котором должна храниться таблица. Может существовать несколько таблиц разного типа. Если параметр монтирования указывается в конце строки, то он по умолчанию применяется ко всем точкам монтирования, указанным в таблице. Администраторы системы Linux всегда должны указывать флаг монтирования `-fstype=nfs4` для серверов, использующих четвертую версию протокола NFS.

В большинстве систем параметры, заданные по умолчанию для элементов главной таблицы, не смешиваются с параметрами, заданными для таблиц прямых и косвенных назначений. Если элемент главной таблицы имеет свой собственный список параметров, то значения, заданные по умолчанию, игнорируются. Тем не менее в системе Linux эти наборы параметров смешиваются. Если один и тот же параметр встречается в двух местах, то значение элемента главной таблицы заменяет значение, заданное по умолчанию.

Главная таблица обычно заменяется или дополняется версией, используемой совместно с сетевой службой NIS. Подробности можно найти в документации.

## Исполняемые таблицы

Если файл, содержащий таблицу косвенных назначений, является исполняемым, то он считается сценарием (или программой), динамически генерирующим информацию об автоматическом монтировании. Демон не читает таблицу в текстовом виде, а запускает файл на выполнение, передавая ему аргумент ("ключ"), где указывается, к какому подкаталогу пользователь пытается получить доступ. Сценарий отвечает за вывод соответствующей записи таблицы. Если переданный ключ неверен, сценарий завершается, ничего не отображая.

Такая методика очень удобна и позволяет компенсировать многочисленные недостатки довольно странной системы конфигурирования демона **automount**. По сути, она дает возможность создать единый для всей организации конфигурационный файл произвольного формата. Можно написать несложный сценарий, декодирующий глобальные конфигурационные параметры на каждом компьютере. В состав некоторых систем входит удобный сценарий **/etc/auto.net**, принимающий в качестве ключа имя компьютера и монтирующий все экспортируемые файловые системы этого компьютера.

## Видимость демона automount

Когда вы перечисляете содержимое родительского каталога файловой системы, монтируемой автоматически, этот каталог оказывается пустым независимо от того, сколько там было автоматически смонтировано файловых систем. Файловые системы, смонтированные автоматически, невозможно просмотреть с помощью пользовательского браузера.

Рассмотрим пример.

```
$ ls /portal
$ ls /portal/photos
art_class_2010     florissant_1003          rmnp03
blizzard2008       frozen_dead_guy_Oct2009  rmnp_030806
boston021130       greenville.021129        streamboat2006
```

Файловая система **photos** прекрасно работает и автоматически монтируется в каталоге **/portal**. Для того чтобы получить к ней доступ, необходимо указать ее полное имя. Однако обзор каталога **/portal** не указывает на ее существование. Если бы вы смонтировали эту систему с помощью команды **fstab** или **mount**, то она ничем бы не отличалась от других каталогов и была бы видимой в родительском каталоге.

Для того чтобы увидеть файловые системы, смонтированные автоматически, используются символические ссылки на точки автоматического монтирования. Например, если **/automounts/photos** — это ссылка на каталог **/portal/photos**, то команда **ls** позволяет увидеть среди содержимого каталога **/automounts**, что каталог **photos** был

смонтирован автоматически. Ссылки на каталог **/authomounts/photos** по-прежнему проходят через механизм автоматического монтирования и работают правильно.

К сожалению, эти символические ссылки требуют сопровождения и могут потерять согласованность с реальными точками автоматического монтирования, если они периодически не обновляются с помощью некоего сценария.

## Реплицированные файловые системы и демон automount

В некоторых случаях файловая система, предназначенная только для чтения, например **/usr/share**, может оказаться идентичной на нескольких серверах одновременно. В этом случае мы можем сообщить демону **automount** о нескольких потенциальных источниках этой файловой системы. Демон самостоятельно выберет источник файловой системы, исходя из того, какой сервер окажется ближайшим по номеру в данной сети, по версии протокола NFS, а также по времени ответа на первоначальный запрос.

Несмотря на то что демон **automount** не видит файловую систему и не следит за ее использованием, реплицированные файловые системы должны предназначаться только для чтения (**/usr/share** или **/usr/local/X11**). Демон **automount** не имеет возможности синхронизировать записи между серверами, поэтому реплицированные файловые системы, предназначенные только для чтения, имеют небольшое практические значение.

В системах Solaris и HP-UX при возникновении проблем демон **automount** может просто переключаться с одного сервера, имеющего реплицированную файловую систему, на другой. Эта функциональная возможность поддерживается исключительно для файловых систем, предназначенных только для чтения, но ходят слухи, что на системы чтения-записи она распространяется больше, чем об этом написано в документации. Однако если демон **automount** изменяет сервер, ссылки на файлы, открытые для записи, становятся некорректными, что лишний раз свидетельствует о том, что использовать реплицированные файловые системы для чтения и записи не рекомендуется.

Несмотря на то что демон **automount** способен выбирать сервер с реплицированной файловой системой, руководствуясь собственными критериями эффективности и близости, при желании вы можете назначить свои приоритеты. Приоритет задается небольшим целым числом, причем чем больше число, тем ниже приоритет. Наиболее подходящим является приоритет по умолчанию, который равен 0.

Файл **auto.direct**, определяющий файловые системы **/usr/man** и **/cs/tools** как реплицированные, может выглядеть следующим образом.

```
/usr/man   -ro harp:/usr/share/man monk(1):/usr/man
/cs/tools  -ro leopard,monk:/cs/tools
```

Обратите внимание на то, что имена серверов могут перечисляться вместе, если пути к их источникам совпадают. Значение `(1)` после имени `monk` в первой строке задает приоритет сервера по отношению к файловой системе **/usr/man**. Отсутствие такого значения после имени `harp` означает, что этот сервер имеет неявный приоритет, равный нулю.

## Автоматическое монтирование (V3; все, кроме Linux)

Вместо перечисления всех возможных монтируемых файловых систем в таблице косвенных или прямых назначений, вы можете сообщить демону **automount** немного информации о правилах именования и позволить ему разбираться в них самостоятельно. Главным обстоятельством при этом является тот факт, что демон **mountd**, применяемый

к удаленному серверу, может посылать запросы о том, какие файловые системы были смонтированы на этом сервере. В четвертой версии протокола NFS экспорт всегда обозначается символом /, что исключает необходимость в такой функциональной возможности.

Существует несколько способов "автоматического конфигурирования автоматического монтирования". Самый простой из них относится к типу монтирования `-hosts`. Если указать флаг `-hosts` в качестве имени таблицы в файле главной таблицы, то демон **automount** отобразит файловые системы, экспортированные на указанные узлы, в заданный каталог автоматического монтирования.

```
/net  -hosts  -nosuid,soft
```

Например, если сервер `harp` экспортировал файловую систему **/usr/share/man**, то каталог должен стать доступным для демона автоматического монтирования с помощью пути **/net/harp/usr/share/man**.

Реализация флага `-hosts` не подразумевает перечисления всех возможных узлов, с которых можно экспортировать монтируемую файловую систему; это просто практически невозможно. Вместо этого команда ждет ссылки на имена конкретных подкаталогов, а затем монтирует экспортированные системы с указанного узла.

Аналогичного, но более тонкого эффекта можно достичь с помощью шаблонных символов "*" и "&" в таблице косвенных назначений. Кроме того, существует множество макросов, используемых вместе с таблицами и именем текущего узла, типом архитектуры и т.д. Подробности можно найти на справочной странице **automount**(1M).

## Специфика системы Linux

Реализация демона **automount** в системе Linux немного отличается от реализации компании Sun. Основные отличия касаются имен команд и файлов.

В системе Linux **automount** — это демон, который на самом деле монтирует и демонтирует удаленные файловые системы. Он выполняет работу, которую демон **automount** делает в других системах, и обычно не требует запуска вручную.

По умолчанию главная таблица записана в файле **/etc/auto.master**. Ее формат и формат таблиц косвенных назначений описаны выше. Однако документацию о них найти трудно. Формат главной таблицы описан на странице **auto.master**(5), а формат таблицы косвенных назначений — на странице **autofs**(5). Будьте осторожны, не перепутайте их со страницей **autofs**(8), которая документирует команду **autofs**. (Как сказано на одной из справочных страниц: "Документация оставляет желать лучшего.") Для того чтобы изменения вступили в силу, выполните команду **/etc/init.d/autofs/ reload**, являющуюся эквивалентом команды из системы Solaris.

Реализация системы Linux не поддерживает флаг `-hosts` для "автоматического конфигурирования автоматического монтирования".

# 18.9. Рекомендуемая литература


- Callaghan, Brent. *NFS Illustrated.* Addison-Wesley, 1999.
- Stern, Hal, Mike Eisler and Ricardo Labiaga. *Managing NFS and NIS, Second Edition.* Sebastopol: O'Reilly & Associates, 2001.


В табл. 18.11 перечислены документы RFC, посвященные протоколу NFS и его расширениям.

**Таблица 18.11. Документы RFC, связанные с NFS**

| RFC | Название | Автор | Дата |
|---|---|---|---|
| 1094 | Network File System Protocol Specification | Sun Mircosystems | Май 1989 |
| 1813 | NFS Version 3 Protocol Specification | B. Callaghan et al. | Июнь 1995 |
| 2623 | NFS Version 2 and Version 3 Security Issues | M.Eisler | Июнь 1999 |
| 2624 | NFS Version 4 Design Considerations | S.Shepler | Июнь 1999 |
| 3530 | NFS Version 4 Protocol | S.Shepler et al. | Апрель 2003 |

# 18.10. Упражнения

18.1. ☆ Проанализируйте локальную конфигурацию NFS в своей системе. Используется ли NFS или применяется какое-то другое решение? Используется ли автоматическое монтирование? На какие компромиссы пришлось пойти, создавая такую конфигурацию?

18.2. ☆ Какова взаимосвязь между демонами **mountd**, **nfsd** и **portmap** в версиях NFS 2 и 3? Как зависимость от демона **portmap** сказывается на безопасности системы?

18.3. ☆☆ Какие принципиальные различия существуют между версиями NFS 3 и 4? Как состояние или его отсутствие влияет на другие атрибуты протокола?

18.4. ☆☆ Ваш начальник хочет экспортировать вам свои каталоги **/usr** и **/usr/local** через систему NFS. Ответьте на следующие вопросы:

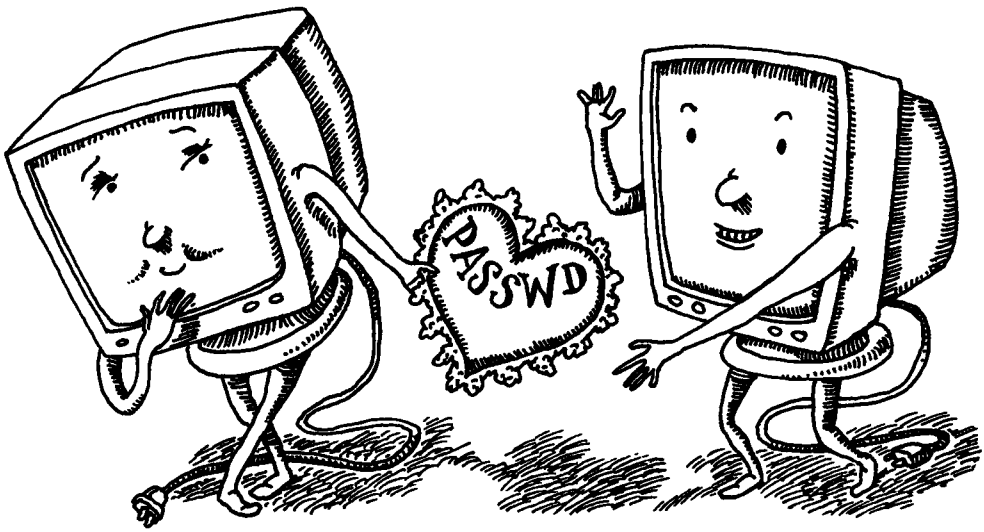
а) предположим, вследствие существующих ограничений вы хотите, чтобы служащие только вашего отдела (подсеть 192.168.123.0/24) могли работать с этими файловыми системами. Какие строки необходимо добавить и в какие файлы, чтобы такая конфигурация стала возможной? Обратите внимание на выбор опций экспортирования;

б) какие действия следует предпринять для того, чтобы демоны **mountd** и **nfsd** смогли распознать эти новые файловые системы? Как проверить факт совместного использования каталогов, не монтируя их?

в) какой стратегии следует придерживаться, чтобы все компьютеры локальной подсети автоматически монтировали экспортируемые каталоги с использованием точек монтирования **/mnt/usr** и **/mnt/usr/local**?

# Глава 19

# Совместное использование системных файлов




Мы знакомы с идеей совместного использования данных разными компьютерами: посредством вложений в сообщения электронной почты, протоколов передачи данных (HTTP и FTP) или служб коллективного доступа к файлам, предоставляемым файловыми системами NFS и CIFS. Эти механизмы позволяют и пользователям совместно использовать файлы и данные приложений. Однако системы UNIX и Linux могут извлечь пользу еще из одного вида коллективного использования ресурсов, а именно распределения административных данных конфигурации. В этом случае имеет место централизация административного управления ресурсами между системами и поддержка согласованности данных.

Регистрационные имена и пароли пользователей — это реальный пример необходимости этого вида коллективного использования ресурсов. Вам редко придется добавлять пользователя на один компьютер; в большинстве случаев вы будете определять этого пользователя в целый класс или сеть компьютеров. Кроме того, большинство организаций в настоящее время сталкивается с необходимостью поддержки платформ смешанного типа — UNIX, Linux и Windows. Разумеется, пользователи испытывают неудобства, поскольку им приходится привыкать к тому, что на компьютерах разных платформ им нужно вводить разные пароли (которые необходимо запоминать и иногда менять). К счастью, синхронизация конфигурации и сведений о пользователях в разных системах (например, в Linux и Windows) — это тривиальная задача.

Совместное использование системных файлов не так просто реализовать, как может показаться. Попытки разработать распределенные административные базы данных для больших сетей продолжались несколько десятилетий, в результате чего были созданы интересные системы. К сожалению, ни одна из этих систем сегодня не кажется нам без-

упречной по идеологии. Некоторые из них просты, но не отличаются достаточной степенью безопасности и масштабирования. Другие обладают хорошими функциональными возможностями, но являются громоздкими. Всем этим системам присущи ограничения, которые могут помешать организовать и настроить сеть так, как нужно администратору, и ни одна из систем не позволяет управлять всей информацией, которую требуется совместно использовать на разных компьютерах.

В этой главе сначала будет рассмотрен ряд базовых методик сохранения файлов конфигурации, синхронизированных по сети. Затем поговорим об облегченном (упрощенном) протоколе доступа к сетевым каталогам (Lightweight Directory Access Protocol — LDAP) — более сложной платформонезависимой системе управления базой данных, которая уже де-факто стала стандартом как в мире UNIX, так и Windows. Сегодня во многих организациях постепенно переходят на использование протокола LDAP. К этому их подталкивает, прежде всего, принятие фирмой Microsoft стандарта LDAP в ее продукте Active Directory, а также желание добиться более качественной интеграции сред Linux и Windows. Наконец, мы поговорим о довольно популярной прежде административной СУБД — NIS, которая остается в рабочем состоянии в некоторых средах, но не устанавливается в новых сетях.

Обратите внимание на то, что совместное использование системных файлов отличается от конфигурации систем и распространения программного обеспечения. Эти области имеют различные цели и потребности и на практике реализуются разными методами.

# 19.1. Предмет совместного использования

В системах Linux и UNIX существует много файлов конфигурации, но далеко не все из них имеет смысл совместно использовать на нескольких компьютерах. В настоящее время совместному использованию подлежат, например, файлы, которые содержат пароли, узлы и псевдонимы. Наиболее распространенные файлы коллективного доступа перечислены в табл. 19.1.

**Таблица 19.1. Системные файлы, которые часто являются объектами совместного использования**

| Имя файла | Назначение |
|---|---|
| `/etc/passwd` | База данных с информацией о пользовательских учетных записях |
| `/etc/shadow`[a] | Файл паролей, соответствующих учетным записям пользователей |
| `/etc/group` | Определения UNIX-групп |
| `/etc/hosts` | Соответствия между именами компьютеров и их IP-адресами |
| `/etc/mail/aliases` | Псевдонимы электронной почты |
| `/etc/sudoers` | Полномочия для команды `sudo` |
| `/etc/skel/a` | Стандартные файлы конфигурации для новых домашних каталогов |

[a] Совместное использование может не поддерживаться другими версиями UNIX из-за несовпадения механизмов шифрования (см. раздел 7.1).

Содержимое табл. 19.1 не претендует на "звание" исчерпывающего списка; точная конфигурация зависит от того, какую степень подобия вы хотите реализовать для своих компьютеров. По большей части, дополнительные файлы конфигурации имеют отношение к конкретным приложениям и не поддерживаются такими административными СУБД, как LDAP, — в этом случае совместное использование файлов достигается их копированием.

О модулях PAM речь пойдет в конце раздела 22.5.

Доступ к файлам, перечисленным в табл. 19.1, традиционно осуществляется через функции стандартной библиотеки языка C. Например, поиск в файле **/etc/passwd** выполняют функции **getpwuid**, **getpwnam** и **getpwent**. Они берут на себя открытие, чтение и синтаксический анализ файла **passwd**, освобождая от этой задачи программы пользовательского уровня. В современных системах используются также модули PAM (Pluggable Authentication Module — подключаемый модуль аутентификации), которые определяют стандартный программный интерфейс безопасного поиска информации. Эти модули позволяют легко интегрировать в Linux и UNIX различные системы, в частности Kerberos и LDAP. Точный порядок поиска данных устанавливается системным администратором (подробнее об этом — в разделе 19.5).

# 19.2. Копирование файлов

Простое копирование файлов не является идеальным решением, но оно применимо на всех типах компьютеров и отличается простотой настройки и сопровождения. Оно также надежно, поскольку число перекрестных зависимостей между компьютерами сводится к минимуму (правда, при этом легче нарушить синхронизацию систем). Благодаря этому мы всегда вольны решать, что распространять по сети и как именно. У нас появляется возможность своевременно обновлять не только системные файлы, но и приложения вместе с файлами данных.

Довольно большое число конфигурационных файлов не поддерживается ни одной из административных СУБД (среди них файл **/etc/ntp.conf**, задающий правила сетевой синхронизации часов). Чтобы поддерживать согласованность этих файлов, не остается другого выхода, кроме как копировать их.

## Использование сервера NFS

Некоторые системы распространяют файлы конфигурации путем опубликования их на сервере NFS. Это, пожалуй, самый простой метод с точки зрения автоматизации — все, что вам нужно на стороне клиента, сосредоточено в **cp** (по крайней мере, теоретически).

Об NFS больше информации можно найти в главе 18.

Раньше протокол сетевого доступа к файловым системам NFS "грешил" проблемами безопасности, что делало его использование несколько рискованным, но в версии NFSv4 основные проблемы были решены. Для улучшения безопасности и защиты от любопытных глаз можно прибегнуть к шифрованию важных файлов.

Подробнее о пакете PGP написано в разделе 22.10.

Следующий шаг на пути повышения безопасности состоит в использовании цифровой подписи применительно к файлам конфигурации с помощью такого пакета шифрования открытых ключей, как PGP (Pretty Good Privacy — вполне хорошая секретность). Клиенты могут удостовериться, что файлы, полученные ими через систему NFS, подлинны и не были модифицированы. Причем это важно сделать еще до их инсталляции.

Многие пакеты программ позволяют указать нестандартное местоположение файлов конфигурации. Следовательно, теоретически можно указать эти пакеты в файлах конфигурации, которые "прописаны" в файловой системе NFS, что избавило бы от необходимости создавать локальные копии. Однако мы настоятельно рекомендуем не

использовать такую схему конфигурации, поскольку она делает каждую систему в мире зависимой от одного сервера NFS, и этот сервер тогда будет вынужден обслуживать всех *таких* клиентов. Хуже того, во многих пакетах не предусмотрена ситуация, когда удаленные системы будут блокировать свои файлы конфигурации или создавать временные файлы в конфигурационных каталогах (и их настройка может даже не сработать корректно). При этом вполне вероятно, что ваша система, прекрасно работая в течение долгого времени, может вдруг напрочь отказать, причем по совершенно непонятной причине, не оставив никаких "улик", по которым бы можно было понять, что произошло. Хотя, если вы любите сюрпризы, добро пожаловать в ад!

## Сравнение модели принудительной рассылки с моделью рассылки по запросу

Отказавшись от модели совместно используемых файловых систем, вы можете рассмотреть и другие варианты. Системы копирования файлов могут работать по *модели принудительной рассылки* (push system) либо *модели рассылки по запросу* (pull system). В первом случае главный сервер периодически отправляет самые свежие версии файлов каждому клиенту независимо от желания последнего. Файлы могут копироваться явно при каждом изменении или просто регулярно высылаться согласно графику (при этом, вероятно, одни файлы будут обновляться чаще других).

Преимущество модели принудительной рассылки заключается в том, что система распространения работает централизованно, на одном компьютере. Файлы, списки клиентов, сценарии обновления и расписания хранятся в одном месте, что делает эту схему простой в управлении. Есть у нее и недостаток: каждый клиент должен позволять главному серверу модифицировать свои системные файлы, что не всегда приемлемо с точки зрения безопасности.

В модели рассылки по запросу каждый клиент отвечает за обновление самого себя с помощью данных, запрашиваемых с сервера. Это менее централизованный способ распространения файлов, зато более гибкий и безопасный. Данная схема копирования особенно привлекательна, когда совместно используемые данные находятся в разных административных областях, поскольку вовсе не обязательно, чтобы главный сервер и компьютер-клиент располагались в одной области.

## Утилита rdist: принудительная рассылка файлов

Проще всего распространять файлы с центрального сервера с помощью утилиты **rdist**. Она немного напоминает программу **make**: вначале пользователь в текстовом редакторе составляет описание (спецификацию) файлов, подлежащих рассылке, а затем утилита **rdist** осуществляет операции копирования в соответствии с описанием. Утилита копирует файлы только в том случае, если их копии устарели, поэтому в спецификации можно задать копирование всех файлов, а утилита **rdist** самостоятельно определит, какие файлы и когда следует копировать.

Утилита **rdist** сохраняет информацию о владельце, группе, правах доступа и времени модификации файла. Когда утилита обновляет существующий файл, то перед инсталляцией новой версии она удаляет старую. Это позволяет использовать данную утилиту для пересылки исполняемых файлов, запущенных во время обновления[1]. Традиционно

[1] Хотя старая версия удаляется из пространства имен файловой системы, она продолжает существовать до тех пор, пока не освобождены все ссылки на нее. Об этом следует помнить при работе с журнальными файлами.

утилита **rdist** работала поверх команды **rsh** и пользовалась ее средствами аутентификации для получения доступа к удаленным системам. К сожалению, она имела ряд слабых мест с точки зрения безопасности и по умолчанию была отключена во многих современных операционных системах. Несмотря даже на то, что в документации к **rdist** до сих пор упоминается **rsh**, не нужно думать, что **rsh** является приемлемым вариантом.

Современные версии утилиты **rdist** примечательны тем, что позволяют заменять команду **rsh** любой другой, понимающей аналогичный синтаксис. Чаще всего это команда **ssh**, которая использует шифрование с открытым ключом для аутентификации узлов и, кроме того, шифрует весь диалог, не давая возможности злоумышленникам, прослушивающим сеть, получать копии системных файлов. Недостаток заключается в том, что удаленные серверы **ssh** должны работать в режиме, позволяющем не указывать пароль (хотя клиента нужно аутентифицировать с помощью пары криптографических ключей), а это менее безопасный режим, чем требуется в обычных условиях. Но все равно это значительный шаг вперед по сравнению с **rsh**. Подробнее о демоне **sshd** и его режимах аутентификации рассказывается в разделе 22.10.

Итак, разобравшись с тем, какую угрозу таит в себе утилита **rdist**, рассмотрим в деталях, как она функционирует. Подобно программе **make**, утилита **rdist** ищет в текущем каталоге управляющий файл (**Distfile** или **distfile**). Команда **rdist -f** задает имя управляющего файла явно. В качестве разделителей в этом файле используются знаки табуляции, пробелы и символы новой строки. Комментарии предваряются знаком #.

Тело управляющего файла состоит из инструкций следующего вида.

```
метка: имена_файлов -> адресаты команды
```

Поле *метка* является именем инструкции. В командной строке можно ввести команду **rdist** *метка*, которая обеспечит рассылку только файлов, упомянутых в указанной инструкции.

Поля *имена_файлов* и *адресаты* — это список файлов, подлежащих копированию, и список компьютеров, куда их нужно переслать соответственно. Если в списке больше одного элемента, нужно заключить его в круглые скобки, а элементы разделить пробелами. Список имен файлов может включать метасимволы, допустимые в интерпретаторе команд (например, **/usr/man/man[123]** или **/usr/lib/***). Допустима также запись *~пользователь*, но соответствующие ей значения на узле-отправителе и узле-адресате могут не совпадать.

По умолчанию утилита **rdist** копирует файлы и каталоги, перечисленные в списке имен, в эквивалентные каталоги на каждом узле-адресате. Этот порядок можно изменить, указав последовательность команд. Каждую команду следует завершать точкой с запятой.

Поддерживаются такие команды.

```
install опции [целевой_каталог];
notify список_адресов;
except список_имен;
except_pat список_шаблонов;
special [список_имен] строка;
cmdspecial [список_имен] строка;
```

Команда **install** задает опции, влияющие на то, как утилита **rdist** копирует файлы. Опции, как правило, используются для обработки символических ссылок, проверки правильности применяемого алгоритма сравнения файлов, а также для контроля над процедурой обработки файлов, отсутствующих в исходном дереве каталогов. Опции,

перед которыми должен стоять флаг **-o**, включают список разделенных запятыми имен. Например, строка

```
install -oremove,follow;
```

заставляет утилиту **rdist** отслеживать символические ссылки (а не копировать их как обычные файлы) и удалять на целевом компьютере файлы, для которых не найдено соответствие на исходном компьютере. Полный список опций приведен на man-странице утилиты **rdist**. В большинстве случаев установки по умолчанию вполне подходят.

Название команды `install` слегка вводит в заблуждение, поскольку файлы копируются независимо от того, присутствует эта команда или нет. Опции задаются так, как они задавались бы в командной строке утилиты **rdist**, но при включении в управляющий файл **Distfile** они действуют только на совокупность файлов, указанную в инструкции.

Необязательный аргумент *целевой_каталог* задает каталог, в который инсталлируются файлы на компьютерах-адресатах. По умолчанию утилита **rdist** использует исходные имена каталогов.

В качестве аргумента команды `notify` задается список адресов электронной почты. При каждом обновлении очередного файла утилита **rdist** посылает по этим адресам почтовое сообщение. Ко всем адресам, не содержащим знак **@**, добавляется имя узла-адресата. Например, при выдаче списка файлов, откорректированных на компьютере **anchor**, запись "pete" превратится в "pete@anchor".

🕮 Больше о регулярных выражениях можно прочитать в разделе 2.3.

Команды **except** и **except_pat** предназначены для удаления имен из списка файлов, подлежащих копированию. Аргументы команды **except** трактуются буквально, аргументы команды **except_pat** интерпретируются как регулярные выражения. Эти команды весьма полезны, поскольку утилита **rdist**, как и **make**, позволяет задавать макроконстанты в начале управляющего файла. Благодаря этому появляется возможность использовать один список файлов в нескольких инструкциях, указывая для каждого компьютера только добавления или удаления.

Команда **special** позволяет выполнить команду интерпретатора **sh** (аргумент *строка*, который следует брать в кавычки) на всех удаленных компьютерах. Если будет задан аргумент *список_имен*, утилита **rdist** выполнит команду после копирования каждого из указанных файлов. Без этого списка **rdist** выполнит команду после каждого файла. Команда **cmdspecial** работает подобным образом, но только выполняет команду интерпретатора **sh** после завершения каждой операции копирования. (Содержимое аргумента *список_имен* передается интерпретатору **sh** в виде переменной окружения.)

Приведем пример файла **Distfile**.

```
SYS_FILES = (/etc/passwd /etc/group /etc/mail/aliases)
GET_ALL = (chimchim lollipop barkadon)
GET_SOME = (whammo spiff)

all: ${SYS_FILES} -> ${GET_ALL}
   notify barb;
   special /etc/mail/aliases "/usr/bin/newaliases";

some: ${SYS_FILES} -> ${GET_SOME}
   except /etc/mail/aliases;
   notify eddie@spiff;
```

🕮 Подробнее о команде **newaliases** рассказывается в конце раздела 20.5.

В данной конфигурации три указанных системных файла копируются на компьютеры chimchim, lollipop и barkadon, а по адресу barb@*адресат* посылается сообщение с описанием всех изменений и замеченных ошибок. После копирования файла **/etc/mail/aliases** утилита **rdist** выполняет на каждом компьютере команду **newaliases**. На компьютеры **whammo** и **spiff** копируются только два файла, а отчет посылается по адресу **eddie@spiff**. Команда **newaliases** на этих двух компьютерах не выполняется.

Для того чтобы утилита **rdist** могла работать, нужно, чтобы демон **sshd** на принимающих узлах доверял узлу, осуществляющему рассылку файлов. Для этого на сервере нужно сгенерировать текстовый ключ и сохранить копию его открытой части в файле **~root/.ssh/authorized_keys** на каждом клиенте. Имеет также смысл ограничить сферу применения ключа и количество узлов, от которых он может приниматься. Более подробно об этом речь пойдет в разделе 22.10 при описании альтернативного метода.

## Утилита rsync: более безопасная рассылка файлов

🕮 Утилита **rsync** доступна по адресу: **rsync.samba.org**.

Утилита **rsync**, написанная Эндрю Триджеллом (Andrew Tridgell) и Полом Маккерасом (Paul Mackerras), подобна утилите **rdist**, но иначе реализована. Она не использует управляющий файл (хотя у сервера все же есть свой конфигурационный файл) и напоминает улучшенную версию команды **rcp**, пытающуюся сохранить ссылки, время модификации и права доступа. Утилита **rsync** более эффективна, чем **rdist**, поскольку анализирует отдельные файлы и пытается передавать только изменения между версиями.

С нашей точки зрения, основным преимуществом утилиты **rsync** является то, что на принимающих компьютерах серверный процесс может запускаться из демона **xinetd** или **inetd**. Сервер (на самом деле это та же утилита **rsync**, но работающая в другом режиме; она должна быть инсталлирована как на главном компьютере, так и на клиентах) допускает гибкую настройку: он может предоставлять удаленный доступ лишь к заданному набору каталогов и требовать, чтобы главный компьютер подтвердил свою подлинность паролем. Поскольку доступ на уровне команды **ssh** не требуется, можно организовать распространение системных файлов посредством утилиты **rsync**, не жертвуя безопасностью. (Тем не менее утилита **rsync** разрешает пользоваться командой **ssh**, а не серверным процессом, работающим под управлением демона **inetd**.) Более того, утилита способна работать в режиме рассылки по запросу (самостоятельно запрашивая файлы у сервера **rsync** и не дожидаясь принудительной рассылки в локальную систему), что делает ее еще более безопасной.

К сожалению, утилита **rsync** в целом не столь гибкая в настройке, как **rdist**, и ее конфигурационный файл не такой сложный, как файл **distfile**. На стороне клиента нельзя выполнить произвольную команду и нельзя рассылать файлы нескольким узлам одновременно.

Например, команда

```
# rsync -gopt --password-file=/etc/rsync.pwd /etc/passwd
      lollipop::sysfiles
```

посылает файл **/etc/passwd** на компьютер **lollipop**. Флаг **-gopt** указывает на сохранение прав доступа, идентификаторов владельцев и времени модификации файла. Два двоеточия в выражении **lollipop::sysfiles** заставляют утилиту связаться с уда-

ленным сервером **rsync** непосредственно через порт 873, а не использовать для этого команду **ssh**. Для аутентификации соединения берется пароль из файла **/etc/rsync.pwd**[2].

В этом примере передается только один файл, хотя утилита **rsync** способна обрабатывать одновременно группу файлов. Кроме того, флаги **--include** и **--exclude** позволяют указать список регулярных выражений, с которым будут сравниваться имена файлов. Благодаря этому можно сформировать довольно сложный набор правил копирования. Если командная строка становится слишком громоздкой, поместите регулярные выражения в отдельные файлы. Для этого существуют опции **--include-file** и **--exclude-file**.

Linux-пакеты **rsync** обычно обеспечивают конфигурацию **xinetd** этой утилиты. Для того чтобы активировать работу сервера, вам нужно будет отредактировать файл **/etc/xinetd.rsync**, поменяв строку **disable=yes** на **disable=no**.

На момент написания этих строк утилита **rsync** не поставляется как часть дистрибутива Solaris. Вы можете загрузить ее исходный код с адреса **rsync.samba.org** и инсталлировать ее либо отыскать с помощью Google исполняемый файл-"полуфабрикат" Solaris. Возможно, для преобразования метода запуска демона в форму, совместимую с новой SMF-оболочкой Solaris, вам придется использовать утилиту **inetconv**.

Система HP-UX также не включает утилиту **rsync**, но вы можете получить предварительно скомпилированные исполняемые файлы HP-UX в depot-форме **swinstall** по адресу:

**hpux.connect.org.uk/hppd/hpux/Networking/Admin/rsync-3.0.6**

Доступ к AIX-версии утилиты **rsync** можно получить с адреса:

**ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/RPMS/ppc/rsync**

На момент написания этих строк пакет RPM для AIX 6.1 находится все еще в стадии разработки. Между тем многим удалось добиться успеха в использовании пакета AIX 5.3 RPM в своих системах AIX 6.1.

После того как утилита **rsync** активизирована, следует модифицировать ряд конфигурационных файлов для настройки сервера **rsync**. Основной из них — это **/etc/rsyncd.conf**, содержащий глобальные конфигурационные параметры и описание набора "модулей", каждый из которых представляет собой дерево каталогов для экспорта или импорта. Разумная конфигурация модуля, в который можно копировать файлы (т.е. который будет принимать запросы на пересылку файлов от подключенного клиента), выглядит примерно так.

```
# Название "sysfiles" для данного модуля выбрано произвольно.
[sysfiles]
# Это каталог, в который будут помещаться рассылаемые файлы.
# Может быть просто '/'.
path = /etc
# Это файл, содержащий пары значений имя_пользователя/пароль
# для аутентификации модуля.
```

[2] Пароль посылается по сети в зашифрованном виде, но сами передаваемые файлы не шифруются. С другой стороны, если в качестве транспортного средства используется команда ***ssh*** (***rsync -gopt -e ssh /etc/passwd /etc/shadow lollipop:/etc*** — обратите внимание на одинарное двоеточие), то шифруется все соединение, однако демон ***sshd*** придется сконфигурировать так, чтобы он не запрашивал пароль. Воистину, мы сами выбираем свою казнь!

```
secrets file = /etc/rsyncd.secrets
# Модуль должен быть доступен только для чтения в случае
# рассылки по запросу.
read only = false
# Идентификаторы пользователя и группы для операций рассылки.
uid = root
gid = root
# Список узлов, которым разрешено подключаться.
hosts allow = главный_сервер_рассылки
```

Существует множество других опций, но установки по умолчанию вполне приемлемы. В данной конфигурации все операции локализованы в каталоге **/etc**, а доступ разрешен только указанному узлу. С точки зрения пользователя или клиента можно осуществлять рассылку файлов на сервер, указывая в качестве пункта назначения выражение *узел*::**sysfile**, которое соответствует описанному выше модулю. Если утилита **rsync** должна работать в режиме рассылки по запросу (файлы загружаются с центрального сервера **rsync**), то показанные выше строки конфигурации тоже подойдут (разве что придется включить режим "только чтение").

Наконец, необходимо настроить файл **rsyncd.secrets**. Он обычно хранится в каталоге **/etc** (это может быть любой другой каталог) и содержит пароли, с помощью которых клиенты аутентифицируют себя.

**root**:*пароль*

В общем случае пароли, используемые утилитой **rsync**, должны отличаться от реальных паролей. Поскольку пароли отображаются в незашифрованном виде, файл **rsyncd.secrets** должен быть доступен для чтения только суперпользователю.

## Рассылка файлов по запросу

Реализовать систему рассылки файлов по запросу можно несколькими способами. Самый простой из них заключается в том, чтобы делать системные файлы доступными на центральном FTP- или веб-сервере[3] и заставлять клиентов автоматически загружать их по мере необходимости. Раньше администраторам приходилось писать для этого собственные утилиты (например, сценарии системы **expect**), но сейчас уже существуют стандартные средства.

Наиболее популярная утилита, входящая в состав большинства систем, называется **wget**. Это довольно простая небольшая программа, которая извлекает содержимое по указанному URL-адресу (с использованием протокола FTP либо HTTP). Например, для загрузки файла средствами FTP введите такую команду.

**wget ftp://***пользователь*:*пароль*@*узел*/*путь*/*файл*

Указанный файл будет помещен в текущий каталог.

В случае протокола FTP альтернативой является команда **ncftp**, которая тоже есть в большинстве систем. Она представляет собой улучшенный вариант FTP-клиента и допускает написание простейших сценариев.

Вы можете также использовать утилиту **rsync**, как описано в предыдущем разделе. Если на центральном узле рассылки выполняется сервер **rsync**, то клиенты могут загружать файлы с помощью данной утилиты. Это сложнее, чем в случае FTP, зато расширяются функциональные возможности.

[3] Следует иметь в виду, что передача данных по протоколам HTTP и FTP осуществляется посредством открытого текста. Если вопрос защиты данных является крайне важным, можно воспользоваться протоколами HTTPS и SFTP соответственно.

Какая бы система ни применялась, следите за тем, чтобы на сервер ложилась посильная нагрузка. Если множество клиентов попытается обратиться к серверу по сети одновременно (например, у каждого из них операция обновления запланирована демоном `cron` на одно и то же время), сервер может, хоть и не преднамеренно, подвергнуться атаке вида "отказ в обслуживании". Это особенно касается крупных систем, где необходимо обеспечить смещение таких операций по времени или же их запуск в случайном порядке.

Проще всего распределять задания демона cron с помощью Perl-сценария следующего вида.

```
#!/usr/bin/perl
sleep rand() * 600; # пауза от 0 до 600 секунд (т.е. 10 минут)
system(command_to_copy_files_down);
```

## 19.3. LDAP: упрощенный протокол доступа к каталогам

Организациям, где применяются системы UNIX и Linux, необходим надежный способ распространения своих административных данных. Но проблема в действительности более глобальна. Как быть с неадминистративными ресурсами, например каталогами электронной почты? Как контролировать информацию, предоставляемую внешнему миру? Решением, которое устроило бы всех, является унифицированная служба каталогов.

Служба каталогов — это просто база данных, но такая, в которой сделан ряд дополнительных предположений. Любой набор данных, характеристики которых подпадают под эти предположения, становится кандидатом на включение в базу данных. Основные предположения таковы:

- информационные объекты относительно невелики;
- база данных будет реплицироваться и кешироваться на множестве компьютеров;
- у информации есть атрибуты;
- данные извлекаются часто, но записываются редко;
- операции поиска выполняются очень часто.

Текущая стандартная система, предложенная организацией IETF для этих целей, называется LDAP (Lightweight Directory Access Protocol — упрощенный протокол доступа к каталогам). В спецификациях LDAP говорится не о самой базе данных, а лишь о том, как получить к ней доступ по сети. В то же время заданы схемы организации данных и осуществления поиска, поэтому подразумевается достаточно четкая модель данных.

Изначально LDAP задумывался как простой шлюзовой протокол, который позволял бы клиентам TCP/IP взаимодействовать с серверами каталогов X.500, теперь уже устаревшими. Со временем стало очевидно, что стандарт X.500 изжил себя и в UNIX необходима стандартная служба каталогов. Все это привело к тому, что из LDAP получилась совершенно самостоятельная, полноценная система управления каталогами (возможно, буква "L" в названии стоит незаслуженно)[4].

Как бы там ни было, но протокол LDAP получил широкое распространение, чему отчасти способствовало принятие его фирмой Microsoft в качестве основы для службы Active Directory. В среде UNIX и Linux стандартной реализацией стал пакет OpenLDAP

[4] Вследствие сложной истории LDAP во многих книгах можно встретить подробные рассказы о X.500 и OSI. Однако эта история не имеет никакого отношения к современному использованию LDAP. Просто забудьте о ней.

(**openldap.org**). Служба каталогов уровня предприятия 389 Directory Server (ранее известная как Fedora Directory Server и Netscape Directory Server) также является проектом с открытым исходным кодом, к которому можно получить доступ на сайте **port389.org**. Эта служба работает в системах Linux, Solaris и HP-UX.

## Структура данных LDAP

Данные протокола LDAP принимают форму списков свойств, которые в мире LDAP называют "записями" (entry). Каждая запись состоит из набора именованных атрибутов (например, "uid", "description") и значений этих атрибутов. Пользователи, работающие в среде Windows, вероятно, заметят, что эта структура напоминает структуру записей в системном реестре. Как и в реестре Windows, отдельный атрибут может иметь несколько различных значений.

В качестве примера рассмотрим обычную (и упрощенную) строку файла **/etc/passwd**, выраженную в виде записи LDAP.

```
uid: ghopper
cn: Grace Hopper
userPassword: {crypt}$1$pZaGA2RL$MPDJoc0afuhHY6yk8HQFp0
loginShell: /bin/bash
uidNumber: 1202
gidNumber: 1202
homeDirectory: /home/ghopper
```

Здесь мы видим простой пример формата LDIF (LDAP Data Interchange Format — формат обмена данными LDAP), который применяется в большинстве инструментов, связанных с LDAP, и в реализациях серверов. Причиной успеха этого формата является то обстоятельство, что LDAP можно без труда преобразовывать в простой текст и обратно.

Записи систематизируются по "отличительным именам" (имя атрибута: **dn** — *d*istinguished *n*ame), которые формируют некоторую разновидность пути поиска. Например, запись **dn** для вышеупомянутого пользователя может выглядеть так.

```
dn: uid=ghopper,ou=People,dc=navy,dc=mil
```

Как и в DNS, самый значащий бит ставится справа. Здесь имя DNS **navy.mil** используется для построения верхних уровней иерархии LDAP. Оно разбивается на два компонента домена: **navy** и **mil**, хотя это всего лишь одно из некоторых обычных соглашений.

Каждая запись имеет одно отличительное имя. Поэтому иерархия записей будет выглядеть подобно простому ветвлению без циклов. Кроме того, существуют также и резервы для символических ссылок между записями и для отсылок на другие серверы.

Записи LDAP обычно составляются на основе использования атрибута **objectClass** (объектный класс). Классы объектов определяют атрибуты, которые может содержать запись, причем некоторые из них могут требоваться для проверки достоверности. Каждому атрибуту назначается тип данных. Схема вложения и комбинирования классов объектов ничем не отличается от схемы, принятой в традиционном объектно-ориентированном программировании. Верхний уровень дерева класса объектов называется **top**; он означает лишь то, что запись должна иметь атрибут **objectClass**.

В табл. 19.2 приведены некоторые обычные атрибуты LDAP, назначение которых с первого взгляда может быть непонятным.

**Таблица 19.2. Некоторые имена атрибутов, которые можно встретить в иерархиях LDAP**

| Атрибут | Расшифровка | Назначение |
|---|---|---|
| `O` | *O*rganization (организация) | Часто идентифицирует запись верхнего уровня сайта[a] |
| `Ou` | *O*rganization *u*nit (организационная единица) | Логическое подразделение, например "marketing" (отдел маркетинга) |
| `Cn` | *C*ommon *n*ame (обычное имя) | Наиболее естественное имя, с помощью которого можно передать смысл записи |
| `Dc` | *D*omain *c*omponent (компонент домена) | Используется в сайтах, в которых иерархия построена на основе DNS |
| `objectClass` | Object class (объектный класс) | Схема, в соответствии с которой формируются атрибуты данной записи |

[a] Обычно не используется системами, в которых LDAP-иерархия построена на основе DNS.

## Особенности LDAP

Для уверенной работы с LDAP вам нужен хотя бы небольшой опыт. Протокол LDAP сам по себе не решает какую-либо административную проблему. Нет какой-то "главной задачи", которую можно было бы решить исключительно с помощью LDAP; к тому же, на разных сайтах по-разному подходят к необходимости развертывания серверов LDAP. Поэтому прежде чем перейти к особенностям инсталляции и конфигурирования OpenLDAP, поговорим о том, в каких случаях целесообразно использовать LDAP.

- LDAP можно использовать в качестве центрального хранилища полной информации о пользователях (от их телефонных номеров и домашних адресов до зарегистрированных имен и паролей).

🕮 Подробнее об использовании LDAP с агентом передачи почты **sendmail** написано в разделе 20.7.

- В дальнейшем LDAP можно использовать для распространения конфигурационной информации для вспомогательных приложений. Многие почтовые системы, включая **sendmail**, Exim и Postfix, могут получить данные о маршрутизации из LDAP, и этим отчасти можно объяснить популярность применения LDAP. Инструменты будут отличаться в зависимости от конфигурации веб-сервера Apache и программы автоматического монтирования файловых систем по запросу **autofs**. Вероятно, со временем поддержка LDAP будет все более и более расширяться.
- LDAP позволяет упростить процесс аутентифицирования пользователей для приложений (даже тех, которые написаны другими командами программистов и в других подразделениях), избавляя разработчиков от необходимости беспокоиться о точных деталях управления учеными записями.
- Изменения, внесенные в LDAP-данные, немедленно вступают в силу и мгновенно становятся видимыми для всех узлов и приложений клиентов.
- Доступ к данным LDAP производится с помощью инструментов командной строки, таких как **ldapsearch**. Кроме того, LDAP всесторонне поддерживается языками написания сценариев вроде Perl и Python (посредством использования библиотек). Следовательно, LDAP выгодно использовать для передачи информации о конфигурации локально написанным сценариям и утилитам администрирования.

- Для управления LDAP доступен превосходный веб-инструментарий, например phpLDAPadmin (`phpldapadmin.sourceforge.com`) и Directory Administrator (`diradmin.open-it.org`). Эти инструменты настолько просты в использовании, что вы сможете приступить к работе с ними, даже не читая руководства.
- LDAP хорошо поддерживается и как служба общедоступных каталогов. Многие ведущие почтовые клиенты применяют LDAP для доступа к каталогам пользователей. Простой поиск с использованием LDAP поддерживается во многих веб-браузерах посредством типа LDAP URL.
- Архитектура Microsoft Active Directory построена на LDAP, а текущая версия Windows Server включает расширения (раньше они назывались "Службы для UNIX", затем "Службы каталогов и защиты Windows для UNIX" а сейчас — "Windows Server 2008 UNIX Interoperability Components"), с помощью которых можно выполнять отображение пользователей и групп UNIX. Подробнее об объединении систем UNIX с протоколом LDAP, ориентированным на использование архитектуры Active Directory, написано в главе 30.

## Документация и спецификации LDAP

В качестве общего введения в LDAP мы можем порекомендовать неплохой "хрестоматийный учебник" *LDAP for Rocket Scientists*, который содержит описание архитектуры и протокола LDAP. Эта книга опубликована в электронном виде на сайте `zytrax.com/books/ldap`. Кроме того, существует множество разнообразных документов RFC, посвященных протоколу LDAP, каждый из которых является довольно сложным. Наиболее важные документы перечислены в табл. 19.3.

**Таблица 19.3. Документы RFC, посвященные протоколу LDAP**

| RFC | Название |
|---|---|
| 2307 | An Approach for Using LDAP as a Network Information Service |
| 2820 | Access Control Requirements for LDAP |
| 2849 | LDAP Data Interchange Format (LDIF)—Technical Specification |
| 3112 | LDAP Authentication Password Schema |
| 3672 | Subentries in the Lightweight Directory Access Protocol (LDAP) |
| 4511 | LDAP: The Protocol |
| 4512 | LDAP: Directory Information Models |
| 4513 | LDAP: Authentication Methods and Security Mechanisms |
| 4514 | LDAP: String Representation of Distinguished Names |
| 4515 | LDAP: String Representation of Search Filters |
| 4516 | LDAP: Uniform Resource Locator |
| 4517 | LDAP: Syntaxes and Matching Rules |
| 4519 | LDAP: Schema for User Applications |

## OpenLDAP: традиционный LDAP-сервер с открытым исходным кодом

OpenLDAP — это расширение проекта, выполненного в Мичиганском университете. В настоящее время он продолжает оставаться проектом с открытым исходным текстом

и распространяется с большинством дистрибутивов Linux (хотя и не всегда включается в стандартный вариант инсталляции). Для использования OpenLDAP в системах Solaris, HP-UX или AIX вам придется предварительно загрузить и инсталлировать необходимые компоненты соответствующей реализации.

В дистрибутиве OpenLDAP стандартным демоном LDAP-сервера является **slapd**. В среде с несколькими серверами OpenLDAP демон **slurpd** выполняется на главном сервере и отрабатывает механизм репликации посредством внесения изменений на подчиненные серверы. Утилиты командной строки позволяют выполнять запросы и модифицировать данные LDAP.

Настройка не является сложной. В первую очередь потребуется создать файл **/etc/openldap/slapd.conf**, скопировав образец, который был инсталлирован вместе с сервером OpenLDAP. Необходимо обратить внимание на следующие строки.

```
database bdb
suffix "dc=mydomain, dc=com"
rootdn "cn=admin, dc=mydomain, dc=com"
rootpw {crypt}abJnggxhB/yWI
directory /var/lib/ldap
```

По умолчанию выбирается формат базы данных Berkley DB, отлично подходящий для данных, манипуляция которыми будет производиться в системе OpenLDAP. Вы можете использовать разнообразные интерфейсы, включая специальные методы (например, сценарии, создающие данные "на лету").

Переменная **suffix** задает "базовое имя" LDAP. Это корень вашей части пространства имен LDAP, подобный тому, что в DNS называется доменным именем. Этот пример показывает обычное использование доменного имени в качестве базового имени LDAP.

Переменная **rootdn** задает административное имя, а переменная **rootpw** — пароль администратора в стандартном формате UNIX (алгоритм DES). Обратите внимание на то, что доменные компоненты, восходящие к административному имени, также должны быть указаны. Пароль можно либо скопировать из файла **/etc/shadow** (если не используются пароли MD5), либо сгенерировать с помощью простейшего Perl-сценария.

```
perl -e "print crypt('пароль','примесь');"
```

Здесь *примесь* — это произвольная двухсимвольная строка. Из-за наличия пароля следует убедиться в том, что режим доступа к файлу **slapd.conf** равен 600, а его владельцем является пользователь **root**.

Нужно также отредактировать файл **/etc/openldap/ldap.conf**, чтобы указать сервер по умолчанию и базовое имя для клиентских запросов LDAP. Это несложно: просто задайте аргумент записи **host** равным имени сервера, а в переменную **base** занесите то же значение, что и в переменной **suffix** файла **slapd.conf** (убедитесь, что обе строки не являются комментариями).

С этого момента можно запускать демон **slapd** без аргументов.

## 389 Directory Server: альтернативный LDAP-сервер с открытым исходным кодом

Подобно OpenLDAP, служба каталогов уровня предприятия 389 Directory Server (**port389.org**) является расширением проекта, выполненного в Мичиганском университете. Однако прошло несколько лет (в компании Netscape Communications), прежде чем этот проект снова стал открытым.

Существует множество причин рассматривать проект 389 Directory Server как альтернативу для OpenLDAP, но среди его явных преимуществ все же стоит выделить более совершенную документацию. Проект 389 Directory Server поставляется с инструкциями профессионального уровня по администрированию и использованию, включая подробные руководства по инсталляции и внедрению.

К ключевым особенностям проекта 389 Directory Server можно отнести следующие:

- использование нескольких полностью равноправных мастер-серверов, что позволяет обеспечить отказоустойчивость и высокую скорость выполнения операций записи;
- возможность синхронизации пользователей, групп и паролей с контроллерами домена Active Directory;
- консоль администрирования с графическим интерфейсом, управления из командной строки и через веб-интерфейс;
- поддержка протокола LDAP, оперативное (без потерь времени) обновление схемы, конфигурации и управления, а также мощный механизм разграничения доступа вплоть до уровня отдельных атрибутов.

Проект 389 Directory Server, по-видимому, имеет более активных разработчиков, чем проект OpenLDAP. Поэтому мы обычно рекомендуем отдавать предпочтение именно проекту 389 Directory Server, а не OpenLDAP.

С административной точки зрения эти два сервера с открытым исходным кодом поразительно сходны в том, что касается структуры и функционирования. И это не удивительно, поскольку оба пакета были построены на базе одного и того же исходного кода.

## LDAP вместо /etc/passwd и /etc/group

Включить поддержку LDAP на стороне клиента будет несложно. В некоторых системах необходимый пакет **nss_ldap** инсталлируется по умолчанию; в противном случае пакет можно найти в качестве опции. Этот пакет включает модуль PAM, с помощью которого можно использовать LDAP с подключаемыми модулями аутентификации в дополнение к переключателю службы имен. (Подробнее об интеграции систем UNIX и Linux с протоколом LDAP, построенным на базе Active Directory, написано в главе 30.)

Параметры этого пакета, используемые по умолчанию в LDAP на стороне клиента, задаются в файле **/etc/ldap.conf**, который совместно использует свой формат с описанным выше файлом **/etc/openldap/ldap.conf**, но который включает дополнительные опции, специфические для службы имен и контекста PAM. Вы должны будете также отредактировать файл **/etc/nsswitch.conf** на каждом клиенте, чтобы добавить **ldap** в качестве источника для каждого типа данных, который будет приведен к стандарту LDAP. (Изменения в файле **nsswitch.conf** приводят к тому, что библиотека языка C передает запросы библиотеке **libnss_ldap**, которая затем использует информацию **/etc/ldap.conf**, чтобы выяснить, как нужно обращаться с запросами LDAP. Подробнее — в разделе 19.5.)

В документе RFC2307 определено стандартное преобразование традиционных наборов данных UNIX, к которым относятся файлы **passwd** и **group**, в пространство имен LDAP. Это полезный справочный документ (по крайней мере, с теоретической точки зрения) для системных администраторов, использующих LDAP в качестве замены NIS. На практике все эти спецификации читать гораздо проще компьютерам, чем людям; вам же лучше просто просмотреть примеры.

Фирма Padl Software предлагает бесплатную коллекцию сценариев, которые переводят существующие текстовые файлы или карты NIS в LDAP. Эти сценарии можно найти по адресу: **padl.com/OSS/MigrationTools.html**. Работать с ними очень просто. Сценарии можно использовать в качестве фильтров для генерации LDIF, а также можно запускать, чтобы загружать данные прямо с сервера. Например, сценарий **migrate_group** преобразовывает взятую из **/etc/group** строку

```
csstaff:x:2033:evi,matthew,trent
```

в следующий блок LDIF.

```
dn: cn=csstaff,ou=Group,dc=domainname,dc=com
cn: csstaff
objectClass: posixGroup
objectClass: top
userPassword: {crypt}x
gidNumber: 2033
memberuid: evi
memberuid: matthew
memberuid: trent
```

Обратите внимание на спецификации классов объектов и отличительные имена, которые были опущены в примере **passwd** в разделе 19.3.

После импортирования базы данных вы сможете проверить работу преобразования, запустив утилиту **slapcat**, которая отображает всю базу данных.

## Создание LDAP-запросов

Для администрирования LDAP вам не обойтись без просмотра содержимого базы данных и управления им. Для этого вам очень пригодится упомянутый выше свободно распространяемый пакет **phpLDAPadmin**, который предоставляет удобный интерфейс, работающий по принципу "указал и щелкнул". Помимо phpLDAPadmin, можно использовать утилиту **ldapsearch** (распространяемую с OpenLDAP и 389 Directory Server), которая предназначена для поиска информации в службе каталога и является аналогичным инструментом командной строки, генерирующим результат в формате LDIF. Утилита **ldapsearch** особенно полезна для вызова из сценариев, а также для сред отладки, в которых служба Active Directory действует в качестве сервера LDAP.

В следующем примере запроса используется утилита **ldapsearch** для просмотра информации о каталогах тех пользователей, у которых обычное имя **cn** (*common name*) начинается с "**ned**". (В данном случае найден только один результат, соответствующий такому запросу.) Назначения используемых здесь флагов рассматриваются ниже.

```
$ ldapsearch -h atlantic.atrust.com -p 389
    -x -D "cn=trent,cn=users,dc=boulder,dc=atrust,dc=com" -W
    -b "CN=users,DC=boulder,DC=atrust,DC=com" "cn=ned*"

Enter LDAP Password: пароль

# LDAPv3
# base <CN=users,DC=boulder,DC=atrust,DC=com> with scope sub
# filter: cn=ned*
# requesting: ALL
#
# ned, Users, boulder.atrust.com
dn: CN=ned,CN=Users,DC=boulder,DC=atrust,DC=com
```

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: ned
sn: McClain
telephoneNumber: 303 245 4505
givenName: Ned
distinguishedName: CN=ned,CN=Users,DC=boulder,DC=atrust,DC=com
displayName: Ned McClain
memberOf: CN=Users,CN=Builtin,DC=boulder,DC=atrust,DC=com
memberOf: CN=Enterprise Admins,CN=Users,DC=boulder,DC=atrust,DC=com
name: ned
sAMAccountName: ned
userPrincipalName: ned@boulder.atrust.com
lastLogonTimestamp: 129086952498943974
mail: ned@atrust.com
```

Флаги **-h** и **-p** команды **ldapsearch** позволяют указать в запросе узел и порт сервера LDAP соответственно.

Обычно вам потребуется аутентифицировать себя на сервере LDAP. В этом случае используются специальные флаги. Флаг **-x** запрашивает простую аутентификацию (в отличие от SASL). Флаг **-D** идентифицирует отличительное имя учетной записи пользователя, который имеет права доступа, необходимые для выполнения запроса. Наконец, флаг **-W** предписывает утилите **ldapsearch** предложить вам ввести соответствующий пароль.

Флаг **-b** указывает утилите **ldapsearch**, где именно в иерархии LDAP начать поиск. Этот параметр известен как **baseDN** (отсюда и имя флага "b"). По умолчанию утилита **ldapsearch** возвращает все соответствующие критерию поиска строки, найденные ниже базовой точки дерева **baseDN**, т.е. поиск будет выполняться только в дочерних элементах базы поиска (включая ее саму). Подстроить характер такого поиска можно с помощью флага **-s**.

Последний аргумент представляет собой "фильтр", который описывает объект вашего поиска. Он не требует использования флагов. Этот фильтр, **cn=ned***, обеспечивает возвращение всех строк LDAP, "обычное имя" которых начинается с "ned". Этот фильтр заключается в кавычки (**"cn=ned*"**), чтобы специальные символы универсализации в строке поиска (в данном случае это "звездочка") не были восприняты системой как управляющие символы командной оболочки.

Если вы хотите извлечь все записи ниже заданной базы **baseDN**, просто используйте в качестве фильтра поиска выражение **objectClass=*** — или же опустите фильтр вообще, поскольку такой характер поиска действует по умолчанию.

Любые аргументы, которые указаны за фильтром, обеспечивают выбор конкретных атрибутов для возвращаемого результата. Например, если бы вы добавили в конец приведенной выше командной строки аргумент **mail givenName**, утилита **ldapsearch** возвратила бы только эти атрибуты отфильтрованных записей.

## LDAP и безопасность

Так уж сложилось, что протокол LDAP использовался чаще всего по принципу телефонного каталога, поэтому отправка данных без шифрования была обычным делом. Как результат, "стандартная" реализация LDAP предоставляет незашифрованный доступ че-

рез TCP-порт 389. Однако мы настоятельно не советуем использовать незашифрованный протокол LDAP для передачи информации об аутентификации, даже если пароли индивидуально хешируются или зашифровываются.

Как вариант, во многих ситуациях (даже в мире Microsoft) LDAP можно использовать на основе SSL (эта комбинация известна как LDAPS; обычно эта пара работает на TCP-порте 686) на стороне как сервера, так и клиента. Такой способ доступа является предпочтительным, поскольку он защищает информацию, содержащуюся как в запросе, так и в ответе. Желательно использовать LDAPS везде, где это возможно.

Столь сложная система, как LDAP, неизбежно содержит бреши, ослабляющие ее защиту, не говоря уже о брешах, издавна существующих в операционной системе. Поэтому системным администраторам постоянно приходится быть начеку.

# 19.4. NIS: сетевая информационная служба

Административная СУБД NIS (Network Information Service — сетевая информационная служба), выпущенная компанией Sun в 80-х годах, была первой СУБД такого рода. Сначала она называлась Sun Yellow Pages (желтые страницы Sun), но по причинам правового характера ее пришлось переименовать. Команды NIS до сих пор начинаются с префикса **yp**, поскольку имя, данное при рождении, забыть трудно. СУБД NIS была с воодушевлением принята поставщиками UNIX-систем и поддерживается во всех дистрибутивах Linux.

Но в наши дни для новых систем все же не следует использовать NIS, причем не только из-за неизбежной интеграции с системами Windows, но и из-за несовершенства NIS-системы безопасности и масштабируемости.

Тем не менее мы кратко рассмотрим службу NIS из уважения к большому числу действующих систем, на которых она все еще используется.

## Модель NIS

Единицей совместного использования в NIS является *запись*, а не файл. Запись обычно соответствует одной строке конфигурационного файла. Главный сервер хранит официальные копии системных файлов, которые находятся в исходных каталогах и имеют текстовый формат. Серверный процесс делает эти файлы доступными по сети. Сервер и его клиенты образуют домен NIS[5].

Для повышения эффективности поиска текстовые файлы преобразуются в файлы базы данных с помощью хеширующих функций библиотеки. После редактирования файлов на главном сервере необходимо попросить NIS преобразовать их в хешированный формат с помощью программы **make**. Полученный файл называется *картой* (map).

С каждой записью может быть связан всего один ключ, поэтому системные файлы иногда приходится транслировать в несколько карт NIS. Например, файл **/etc/passwd** преобразуется в две карты: **passwd.byname** и **passwd.byuid**. Первая служит для отбора записей по имени пользователя, а вторая — для поиска по идентификатору. Любую из них можно использовать для выборки всех записей файла **passwd**, но хеширующие функции не сохраняют порядок записей, поэтому нельзя воссоздать точную копию исходного файла.

[5] Не следует путать домены NIS с доменами DNS. Это абсолютно разные понятия, не имеющие ничего общего.

СУБД NIS позволяет реплицировать карты сети среди группы подчиненных серверов. Это дает возможность ослабить нагрузку на главный сервер и поддерживать функционирование клиентов даже в том случае, когда некоторые серверы недоступны. Если на главном сервере файл изменился, необходимо разослать соответствующую карту NIS всем подчиненным серверам, чтобы везде были одинаковые данные. Клиенты не различают главный и подчиненные серверы.

## Схема работы NIS

Файлы данных NIS хранятся в одном каталоге, обычно в каталоге **/var/yp**. Далее мы будем называть его "NIS-каталогом". Все карты NIS хранятся в хешированном виде (в формате базы данных) в подкаталоге NIS-каталога, соответствующем домену NIS. Точное имя и количество таких файлов зависит от используемой библиотеки хеширующих функций. Для каждого поля (ключа), по которому можно осуществить поиск в файле, должна быть создана отдельная карта. Например, в домене **cssuns** карты DB для файла **/etc/passwd** могут называться так.

```
/var/yp/cssuns/passwd.byname
/var/yp/cssuns/passwd.byuid
```

Команда **makedbm** создает карты NIS из обычных файлов. Ее никогда не нужно вызывать непосредственно. Файл **Makefile** в каталоге **/var/yp** сконфигурирован так, чтобы автоматически генерировать все распространенные карты NIS. После модификации какого-нибудь системного файла перейдите в каталог **/var/yp** и запустите программу **make**. Она сверит время модификации каждого файла со временем модификации соответствующих карт и выполнит команду **makedbm** для каждой карты, который необходимо перестроить.

В системах HP-UX вместо команды **make** используется команда **ypmake**.

Копирование карт с главного сервера на подчиненные осуществляет команда **ypxfr**. Она работает по модели запроса: для того чтобы она импортировала карты, ее нужно запускать на каждом подчиненном сервере. Подчиненные серверы время от времени выполняют команду **ypxfr** для того, чтобы проверить, последние ли версии карт находятся в их распоряжении. С помощью демона **cron** можно управлять периодичностью этих проверок.

Стандартная реализация механизма копирования карт несколько неэффективна, поэтому в Linux существует демон **rpc.ypxfrd**, который можно запустить на главном сервере для ускорения ответов на запросы команды **ypxfr**. Этот демон работает в обход стандартного протокола NIS и просто рассылает копии файлов карт. К сожалению, в разных системах файлы карт хранятся с использованием различных форматов баз данных и разного порядка следования байтов, поэтому применение команды **ypxfrd** чревато возможной несовместимостью.

Команда **yppush** используется на главном сервере. Она не пересылает никаких данных, а просто заставляет каждый подчиненный сервер выполнить команду **ypxfr**. Команда **yppush** задается в файле **Makefile**, находящемся в NIS-каталоге, и обеспечивает принудительную рассылку откорректированных карт на подчиненные серверы.

Существует специальная карта **ypservers**, которая не соответствует ни одному обычному файлу. Он содержит список всех серверов домена и создается автоматически при

конфигурировании домена с помощью команды **ypinit**. Содержимое этой карты изучается всякий раз, когда главному серверу нужно разослать карты на подчиненные серверы.

После начального конфигурирования единственными активными компонентами системы NIS остаются демоны **ypserv** и **ypbind**. Первый из них работает только на серверах (и главном, и подчиненных); он принимает запросы от клиентов и отвечает на них, осуществляя поиск информации в хешированных файлах карт.

Демон **ypbind** работает на всех компьютерах NIS-домена, включая серверы. Функции библиотеки языка C обращаются к локальному демону **ypbind** всякий раз, когда им нужно ответить на административный запрос (при условии, что это разрешено установками файла **/etc/nsswitch.conf**). Демон **ypbind** находит в соответствующем домене демон **ypserv** и возвращает его адрес библиотечной функции, которая затем обращается непосредственно к серверу.

Современные версии демона **ypbind** периодически проверяют, работают ли они с наиболее активным сервером в домене NIS. Это является улучшением по сравнению с традиционной реализацией демона, в которой осуществлялась привязка к определенному серверу.

В NIS есть ряд вспомогательных команд, которые предназначены для изучения карт, определения версий карт, используемых каждым сервером, и управления привязкой клиентов к серверам. Полный перечень команд и демонов NIS приведен в табл. 19.4.

**Таблица 19.4. Команды и демоны NIS**

| Утилита | Описание |
|---|---|
| **ypserv** | Демон сервера NIS; запускается на этапе начальной загрузки |
| **ypbind** | Демон клиента NIS; запускается на этапе начальной загрузки |
| **domainname** | Задает домен NIS, в который входит компьютер (выполняется на этапе начальной загрузки) |
| **ypxfr** | Загружает текущую версию карты с главного сервера |
| **ypxfrd** | Обслуживает запросы, поступающие от команды **ypxfr** (работает на главном сервере) |
| **yppush** | Заставляет подчиненные серверы обновить свои версии карты |
| **makedbm** | Создает хешированную карту из обычного файла |
| **ypmakeà** | Обновляет хешированные карты для тех файлов, которые изменились |
| **ypinit** | Конфигурирует компьютер как главный или подчиненный сервер |
| **ypset** | Заставляет демон **ypbind** установить соединение с конкретным сервером[a] |
| **ypwhich** | Определяет, с каким сервером работает текущий компьютер |
| **yppoll** | Определяет, какую версию карты использует сервер |
| **ypcat** | Отображает значения, содержащиеся в карте NIS |
| **ypmatch** | Отображает элементы карты, соответствующие заданному ключу |
| **yppasswd** | Изменяет пароль на главном сервере NIS |
| **ypchfn** | Изменяет содержимое поля GECOS на главном сервере NIS |
| **ypchsh** | Меняет регистрационный интерпретатор команд на главном сервере NIS |
| **yppasswdd** | Сервер для команд **yppasswd**, **ypchsh** и **ypchfn** |
| **ypupdated**[б] | Сервер для обновления карты NIS (управляется демоном **inetd**) |

[a] Нужно отдельно включать с помощью команды **ypbind -ypsetme** или **ypbind -ypset** (второй вариант опаснее).

[б] Не используется или не поддерживается во всех системах.

## Безопасность NIS

Система NIS небезопасна. Особенно неприятен широковещательный режим. Любой узел сети может выступить в качестве сервера для того или иного домена и распространить среди NIS-клиентов ложные административные данные. В системах Linux эту проблему можно смягчить, если явно указать каждому клиенту доступные ему серверы NIS.

Если вопросы безопасности системы для вас очень важны, не следует использовать NIS для управления скрытыми паролями. Используйте для этого такие альтернативные распределенные механизмы аутентификации, как LDAP.

В старых версиях СУБД NIS содержала множество "прорех". Поэтому используйте только современную версию NIS.

# 19.5. Задание приоритетов для источников административной информации

Информацию о конфигурации можно распространять несколькими способами. Любая система способна обрабатывать обычные файлы и осуществлять поиск имен компьютеров и IP-адресов в DNS. Поскольку для каждого элемента информации может существовать несколько потенциальных источников, предусмотрен способ задания источников и порядка их опроса.

Конфигурационный файл **/etc/nsswitch.conf** (**/etc/netsvc.conf** в системе AIX) позволяет для каждого типа административной информации указывать явный путь поиска. Типичный файл **nsswitch.conf** выглядит следующим образом.

```
passwd: files ldap
hosts: files dns
group: files
...
```

Каждая строка соответствует отдельному типу информации (обычно это эквивалент одного текстового файла). Обычные источники таковы: **nis**, **nisplus**, **files**, **dns**, **ldap** и **compat**. Им соответствуют (в порядке перечисления) NIS, NIS+[6], обычные текстовые файлы (без учета специальных обозначений вроде "+"), DNS, LDAP и обычные файлы системы NIS. DNS предоставляет информацию только о узлах и сетях.

Источники просматриваются слева направо, пока один из них не выдаст ответ на запрос. В показанном выше примере функция **gethostbyname** сначала проверит файл **/etc/hosts** и, если требуемый узел там не указан, обратится к DNS. В процессе обработки запросов, касающихся UNIX-групп, будет проверяться только файл **/etc/group**.

В случае необходимости можно явно указать, как следует поступать при получении отрицательного ответа на запрос от того или иного источника. Соответствующее условие берется в квадратные скобки. Например, строка

```
hosts:  dns [NOTFOUND=return] files
```

заставляет получать информацию только от DNS, если эта служба доступна. Получение отрицательного ответа от сервера имен приведет к немедленному завершению запроса (с выдачей кода ошибки) без обращения к текстовым файлам. В то же время текстовые файлы будут задействованы, если все серверы имен окажутся недоступными. В табл. 19.5 перечислены различные условия проверки ошибок. Каждое из них может быть задано

[6] Неудачный "наследник" исходной версии NIS уже не поддерживается фирмой Sun, хотя в некоторых системах его все еще можно встретить.

равным **return** или **continue**, что означает прерывание запроса или переход к следующему источнику соответственно.

**Таблица 19.5. Условия проверки ошибок в файле /etc/nsswitch.conf**

| Условие | Смысл |
|---|---|
| UNAVAIL | Источник не существует или недоступен |
| NOTFOUND | Источник существует, но не может ответить на запрос |
| TRYAGAIN | Источник существует, но занят |
| SUCCESS | Источник смог ответить на запрос |

В большинство дистрибутивов систем входит файл **nsswitch.conf**, который подходит для изолированного компьютера. Источником всех данных служат обычные файлы, за исключением информации о узлах (сначала опрашиваются файлы, потом — DNS). В некоторых системах для файлов **passwd** и **group** задан источник **compat**, что, скорее всего, нежелательно. Если вы действительно используете NIS, просто вставьте этот источник в файл **nsswitch.conf**.

## Демон nscd: кеширование результатов поиска

В некоторых дистрибутивах Linux большую роль в обработке системных файлов играет демон **nscd**, название которого немного сбивает с толку (*n*ame *s*ervice *c*ache *d*aemon — демон кеширования для службы имен).

Демон **nscd** работает в связке с библиотекой языка C, кешируя результаты таких функций, как **getpwent**, и подобных ей. По сути, демон является просто оболочкой этих функций; ему не известно, какие источники данных опрашивались. Теоретически демон должен способствовать повышению эффективности поиска информации в системных файлах, но с точки зрения пользователей реальный эффект практически не ощутим.

О DNS рассказывалось в главе 17.

Мы говорим, что название "демон кеширования для службы имен" способно ввести в заблуждение, поскольку термин "служба имен" обычно относится к DNS, распределенной СУБД, осуществляющей связь между доменными именами и IP-адресами. Демон **nscd**, в числе прочего, кеширует и результаты DNS-запросов (одна из контролируемых им функций — **gethostbyname**), он также связан с функциями, черпающими информацию из файлов **passwd** и **group**, и их сетевыми эквивалентами (из соображений безопасности, результаты запросов к файлу **/etc/shadow** не кешируются).

В принципе, демон **nscd** не должен влиять на работу системы; он лишь ускоряет выполнение повторяющихся операций поиска. Но на практике влияние демона может оказаться непредсказуемым из-за того, что он хранит собственную копию результатов поиска. Эти данные находятся в кеше в течение фиксированного интервала времени (задается в конфигурационном файле демона — **/etc/nscd.conf**), поэтому всегда есть вероятность того, что самые последние изменения не будут отражены в кеше, пока прежние данные не устареют. Вообще-то, демон достаточно "умен" и следит за состоянием локальных источников данных (например, файлом **/etc/passwd**), поэтому локальные изменения должны вступать в силу в течение 15 секунд. Что касается удаленных источников, например NIS, то в худшем случае величина задержки будет равна периоду устаревания.

Мы с вами упоминали несколько примеров дистрибутивов. Так вот, только SUSE запускает **nscd** по умолчанию. В системе Red Hat демон **nscd** инсталлируется, но по

умолчанию не запускается на этапе загрузки системы; чтобы разрешить использование демона **nscd**, выполните команду **chkconfig nscd on**. Система Ubuntu разрешает работу демона **nscd**, но не инсталлирует его по умолчанию; чтобы его загрузить, нужно выполнить команду **apt-get install nscd**.

Демон **nscd** запускается на этапе начальной загрузки и работает непрерывно. По умолчанию в файле **/etc/nscd.conf** задан период 10 минут для файла **passwd** и 1 час — для файлов **hosts** и **group**. Интервал между попытками в случае неудачного запроса составляет 20 секунд. Эти значения редко приходится менять. Если вас удивляет, что сделанные вами изменения не вступили в силу, то, очевидно, виной тому демон **nscd**.

## 19.6. Рекомендуемая литература

- Carter, Gerald. *LDAP System Administration*. Sebastopol, CA: O'Reilly Media, 2003.
- Malere, Luiz Ernesto Pinheiro. *LDAP Linux HOWTO*. **tldp.org**
- Volgmaier, Reinhard. *The ABCs of LDAP: How to Install, Run, and Administer LDAP Services*. Boca Raton, FL: Auerbach Publications, 2004.
- *LDAP for Rocket Scientists*. **zytrax.com/books/ldap**

## 19.7. Упражнения

19.1. Почему метод обновления файлов локального компьютера по запросу считается безопаснее метода принудительной рассылки?

19.2. Поясните приведенный ниже фрагмент управляющего файла утилиты **rdist**.

```
LINUX_PASSWD = ( redhatbox ubuntubox susebox )

passwd:
   ( /etc/passwd ) -> ( ${LINUX_PASSWD} )
   install /etc/passwd.rdist;
   cmdspecial /etc/passwd.rdist "/usr/local/sbin/mkpasswd";
```

19.3. ☆ Объясните разницу между утилитами **rdist** и **rsync**. В каких случаях предпочтительнее использовать ту или другую?

19.4. ☆ Какой метод совместного использования файлов применяется в вашей системе? Какие проблемы безопасности при этом возникают? Предложите альтернативную схему совместного использования файлов и укажите ее преимущества и недостатки.

19.5. ☆☆☆☆☆ Придумайте схему LDAP, которая хранила бы такую информацию о пользователе, как регистрационное имя, пароль, оболочка, авторизированные компьютеры и т.п. Создайте инструмент, с помощью которого можно в интерактивном режиме добавлять новых пользователей в базу данных из файла, содержащего список пользователей. Создайте инструмент, который будет генерировать файлы **passwd**, **group** и **shadow** из базы данных LDAP для компьютеров, работающих в вашем отделе. Разрешите пользователям иметь различные пароли на каждом компьютере. (Чтобы пользоваться каждым компьютером, не всем пользователям нужна обязательная авторизация.) Ваша система **adduser** должна уметь печатать списки регистрационных имен существующих пользователей и пары "регистрационное имя/пароль" для новых пользователей.

# Глава 20

# Электронная почта


Социальные сети и пересылка SMS-сообщений постепенно вытесняют электронную почту в категорию "устаревших технологий", поскольку возникают новые отношения и новые темы для обсуждения (microthoughts). Тем не менее электронная почта остается универсальным стандартом общения по сети. Все, от старушек до огромных корпораций, привыкли использовать электронную почту для общения с семьей, коллегами, партнерами, потребителями и даже с правительством. Этот безумный, безумный, безумный мир электронной почты![1]

Электронная почта проста и удобна; если вы знаете чей-нибудь почтовый адрес, то набираете сообщение и щелкните на кнопке Send. Все! Уже через несколько секунд это сообщение окажется в электронном почтовом ящике адресата, независимо от того, где он находится: в соседней комнате или в другой части Земного шара. С точки зрения пользователя, ничего не может быть проще.

Инфраструктура, лежащая в основе электронной почты и обеспечивающая все ее удобства, напротив, довольно сложная. Существует несколько программных пакетов, с помощью которых можно посылать сообщения электронной почты и управлять ею (три из них будут рассмотрены в этой главе), но все они требуют довольно сложного конфигурирования и управления. Кроме того, необходимо понимать основные концепции и протоколы, связанные с электронной почтой, чтобы у вас не сложилось ложное представление, что кроссплатформенная и интерорганизованная (interorganizational) электронная почта — это подарок небес, работающий по мановению волшебной палочки.

Понимание собственной инфраструктуры электронной почты и управление ею требуется не только от системного администратора. Многие провайдеры сегодня предлага-

[1] Даже когда Эви пересекала на паруснике океан, она всегда находилась в контакте по электронной почте с помощью своего бортового радиопередатчика HAM/SSB и "скоростного" пакета для радиосвязи, скорость которого в лучшем случае приближалась к 30 бод.

ют "управляемую" службу электронной почты, которая размещает систему электронной почты на удаленных серверах за ежемесячную или ежегодную плату (иногда за каждого пользователя отдельно). Существует также множество аналогичных "бесплатных" служб, таких как Gmail компании Google, Yahoo! Mail и MSN Hotmail, имеющих большую популярность среди индивидуальных пользователей. Если вы хотите получить персональную учетную запись электронной почты или открыть почтовый адрес для маленькой компании, то эти службы будут удобными для вас. Служба Gmail не только предоставляет персональные учетные записи электронной почты, но и обладает интересной возможностью настраивать электронную почту для всего домена. Детали этого механизма и описание процесса настройки можно найти на странице `google.com/a` или наберите в поисковой строке системы Google запрос "hosted Gmail".

Служба, размещенная на удаленном сервере, решает массу проблем, включая хранение, управление сервером, обновление программного обеспечения, конфигурирование, фильтрацию спама, резервное копирование и обеспечение безопасности. Как плата за услуги "бесплатных" служб, вы, возможно, будете вынуждены получать рекламные объявления и пожертвовать конфиденциальностью. Во многих ситуациях это хороший выбор; если служба, расположенная на удаленном сервере, вас устраивает, то остальную часть это огромной главы можно не читать.

Однако служба электронной почты, расположенная на удаленном сервере, устраивает не всех. Коммерческие и другие крупные организации, работа которых зависит от электронной почты, не желают рисковать, размещая службу электронной почты за пределами своего офиса. Такие организации могут иметь множество причин для размещения службы электронной почты на своих серверах, включая вопросы безопасности, производительности и доступности. Эта глава предназначена именно для них.

Огромный объем этой главы свидетельствует о том, что почтовые системы — достаточно сложная тема. Здесь изложены только основные сведения. Схематическое описание главы представлено в табл. 20.1.

**Таблица 20.1. Структура главы**

| Описание | Раздел |
|---|---|
| Основы | 20.1 |
| Вопросы проектирования почтовых систем | 20.4 |
| Спам и вредоносные программы | 20.6 |
| Фильтрация вирусов и спама с помощью интерфейса amavisd | 20.6 |
| Настройка программы **sendmail** | 20.8 |
| Настройка агента exim | 20.14 |
| Настройка агента postfix | 20.15 |
| Технология DKIM | 20.16 |
| Интегрированные системы электронной почты | 20.17 |

# 20.1. Системы электронной почты

Теоретически система электронной почты состоит из нескольких компонентов.

- *Пользовательский агент ("mail user agent"— MUA или UA)*, который дает пользователям возможность читать и составлять сообщения.

- *Агент передачи электронной почты (mail submission agent — MSA), принимающий почту, исходящую от агента MUA, обрабатывающий ее и передающий транспортной системе.*
- *Транспортный агент* (mail transport agent — MTA), который пересылает сообщения с одного компьютера на другой.
- *Агент доставки* (delivery agent — DA), который помещает сообщения в локальное хранилище[2].
- Необязательный *агент доступа* (acess agent — AA), который связывает пользовательский агент с хранилищем сообщений (например, посредством протокола IMAP или POP).

К некоторым из этих агентов присоединяются инструменты для распознания спама, вирусов и (исходящих) внутренних секретов компании. Схема взаимодействия всех указанных компонентов представлена на рис. 20.1.

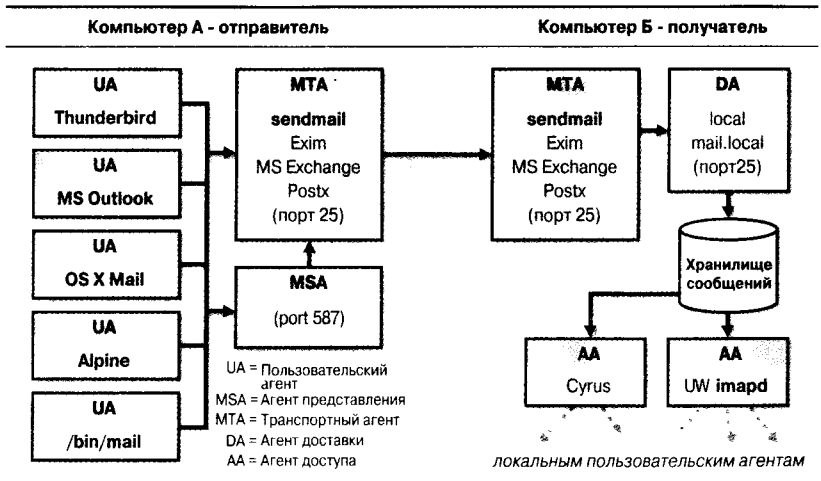



*Рис. 20.1. Компоненты почтовой системы*

## Пользовательские агенты

Пользовательский агент (иногда говорят "клиент электронной почты") применяется для чтения и составления электронных сообщений. Первоначально сообщения могли содержать только простой текст, однако благодаря стандарту MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения электронной почты в сети Интернет) появилась возможность включать в них форматированный текст и присоединять разные файлы (в том числе вирусы). Стандарт MIME поддерживается большинством пользовательских агентов. Поскольку он не влияет на процесс адресации и доставки почты, мы не будем его рассматривать.

Самым первым пользовательским агентом была утилита `/bin/mail`, которая и сейчас остается удобным инструментом для чтения текстовых сообщений. Поскольку электронная почта в Интернете давно вышла за пределы "текстовой эры", пользовательские агенты, ориентированные на текст, для большинства пользователей являются непрак-

[2] Иногда это почтовые ящики пользователей, иногда — база данных.

тичными. Но не следует отбрасывать утилиту **/bin/mail**; она остается удобным интерфейсом для сценариев и других программ. (Один "завзятый" пользователь системы Linux каждую ночь посылает сообщение электронной почты от демона **cron** программе-планировщику, чтобы при первом взгляде на календарь становился ясным статус всех его программ. По умолчанию демон **cron** использует утилиту **/bin/mail** для уведомления пользователя в ситуациях, когда он не может выполнить задание, указанное в расписании.)

Одна из элегантных функциональных возможностей, проиллюстрированных на рис. 20.1, состоит в том, что пользовательский агент не обязан выполняться в той же самой системе и даже на той же самой платформе, на которой выполняется остальная часть почтовой системы. Пользователи могут получить свои сообщения электронной почты на ноутбуках или смартфонах, работающих под управлением системы Windows, с помощью протоколов агентов доступа IMAP или POP.

Страница `http://en.wikipedia.org/wiki/Comparison_of_e-mail_clients`[3] содержит подробный список многих клиентов электронной почты, операционных систем, в которых они выполняются, и функциональные возможности, которые они поддерживают. К популярным клиентам относятся Thunderbird, Alpine, Zimbra и, разумеется, Microsoft Outlook. Страница `http://en.wikipedia.org/wiki/Comparison_of_webmail_providers` содержит информацию о таких веб-службах, как Gmail, Hormail и Yahoo! Mail.

## Агенты представления

Агенты MSA — последнее новшество в пантеоне электронной почты — были изобретены для того, чтобы разгрузить агентов MTA от некоторых задач. Агенты MSA облегчают серверам-концентраторам задачу различения входящих и исходящих сообщений (например, для принятия решения об ответе) и обеспечивают пользовательским агентам единообразную и простую конфигурацию исходящей почты.

Агент MSA напоминает секретаря, который ведет прием новых сообщений и внедрен в систему локальными пользовательскими агентами. Агент MSA располагается между пользовательским и транспортным агентами и выполняет несколько функций, которые ранее относились к компетенции агента MTA. Агент MSA реализует безопасную (зашифрованную и аутентифицированную) связь с пользовательскими агентами и часто немного изменяет заголовки и удаляет входящие сообщения. Во многих ситуациях агент MSA просто прослушивает агентов MTA на разных портах, применяя разные конфигурации.

Агенты MSA используют тот же протокол передачи почты, что и агенты MTA, поэтому с точки зрения пользовательских агентов они выглядят как агенты MTA. Однако они обычно прослушивают соединение на порту 587, а не 25, который является стандартным для агентов MTA. При такой схеме работы пользовательские агенты должны соединяться с портом 587, а не 25. Если ваш пользовательский агент не может использовать порт 587, вы можете запустить агента MSA на порту 25, но в системе, которая отличается от системы, в которой был запущен агент MTA; в каждый момент времени на конкретном порту можно прослушивать только один процесс.

Агент MSA может помочь решить несколько задач, возникающих из-за спама. Для рассылки большого количества спама используются инфицированные домашние персональные компьютеры. В результате многие интернет-провайдеры, обслуживающие домашние компьютеры, либо блокируют исходящее соединение на порту 25, либо требу-

[3] Актуальность ссылок на момент выхода русского издания не гарантируется. — *Примеч. ред.*

ют верификации учетной записи в процессе обмена информацией по протоколу SMTP. Домашний персональный компьютер может использовать собственный почтовый сервер интернет-провайдера для исходящих сообщений, но некоторые современные механизмы фильтрации спама, такие как протоколы SPF (раздел 20.6) и DKIM (20.16), требуют, чтобы почта, отправляемая организацией, действительно создавалась именно указанной организацией.

Если вы используете агент MSA, проверьте, что конфигурация вашего транспортного агента задана так, что он не дублирует работу, выполняемую агентом MSA. Обработка дубликатов никак не влияет на корректность обработки почты, но заставляет систему выполнять дополнительную ненужную работу.

Поскольку агент MSA для передачи сообщений использует агент MTA, для взаимной аутентификации оба агента должны использовать протокол SMTP-AUTH. В противном случае возникнет так называемая "открытая передача" (open relay), которую могут использовать спамеры, и другие сайты занесут ваш адрес в "черный список".

## Транспортные агенты

Задача транспортного агента — принимать почту от пользовательского агента или агента представления, интерпретировать адреса получателей и перенаправлять почту на соответствующие компьютеры для последующей доставки. Транспортные агенты работают по протоколу SMTP (Simple Mail Transport Protocol — простой протокол передачи электронной почты), который вначале был определен в документе RFC821, а затем дополнен в документе RFC5321. Расширенная версия этого протокола называется ESMTP (Extended SMTP).

Список заданий транспортных агентов как отправителя, так и получателя содержит следующие пункты.

- Получение сообщений электронной почты от удаленных почтовых серверов.
- Распознавание адресов получателей.
- Перезапись адресов получателей в форме, понятной для агента доставки.
- Перенаправление сообщения следующему ответственному серверу или передача его локальному агенту доставки для сохранения в почтовом ящике пользователя.

Большая часть работы, связанной с настройкой почтовой системы, сводится к конфигурированию транспортного агента. В книге мы рассматриваем три агента MSA с открытым кодом: **sendmail**, Exim и Postfix.

## Локальные агенты доставки

Агент доставки, которого иногда называют также локальным агентом доставки (local delivery agent — LDA), отвечает за прием почты от транспортного агента и ее передачу соответствующим получателям на локальном компьютере. Почта может доставляться конкретному пользователю, в список рассылки, в файл и даже в программу. Однако два последних получателя могут ослабить конфиденциальность и безопасность вашей системы.

Транспортные агенты обычно содержат встроенных локальных агентов доставки. Например, программы **procmail** (`procmail.org`) и Maildrop (`courier-mta.org/maildrop`) являются локальными агентами доставки, которые способны фильтровать и сортировать почту перед ее доставкой. Некоторые агенты доступа (AA) также имеют встроенных агентов доставки, выполняющих как доставку, так и другие задания.

## Хранилища сообщений

Хранилище сообщений — пункт назначения, находящийся в конце долгого пути, который сообщение электронной почты проходит по Интернету от отправителя к получателю.

Почта обычно хранится в формате **mbox** или **Maildir**. В первом случае вся почта хранится в одном файле, как правило, **/var/mail/***имя_пользователя*, а индивидуальные сообщения отделяются специальной линией From. Во втором варианте каждое сообщение хранится в отдельном файле. Хранить сообщения в отдельных файлах намного удобнее, но при этом в каталоге находится очень много маленьких файлов; некоторые файловые системы этого не одобряют.

Эти простые файлы до сих пор используются в качестве хранилищ сообщений, но интернет-провайдеры, имеющие тысячи и миллионы клиентов электронной почты, ищут другие технологии для реализации своих хранилищ, как правило, на основе баз данных. Хранилища сообщений становятся все более сложными.

## Агенты доступа

Для доступа к хранилищам и загрузки сообщений электронной почты на локальное устройство (рабочую станцию, ноутбук, телефон и т.п.) используется два протокола: IMAP4 и POP. Ранние версии этих протоколов имели проблемы с безопасностью. Убедитесь, что вы используете версию (IMAPS или POP3S), которая реализует шифрование SSL и не передает через Интернет пароли в открытом виде.

Мы предпочитаем протокол IMAP (Internet Message Access Protocol — протокол доступа к электронной почте Интернета). Он лучше, чем протокол POP, поскольку доставляет ваши почтовые сообщения по одному, а не все сразу, что более удобно для работы в сети (особенно если линии связи не обладают высокой пропускной способностью) и комфортнее для людей, перемещающихся с места на место. Протокол IMAP также лучше обрабатывает огромные файлы, которые многие любят присоединять к своим сообщениям: вы можете просматривать заголовки своих сообщений и не загружать вложенные файлы, пока не будете готовы к работе с ними.

Протокол IMAP управляет почтовыми каталогами на нескольких сайтах; например, на почтовом сервере и на вашем персональном компьютере. Почта, размещенная на почтовом сервере, может стать частью процедуры резервного копирования. Страница Википедии, посвященная протоколу IMAP, содержит много информации о его доступных реализациях.

Протокол POP (Post Office Protocol — протокол почтового отделения) аналогичен протоколу IMAP, но основан на модели, в которой все почтовые сообщения загружаются с сервера на клиентский компьютер. При этом почта может быть либо удалена с сервера (в этом случае ее невозможно впоследствии восстановить), либо сохранена на нем (в этом случае буферный файл почты становится все больше и больше). Парадигма "вся почта за один раз" неудобна для работы в сети и некомфортна для пользователя. Если пользователь никогда не выбрасывает ненужного старья, то передача почты может стать очень медленной, а буферный файл достигнет огромных размеров.

Серверы IMAP/POP имеют несколько реализаций: Courier, Cyrus IMAP, Dovecot, **imapd** университета штата Вашингтон (Сиэтл) и Zimbra. Мы предпочитаем серверы Dovecot и Zimbra[4]. Протоколы IMAP и POP поддерживают практически все почтовые пользовательские агенты.

[4] Zimbra — это не просто агент доступа, а, скорее, полноценная промышленная почтовая система (см. раздел 20.17).

## Так много компонентов, так мало времени

Если почтовая система состоит из множества компонентов (а мы еще даже не упоминали о сканировании спама и вирусов!), то ее архитектура, скорее всего, является слишком сложной. Однако в небольших организациях транспортные агенты могут адсорбировать функции агентов представления и локальных агентов доставки, что позволяет упростить почтовую систему. Более крупные организации могут пожелать хранить все компоненты по отдельности и выполнять несколько их экземпляров одновременно, чтобы обеспечить распределенную загрузку. На самом деле почтовая система может быть настолько же сложной, насколько и простой, в зависимости от вашего желания. Проблемы ее проектирования рассматриваются в разделе 20.4.

# 20.2. Структура почтового сообщения

Структура электронного сообщения состоит из трех частей.

- Конверт
- Заголовки
- Тело

Конверт определяет, куда должно быть доставлено сообщение или куда его требуется возвратить в случае, если доставка невозможна. Обычно конверт невидим для пользователя и не является частью самого сообщения; он используется транспортным агентом.

Если отправителем и получателем являются обычные люди, то адреса конверта обычно согласованы со строками From и To заголовка. Если же сообщение направлено списку рассылки или было сгенерировано спамером, пытающимся подделать идентичность отправителя, то конверт и заголовки могут быть не согласованы друг с другом.

Заголовки — это набор пар "свойство/значение", отформатированных в соответствии с документом RFC5322. В них содержится различная информация о сообщении, включая дату и время его отправки, а также сведения о транспортных агентах, через которые оно прошло на своем пути. Заголовки являются неотъемлемой частью сообщения, но пользовательские агенты часто скрывают некоторые менее интересные заголовки при отображении сообщения.

Тело сообщения — это та информация, которую, собственно, и требуется переслать. Оно должно содержать обычный текст, однако часто он представляет собой двоичные данные в специальной почтовой кодировке.

## Заголовки почтовых сообщений

Системный администратор должен уметь анализировать заголовки почтовых сообщений и выявлять с их помощью возникшие проблемы. Многие пользовательские агенты скрывают заголовки, но существует способ их увидеть, открыв хранилище сообщений в текстовом редакторе. Ниже приведены заголовки (с сокращениями, обозначенными многоточиями) из обычного сообщения, которое не является спамом. Мы удалили половину страницы заголовков, которые система Gmail использует для фильтрации спама.

```
Delivered-To: sailingevi@gmail.com
Received: by 10.231.39.205 with SMTP id…; Fri, 16 Oct 2009 08:14:27 -700 (PDT)
Received: by 10.114.163.26 with SMTP id…; Fri, 16 Oct 2009 08:14:26 -700 (PDT)
Return-Path: <david@schweikert.ch>
Received: from mail-relay.atrust.com (mail-relay.atrust.com [63.173.189.2])
```

```
    by mx.google.com with ESMTP id 17si2166978pxi.34.2009.10.16.08.14.20;
    Fri, 16 Oct 2009 08:14:25 -0700 (PDT)
Received-SPF: fail (google.com: domain of david@schweikert.ch does not
    designate 63.173.189.2 as permitted sender) client-ip=63.173.189.2;
Authentication-Results: mx.google.com; spf=hardfail (google.com: domain of
    david@schweikert.ch does not designate 63.173.189.2 as permitted sender)
    smtp.mail=david@schweikert.ch
Received: from mail.schweikert.ch (nigel.schweikert.ch [88.198.52.145])
    by mail-relay.atrust.com (8.12.11/8.12.11) with ESMTP id n9GFEDKA029250
    for <evi@atrust.com>; Fri, 16 Oct 2009 09:14:14 -0600
Received: from localhost (localhost.localdomain [127.0.0.1])
    by mail.schweikert.ch (Postfix) with ESMTP id 3251112DA79;
    Fri, 16 Oct 2009 17:14:12 +0200 (CEST)
X-Virus-Scanned: Debian amavisd-new at mail.schweikert.ch
Received: from mail.schweikert.ch ([127.0.0.1])
    by localhost (mail.schweikert.ch [127.0.0.1]) (amavisd-new, port 10024)
    with ESMTP id dV8BpT7rhJKC; Fri, 16 Oct 2009 17:14:07 +0200 (CEST)
Received: by mail.schweikert.ch (Postfix, from userid 1000)
    id 2A15612DB89; Fri, 16 Oct 2009 17:14:07 +0200 (CEST)
Date: Fri, 16 Oct 2009 17:14:06 +0200
From: David Schweikert <david@schweikert.ch>
To: evi@atrust.com
Cc: Garth Snyder <garth@garthsnyder.com>
Subject: Email chapter comments
```

Для того чтобы прочитать эту тарабарщину, начнем со строк Received, но в направлении снизу вверх (т.е. со стороны отправителя). Это сообщение пришло с домашнего компьютера Дэвида Швайкерта (David Schweikert), находящегося в домене `schweikert.ch`, на его почтовый сервер (`mail.schweikert.ch`) , где оно прошло сканирование на вирусы. Затем оно было переслано получателю по адресу `evi@atrust.com`. Однако получатель `mail-relay.atrust.com` послал его по адресу `sailingevi@gmail.com`, где находится почтовый ящик Эви.

🕮 Более подробная информация о технологии SPF приведена в разделе 20.6.

Посреди заголовков мы видим, что проверка сообщения на основе технологии SPF завершилась неудачно. Это произошло потому, что система Google проверила IP-адрес сервера `mail-relay.atrust.com`, сравнила его с записью SPF в домене `schweirkert.ch` и они не совпали. Это недостаток технологии SPF — она не распознает сообщения, которые предназначены для пересылки.

Мы можем видеть следы частого использования транспортных агентов (Postfix — в домене `schweirkert.ch` и **sendmail** 8.12 — в домене `atrust.com`). В данном случае сканирование вирусов было выполнено с помощью агента **amavisd-new** на порту 10024 на компьютере, работающем под управлением операционной системы Debian Linux. Мы можем проследить путь, который проделало сообщение из часового пояса центральноевропейского летнего времени (`CEST +0200`) в Колорадо (`-0600`) и на сервер Google (`PDT -0700`); эти числа представляют собой разницу между местным временем и всемирным координированным временем (UTC — Coordinated Universal Time). В заголовках можно найти много скрытой информации!

Рассмотрим заголовки (тоже сокращенные) сообщения, которое является спамом.

```
Delivered-To: sailingevi@gmail.com
Received: by 10.231.39.205 with SMTP id…; Mon, 19 Oct 2009 08:59:32 -0700…
Received: by 10.231.5.143 with SMTP id…; Mon, 19 Oct 2009 08:59:31 -0700…
```

```
Return-Path: <smotheringl39@sherman.dp.ua>
Received: from mail-relay.atrust.com (mail-relay.atrust.com [63.173.189.2]) …
Received-SPF: neutral (google.com: 63.173.189.2 is neither permitted nor denied
  by best guess record for domain of smotheringl39@sherman.dp.ua) clientip=
  63.173.189.2;
Authentication-Results: mx.google.com; spf=neutral (google.com: 63.173.189.2 is
  neither permitted nor denied by best guess record for domain of
  smotheringl39@sherman.dp.ua) smtp.mail=smotheringl39@sherman.dp.ua
Received: from SpeedTouch.lan (187-10-167-249.dsl.telesp.net.br
  [187.10.167.249] (may be forged)) by mail-relay.atrust.com …
Received: from 187.10.167.249 by relay2.trifle.net; Mon, 19 Oct 2009 13:59: …
From: "alert@atrust.com" <alert@atrust.com>
To: <ned@atrust.com>
Subject: A new settings file for the ned@atrust.com mailbox
Date: Mon, 19 Oct 2009 13:59:12 -0300 …
```

В соответствии с заголовком `From`, отправителем этого сообщения является почтовый ящик `alert@atrust.com`. Однако заголовок `Return-Path`, содержащий копию конверта отправителя, свидетельствует о том, что источником сообщения является почтовый ящик `smotheringl39@sherman.dp.ua`, адрес которого указывает на Украину. Первый транспортный агент, обработавший сообщение, имеет IP-адрес 187.10.167.249 и находится в Бразилии. Хитрые спамеры ...[5]

Проверка по технологии SPF, которую выполнил сервер Google, снова завершилась провалом, но на этот раз с "нейтральным" результатом, потому что домен `sherman.dp.ua` не имеет записи SPF, с которой можно было бы сравнить IP-адрес сервера `mail-relay.atrust.com`.

Информация о получателе также не совсем правдива. Заголовок To указывает, что сообщение направлено по адресу `ned@atrust.com`. Однако для того, чтобы сообщение было направлено по адресу `sailingevi@gmail.com`, среди адресов получателя на конверте должен находиться адрес `evi@atrust.com`.

# 20.3. Протокол SMTP

Протокол SMTP (Simple Mail Transport Protocol — простой протокол передачи электронной почты) и его расширенная версия ESMTP были стандартизованы в документах RFC (в частности, RFC5321) и используются для передачи сообщений между разными частями почтовой системы.

- От пользовательского агента к агенту представления или транспортному агенту, когда сообщение поступает в почтовую систему.
- От агента представления к транспортному агенту, когда сообщение начинает свой путь.
- От транспортного агента или агента представления к программам сканирования спама и вирусов.
- От одного транспортного агента к другому при передаче сообщений от одной организации другой.
- От транспортного агента к агенту доставки, когда сообщение поступает в локальное хранилище.

[5] Следует отметить, что многие строки заголовка, включая строки Received, могут оказаться поддельными. С этими данными необходимо быть крайне осторожными.

Поскольку формат сообщений и протокол передачи стандартизированы, транспортные агенты отправителя и получателя не обязательно должны быть одинаковыми и даже знать друг о друге; просто они оба должны придерживаться протоколов SMTP и ESMTP. На разных почтовых серверах могут работать различные транспортные агенты, и их взаимодействие будет безошибочным.

В соответствии со своим именем, протокол SMTP является довольно простым. Транспортный агент связывается с вашим почтовым сервером и, по существу, сообщает следующее: "Вот сообщение; пожалуйста, доставь его по адресу `user@your.domain`". Ваш транспортный агент отвечает: "Хорошо".

Требование строго придерживаться протокола SMTP стало основой для борьбы со спамом и вредоносными программами, поэтому системные администраторы должны хорошо в нем разбираться. Его язык имеет всего несколько команд; самые важные перечислены в табл. 20.2.

**Таблица 20.2. Команды протокола SMTP**

| Команда | Функция |
|---|---|
| `HELO` имя_компьютера | Проверяет, поддерживает ли компьютер протокол SMTP |
| `EHLO` имя_компьютера | Проверяет, поддерживает ли компьютер протокол ESMTP |
| `MAIL FROM:` обратный_адрес | Идентифицирует конверт отправителя |
| `RCPT TO:` прямой_адрес[a] | Идентифицирует конверт получателя |
| `VRFY` адрес | Проверяет корректность адреса (возможность доставки) |
| `EXPN` адрес | Демонстрирует расширение альтернативных имен и отображения файла **.forward** |
| `DATA` | Отмечает начало тела сообщения[б] |
| `QUIT` | Завершает сообщение и прерывает соединение |
| `RSET` | Восстанавливает состояние соединения |
| `HELP` | Выводит на экран описание команд SMTP |

[a] У сообщения может быть несколько команд RCPT.

[б] Тело завершается точкой в строке.

## Вы прислали мне привет

Серверы, использующие протокол ESMTP, начинают общение, посылая команду EHLO, а не HELO. Если процесс на другом конце соединения понимает эту команду и отвечает "OK", то участники согласовывают расширения и находят наименьший общий знаменатель для обмена. Если в ответ на команду EHLO собеседник присылает ошибку, то сервер, использующий протокол ESMTP, переключается на протокол SMTP. Однако в настоящее время практически все серверы используют протокол ESMTP.

Типичное общение по протоколу SMTP для доставки сообщения состоит из следующих команд: HELO или EHLO, MAIL FROM:, RCPT TO:, DATA и QUIT. Большая часть команды выполняется отправителем. Получатель лишь высылает коды ошибок и подтверждения.

Протоколы SMTP и ESMTP основаны на тексте, поэтому их можно использовать непосредственно в процессе отладки почтовой системы. Достаточно просто применить утилиту **`telnet`** к TCP-порту 25 или 587 и начать ввод команд SMTP. Пример приведен в разделе 20.15.

## Коды ошибок протокола SMTP

Кроме протокола SMTP, в документах RFC определена совокупность кодов временных и постоянных ошибок. Изначально эти коды состояли из трех цифр (например, 550), каждая из которых интерпретировалась отдельно. Если первая цифра равнялась 2, значит, все прошло успешно, если 4 — произошла временная ошибка, а если 5 — постоянная ошибка.

Трехзначная система кодирования ошибок слишком жесткая, поэтому в документе RFC3463 была описана ее гибкая модификация. В этом документе определен расширенный формат кодирования ошибок, получивший название уведомление о состоянии доставки, или DSN (delivery status notification). Коды DNS имеют формат X.X.X, а не XXX, как раньше, причем каждый символ X может означать многозначное число. Первое число X по-прежнему может принимать значения 2, 4 и 5. Второе число означает тему, а третье кодирует подробности. В новой системе кодирования второе число используется для того, чтобы отличать ошибки компьютера от ошибок почтового ящика. Некоторые коды DSN перечислены в табл. 20.3. Все коды перечислены в приложении А к документу RFC3463.

**Таблица 20.3. Команды протокола SMTP**

| Временная ошибка | Постоянная ошибка | Смысл |
|---|---|---|
| 4.2.1 | 5.2.1 | Почтовый ящик недоступен |
| 4.2.2 | 5.2.2 | Почтовый ящик полон |
| 4.2.3 | 5.2.3 | Слишком длинное сообщение |
| 4.4.1 | 5.4.1 | Нет ответа от компьютера |
| 4.4.4 | 5.4.4 | Невозможно выполнить маршрутизацию |
| 4.5.3 | 5.5.3 | Слишком много получателей |
| 4.7.1 | 5.7.1 | Доставка не авторизована, сообщение отклонено |
| 4.7.* | 5.7.* | Нарушение правил организации |

## Аутентификация SMTP

Документ RFC4954 определяет расширение исходного протокола SMTP, позволяющее SMTP-клиенту идентифицировать себя и проходить аутентификацию у почтового сервера. После этого сервер может позволить клиенту использовать себя для пересылки почты. Этот протокол поддерживает несколько механизмов аутентификации. Обмен информацией состоит из следующих этапов.

- Клиент посылает команду EHLO, сообщая, что он использует протокол ESMT.
- Сервер отвечает и уведомляет о своих механизмах аутентификации.
- Клиент посылает команду AUTH и называет конкретный механизм аутентификации, который он хочет использовать, включая свои данные для аутентификации.
- Сервер принимает данные, присланные с командой AUTH, или начинает последовательность команд "вызов/ответ" для обмена информацией с клиентом.
- Сервер либо принимает, либо отвергает попытку аутентификации.

Для того чтобы узнать, какой механизм аутентификации поддерживает сервер, можно применить утилиту `telnet` к порту 25 и выполнить команду EHLO. Например, ниже

приведен усеченный вариант обмена сообщениями с почтовым сервером mail-relay.atrust.com (команды набраны полужирным шрифтом).

```
solaris$ telnet mail-relay.atrust.com 25
Trying 192.168.2.10…
Connected to mail-relay.atrust.com.
Escape character is '^]'.
220 mail-relay.atrust.com ESMTP ATE Mail Service 24.1.2/24.1.2; Tue, 20 Oct
    2009 14:28:53 -0600
ehlo solaris.booklab.atrust.com
250-mail-relay.atrust.com Hello solaris.booklab.atrust.com, pleased to meet
    you
250-ENHANCEDSTATUSCODES
250-AUTH LOGIN PLAIN
…
250 HELP
quit
221 2.0.0 mail-relay.atrust.com closing connection
```

В данном случае почтовый сервер поддерживает механизмы аутентификации LOGIN и PLAIN. Серверы **sendmail**, Exim и Postfix поддерживают аутентификацию SMTP; детали их конфигураций описаны в разделе 20.11, 20.14 и 20.15.

# 20.4. Принципы организации электронной почты

Соблюдение принципов работы с электронной почтой, описанных в этой главе, является обязательным условием нормального администрирования почтовых систем в средних и крупных организациях. Естественно, ими можно руководствоваться и в небольших организациях. Есть несколько основных факторов, которые позволяют значительно облегчить решение административных задач.

- Наличие серверов для входящей и исходящей почты или, в случае крупных организаций, иерархии серверов.
- Фильтрация спама и вирусов прежде, чем разрешить доставку сообщения в вашу организацию.
- Фильтрация спама, вирусов и утечки данных перед отправкой сообщений во внешний мир.
- Для загруженных сайтов наличие резервного копирования, выполняемого агентами MTA для исходящих сообщений, не отправленных с первой попытки.
- Ведение журнала и архива в соответствии с законом (например, для проведения расследований).
- Наличие почтового каталога для каждого пользователя в организации.
- Поддержка протоколов IMAP или POP для интеграции персональных компьютеров, систем Macintosh и удаленных клиентов.

Каждый фактор будет подробно проанализирован ниже. Кроме того, необходимо добиться хорошего взаимодействия и функционирования других подсистем и средств: записи MX базы данных DNS должны быть правильными, брандмауэры должны пропускать входящую и исходящую почту, должны быть определены компьютеры, на которых организованы хранилища сообщений, и т.д.

🕮 Подробнее о записях MX рассказывалось в разделе 17.8.

Почтовые серверы решают пять задач.

- Принимают исходящую почту от агентов представления или пользовательских агентов.
- Принимают входящую почту из внешнего мира.
- Фильтруют сообщения от спама, вирусов и других вредоносных программ.
- Доставляют сообщения в почтовые ящики пользователей.
- Позволяют пользователям получать доступ к почтовым ящикам посредством протокола IMAP или POP.

В небольшой организации серверы, реализующие указанные функции, могут работать на одном компьютере. В крупных сетях это должны быть отдельные компьютеры. Настройка брандмауэра значительно упростится, если входящая почта будет поступать на один компьютер и вся исходящая почта тоже будет проходить через один компьютер. Кроме того, реалии современного Интернета вынуждают почтовые серверы заниматься сканированием содержимого сообщений.

## Почтовые серверы

Существует два типа почтовых серверов: серверы, взаимодействующие с Интернетом для обработки входящих и исходящих сообщений, и внутренние серверы, взаимодействующие с пользователями. Ниже мы опишем структуру почтовой системы, которая, как нам кажется, является легко масштабируемой, хорошо управляемой и безопасной. Задачи управления входящей и исходящей почтой в ней решают специально выделенные для этих целей серверы. Один из вариантов такой структуры приведен на рис. 20.2.

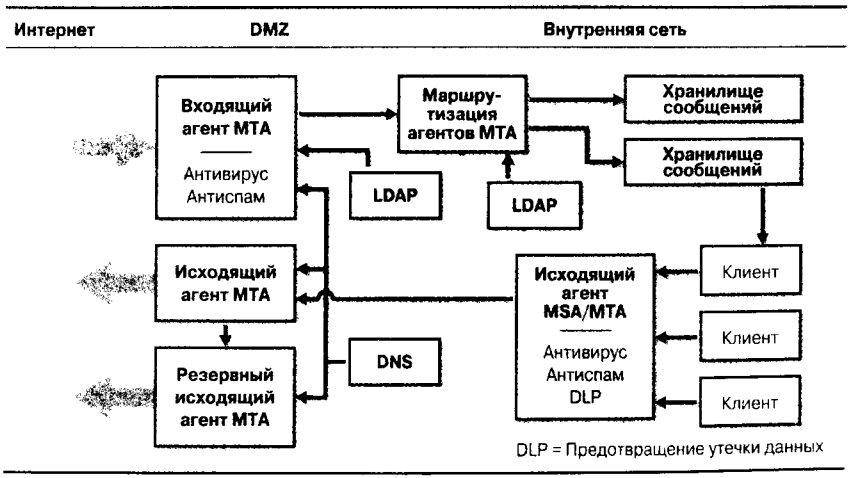



*Рис. 20.2. Структура почтовой системы (вариант 1)*

Почтовая система, изображенная на рис. 20.2, состоит из двух частей: DMZ (демилитаризованной зоны), компьютеры которой подключены непосредственно к Интернету, и внутренней зоны, отделенной от зоны DMZ и Интернета с помощью брандмауэра. В зоне DMZ расположено несколько серверов.

- Транспортный агент, прослушивающий порт 25 и передающий входящую почту на фильтры.

- Фильтры вирусов и спама, отвергающие или отправляющие "в карантин" опасные сообщения.
- Копия базы данных LDAP (Lightweight Directory Access Protocol — облегченный протокол доступа к каталогам), содержащая исходящую информацию почтовой системы.
- Исходящий транспортный агент, пытающийся доставить почту, отправленную агентом представления.
- Резервный исходящий агент MTA для исходящих сообщений, не отправленных с первой попытки.
- Кеширующий сервер DNS, используемый исходящим агентом MTA для просмотра записей MX и входящим агентом MTA — для просмотра "черного списка" (доменов отправителей), а также криптографической информации для подписанных сообщений.

Сервер исходящей почты, который непосредственно подключен к Интернету, наиболее уязвим. Он должен быть хорошо защищен, иметь мало пользователей и не выполнять посторонних процессов или услуг. Каждое сообщение, которое он обрабатывает, должно быть проверено, чтобы выполнялись следующие условия.

- Организация отправителя не занесена в черный список.
- Запись SPF отправителя является корректной.
- Местный получатель является корректным.
- Если сообщение подписано, можно проверить его подпись DKIM.
- Сообщение не содержит вредоносных программ.
- Сообщение не является спамом.

Все эти задачи, связанные со сканированием сообщений, можно выполнить внутри агента MTA или с помощью отдельного пакета, например **amavisd-new**. Сканирование спама и вредоносных программ описывается в разделе 20.6.

Сервер исходящей почты должен не менее хорошо контролироваться. Если в организации используются большие списки рассылки, то резервный агент MTA может повысить общую производительность работы, изолировав проблему, связанную с получателем, и решив ее отдельно. Мы предполагаем, что фильтрация и сканирование исходящей почты выполняются агентом MTA во внутренней зоне.

К серверам внутренней зоны относятся следующие агенты.

- Транспортный агент внутренней маршрутизации, выполняющий маршрутизацию принятой почты в хранилище сообщений.
- Исходная база данных LDAP, содержащая информацию о маршрутизации почты.
- Агенты представления или транспортные агенты для исходящей почты.
- Фильтры вирусов, спама и механизм предотвращения утечки данных (DLP — data leak prevention).

Исходящая почта должна сканироваться на вирусы и спам, чтобы убедиться, что локальные компьютеры не инфицированы, и ограничить распространение вредоносных программ среди других организаций. Если ваша организация обеспокоена вопросами утечки конфиденциальной или секретной информации (например, номеров кредитных карточек или карточек социального страхования), то фильтрация DLP должна быть выполнена во внутреннем агенте представления до того, как сообщение достигнет исходящего транспортного агента в более уязвимой зоне DMZ.

Большинство современных систем фильтрации DLP встроено в коммерческие продукты для работы в сети веб и обработки электронной почты (например, ClearEmail, IronPort компании Cisco, WebSense, Content Control и так далее) и "раскручено" с помощью агрессивной рекламы. Некоторые из этих программ содержат механизмы для перекодировки номеров карточек социального страхования и кредитных карточек, а также слов и фраз, которые можно задать заранее. Сканирование DLP пока находится в первичном состоянии и имеет определенные юридические последствия. Следует убедить сотрудников и заключить с ними соглашение о том, что вы имеете право сканировать входящую и исходящую почту в поисках спама, вредоносных программ и конфиденциальной информации.

В конце пути, который проходит электронное сообщение, находятся пользователи, работающие во внутренней зоне и имеющие доступ как к хранилищу сообщений (для отправки входящей почты), так и к агенту представления (для посылки исходящей почты). Эти пользователи могут быть удаленными. В этом случае они должны использовать команду SMTP-AUTH, чтобы аутентифицировать себя.

При необходимости почтовые серверы для входящих и исходящих сообщений можно реплицировать. Например, несколько серверов для входящих сообщений можно скрыть за блоком балансировки загрузки или можно использовать записи DNS MX, чтобы примерно сбалансировать загрузку. Разные клиентские машины могут направлять почту через разные серверы исходящих сообщений.

И наоборот, организации с небольшой загрузкой могут разумно сочетать серверы входящих и исходящих сообщений. Некоторые механизмы, такие как BATV или отражатель Pen-pals, легче реализовать на одном сервере. BATV (bounce address tag valifation — проверка обратного адреса) — это схема, определяющая, истинным или поддельным является обратный адрес. Он не дает отражателю электронной почты отправлять сообщения по поддельным адресам. Pen-pals (часть утилиты **amavisd-new**) — это схема, уменьшающая вероятность того, что письмо является спамом, если его отправитель ранее уже отвечал на сообщения, посланные одним из ваших пользователей.

🕮 Обсуждение механизмов распределения файлов приведено в разделе 19.2.

Большинство компьютеров вашей организации могут использовать минимальную конфигурацию агентов MSA/MTA, которые пересылают всю исходящую почту для обработки на сервер с более сложными механизмами. Они не обязаны принимать почту из Интернета и могут использовать одну и ту же конфигурацию. Эту конфигурацию можно распространить с помощью команд **rdist** или **rsync**.

Организации, использующие такие программы, как Microsoft Exchange или Lotus Notes, но не желающие подсоединять их непосредственно к Интернету, могут использовать схему, описанную выше, в которой программа Exchange играет роль маршрутизатора во внутренней зоне.

Независимо от выбранной схемы организации почтовой системы, убедитесь, что ваша конфигурация агента MTA, записи DNS MX и правила брандмауэра соответствуют одним и тем же правилам обработки почты.

## 20.5. Почтовые псевдонимы

Псевдонимы позволяют системному администратору или отдельным пользователям переадресовывать почту[6]. Их можно применять для задания списков рассылки, для пересылки почты между компьютерами, а также для того, чтобы к пользователям можно было обращаться по нескольким именам. Псевдонимы обрабатываются рекурсивно, поэтому могут указывать на объекты, которые, в свою очередь, тоже являются псевдонимами.

Системные администраторы часто используют роли или функциональные псевдонимы (например, `printers@example.com`), чтобы направлять сообщения определенной темы именно тому пользователю, который ею сейчас занимается. Кроме того, псевдонимы используются для пересылки результатов ночного сканирования на вирусы и для указания администратора, отвечающего за электронную почту.

Почтовые системы поддерживают несколько механизмов поддержки псевдонимов.

- Отображения простых файлов, сгенерированных из файла **`/etc/mail/alias`**.
- Различные маршрутные почтовые базы данных, связанные с конкретным транспортным агентом.
- Базы данных LDAP.
- Другие коллективные механизмы, такие как протокол NIS.

🕮 Протокол LDAP описан в главе 19.

Простые файлы, такие как **`/etc/mail/alises`** (см. ниже), — самый простой способ задать псевдоним в малых и средних организациях. Если вы хотите использовать концепцию почтового дома (mail home) и обслуживаете крупную организацию со сложной структурой, мы рекомендуем реализовать почтовый дом, сохраняя псевдонимы на сервере LDAP.

Большинство пользовательских агентов в некоторой степени поддерживают концепцию псевдонимов (обычно называемых "моя группа", "мои письма" или как-то еще). Однако пользовательский агент раскрывает такие псевдонимы еще до того, как почта достигнет агента представления или транспортного агента. Эти псевдонимы являются внутренними для пользовательского агента и не требуют поддержки от остальных компонентов почтовой системы.

Псевдоним можно также определить в файле пересылки в рабочем каталоге каждого пользователя (**`~.forward`**). Эти псевдонимы, имеющие несколько необычную синтаксическую структуру, применяются ко всей почте, поступающей конкретному пользователю. Они часто используются для пересылки почты по другому адресу или для реализации автоответчика.

Транспортные агенты ищут псевдонимы в глобальных файлах **`aliases`** (**`/etc/mail/address`** или **`/etc/aliases`**), а затем в файлах пересылки получателя. Псевдонимы применяются только к сообщениям, которые транспортный агент считает локальными.

Формат записи в файле **`alises`** имеет следующий вид.

```
локальное_имя: получатель1,получатель2, ...
```

Здесь *локальное_имя* — оригинальный адрес, с которым будут сравниваться входящие сообщения, а список получателей содержит либо адреса получателей, либо другие псевдонимы. Строки, начинающиеся с отступа, считаются продолжением предыдущих строк.

[6] По техническим причинам псевдонимы настраивает только системный администратор. Пользователи управляют маршрутизацией почты при помощи файла ***.forward***, который в действительности не имеет прямого отношения к псевдонимам, но мы использовали оба этих средства.

С точки зрения почтовой системы, файл **aliases** заменяет файл **/etc/passwd**, поэтому запись

```
david: david@somewhere-else.edu
```

не позволит локальному пользователю david вообще получать сообщения электронной почты. Следовательно, при выборе новых имен для пользователей администраторы и инструменты **adduser** должны проверять как файл **passwd**, так и файл **aliases**.

Файл **aliases** всегда должен содержать псевдоним "postmaster", перенаправляющий сообщение тому, кто обслуживает почтовую систему. Аналогично псевдоним "abuse" подходит для ситуаций, когда кто-то извне вашей организации хочет прислать вам спам или предпринимает подозрительные действия в вашей сети. Кроме того, в файле должен присутствовать псевдоним для автоматических сообщений от транспортного агента; обычно этот агент называется Mailer-Daemon и часто имеет псевдоним "postmaster".

К сожалению, в настоящее время в почтовой системе происходит столько злоупотреблений, что некоторые организации настраивают эти стандартные контактные адреса так, чтобы отбрасывать почту вообще, вместо ее пересылки индивидуальным пользователям. Записи вроде

```
# Basic system aliases - these MUST be present
mailer-daemon:  postmaster
postmaster:     "/dev/null"
```

стали обычными. Мы не рекомендуем эту практику, поскольку люди, имеющие проблемы с пересылкой почты в вашу организацию, должны сообщить об этом пользователю postmaster.

Мы рекомендуем использовать такие записи.

```
# Basic system aliases - these MUST be present
mailer-daemon:  "/dev/null"
postmaster:     root
```

Вы должны перенаправлять почту, поступающую на корневой сервер, системным администраторам или тому лицу, которое ведет ежедневные регистрационные записи. Учетные записи bin, sys, daemon, nobody и hostmaster (а также любые другие учетные записи псевдопользователей) должны иметь похожие псевдонимы.

Помимо списков пользователей, псевдонимы могут ссылаться на следующие файлы и программы.

- Файл, содержащий список адресов.
- Файл, в который должны добавляться сообщения.
- Программу, на вход которой должны передаваться сообщения.

Два последних адресата должны вызывать подозрения с точки зрения безопасности, поскольку отправитель сообщения полностью определяет его содержание. Иметь возможность добавлять это содержание в файл или передавать его командам в качестве аргументов — значит подвергать систему опасности. Многие транспортные агенты либо не разрешают передавать сообщения таким адресатам, либо строго ограничивают список разрешенных команд и файлов.

Псевдонимы могут создавать циклические ссылки. Транспортные агенты пытаются распознавать циклы, которые вызывают бесконечную циркуляцию сообщений от получателя к отправителю и обратно, и возвращать ошибочные сообщения их отправителям. Для того чтобы распознать циклическую ссылку, транспортный агент может подсчитывать количество строк Received в заголовке сообщения и прекращать ретрансляцию,

если это количество превышает установленный предел (обычно 25). Каждое посещение нового компьютера, на жаргоне электронной почты, называется "скачком" ("hop"). Возвращение сообщения его отправителю называется "рикошетом" ("bouncing"). Итак, типичное выражение на этом жаргоне может звучать следующим образом: "Письмо отскакивает после 25 рикошетов".[7] Другой способ распознавания циклов заключается в добавлении заголовка Delivered-To для каждого компьютера, которому пересылается сообщение. Если транспортный агент обнаружит в этом заголовке адрес, который уже упоминался ранее, то он будет знать, что письмо "ходит по кругу".

## Загрузка списков рассылки из файла

Директива `:include:` в файле **aliases** (или пользовательском файле **.forward**) является прекрасным средством управления псевдонимами с помощью специального файла. Это отличное средство, позволяющее пользователям создавать свои собственные списки рассылки. Этот файл управляется только пользователем и его можно менять без вмешательства системного администратора. Однако такой псевдоним легко может стать пособником при рассылке спама, поэтому не следует разрешать ретрансляцию сообщений, поступающих извне, на эти адреса. Если внешнему пользователю необходимо послать сообщение с использованием псевдонима, применяйте такие программы, как Mailman (см. ниже), чтобы сохранить систему в безопасности.

Для создания такого списка рассылки с помощью директивы `:include:` системный администратор должен ввести псевдоним в глобальный файл **aliases**, а затем создать внешний файл и при помощи команды **chown** сделать его владельцем пользователя, управляющего списком рассылки. Например, файл **aliases** может содержать следующую строку.

```
sa-book: :include:/usr/local/mail/ulsah.authors
```

Файл **ulsah.authors** должен находиться в локальной файловой системе. Кроме того, запись в этот файл должна быть разрешена только его владельцу. Для полноты следует также создать псевдоним, соответствующий владельцу списка рассылки, чтобы сообщения об ошибках ("рикошеты") посылались владельцу, а не отправителю сообщения.

```
owner-sa-book: evi
```

О списках рассылки, а также об их взаимосвязи с файлом **aliases** пойдет речь чуть ниже.

## Направление почты в файл

Если объект, на который ссылается псевдоним, — полное имя файла (заключенное в двойные кавычки при наличии специальных символов), то сообщения добавляются в конец указанного файла. Сам файл должен существовать. Вот пример такого псевдонима.

```
cron-status: /usr/local/admin/cron-status-messages
```

Возможность посылать почту в файл или на вход программы очень полезна, но она ослабляет безопасность и потому должна быть ограничена. Приведенный синтаксис допустим только в файле **aliases** и пользовательском файле **.forward** (а также в файлах,

[7] Мы не придерживались последовательной терминологии в этой главе, иногда называя возврат сообщения "рикошетом", а иногда "ошибкой". На самом деле мы имеем в виду, что в этом случае генерируется уведомление о состоянии доставки (DSN, представляющее собой сообщение, форматированное специальным образом). Такое уведомление обычно означает, что сообщение не было доставлено и, следовательно, возвращается отправителю.

которые внедряются в них посредством директивы :include:). Имя файла не трактуется как адрес, поэтому почта, направленная по адресу /etc/passwd@host.domain, будет возвращена.

Если ссылка на внешний файл содержится в файле **aliases**, то адресуемый файл либо должен быть полностью открыт для записи (что нежелательно), либо иметь установленный бит смены идентификатора пользователя (SUID) и быть неисполняемым, либо принадлежать основному пользователю транспортного агента. Этот пользователь задается в файле конфигурации транспортного агента.

Если же ссылка на файл содержится в файле **.forward**, то адресуемый файл должен принадлежать и быть доступным для записи основному получателю сообщения. Необходимо, чтобы у этого пользователя была своя запись в файле **/etc/passwd**, а его интерпретатор команд был упомянут в файле **/etc/shells**. Для файлов, принадлежащих пользователю **root**, следует задать режим доступа 4644 или 4600, т.е. установить бит SUID и отменить возможность выполнения.

## Направление почты в программу

Благодаря псевдонимам можно организовать пересылку почты на вход заданной программы. Это реализуется с помощью строки примерно следующего вида.

```
autoftp: "|/usr/local/bin/ftpserver"
```

Однако использование такого способа доставки почты создает еще более серьезные бреши в защите, чем направление почты в файл, поэтому данный механизм тоже разрешен только для файлов **aliases** и **.forward** и для файлов, которые подключаются к ним посредством директивы :include:, и часто требует использования ограниченной оболочки (restricted shell).

## Примеры псевдонимов

Ниже представлен ряд псевдонимов, которые могут понадобиться системному администратору.

```
# Перенаправление на псевдозаписи
bin: root
daemon: root
adm: root
abuse: root
junk: "/dev/null"
root: ned

# Псевдонимы пейджеров
pigdog: :include:/usr/local/etc/pigdog
tier1coverage: :include:/usr/local/etc/tier1coverage
tier2coverage: :include:/usr/local/etc/tier2coverage

# Соглашения системных администраторов
diary: "/usr/local/admin/diary"
info: "|/usr/local/bin/sendinfo"

# Псевдонимы учебных групп, которые изменяются каждый семестр
sa-class: real-sa-class@nag.cs.colorado.edu
real-sa-class: :include:/usr/local/adm/sa-class.list
```

Псевдоним sa-class имеет два уровня, чтобы файл данных, содержащий список студентов, хранился только на одной машине. Псевдоним diary очень удобен для документирования работы этих странных системных администраторов, которые терпеть не могут документировать свои действия. Системные администраторы могут просто запоминать важные события, происшедшие с компьютером (обновление операционной системы, изменение аппаратного обеспечения, сбои и прочее), посылая сообщения в файл diary.

## Хешированная база данных псевдонимов

Поскольку записи файла **aliases** не упорядочены, прямой поиск в данном файле был бы для программы **sendmail** неэффективным. По этой причине создается хешированная версия файла **aliases**. Это делается средствами системы Berkeley DB. Хеширование значительно ускоряет поиск псевдонимов, особенно в больших файлах.

Файлы, образованные из файла **/etc/mail/aliases**, называются **aliases.db**. При каждом изменении файла **aliases** хешированную базу данных следует перестраивать с помощью команды **newaliases**. Если команда **newaliases** вызывается в автоматическом режиме, сохраняйте выводимые ею сообщения об ошибках, поскольку могут возникнуть проблемы с форматированием.

## Списки рассылки и программы для работы с ними

Список рассылки — это гигантский псевдоним, при помощи которого копия каждого адресованного ему сообщения передается каждому члену списка. Некоторые списки рассылки содержат адреса тысяч получателей.

Списки рассылки обычно объявляются в файле **aliases**, но формируются во внешнем файле. Существуют стандартные соглашения по присвоению имен, понятные транспортным агентам и специальным программам ведения таких списков. Опытные пользователи тоже придерживаются их. Эти соглашения иллюстрируются следующими примерами.

```
mylist: :include:/etc/mail/include/mylist
owner-mylist: mylist-request
mylist-request: evi
owner-owner: postmaster
```

В данном случае **mylist** — это имя списка рассылки; в файле **/etc/mail/include/mylist** содержится список его членов. Сообщения о рикошетах посылаются его владельцу (пользователю **evi**). Косвенная адресация "владелец—запрос—пользователь" удобна, так как адрес владельца (**mylist-request**) указывается в заголовке "Return-Path" каждого сообщения, посылаемого в список рассылки. Выражение "mylist-request" — это более подходящее содержимое заголовка, чем настоящее имя владельца. Сообщения об ошибках, возникающие при отправке писем псевдониму **owner-mylist** (на самом деле это пользователь **evi**), направляются псевдониму **owner-owner**.

Когда используется общесистемный файл псевдонимов, необходимо ввести еще один уровень косвенной адресации, связав псевдоним **mylist** с адресом `реальное_имя_списка@главный_компьютер`, чтобы файл, содержащий список членов, существовал только в одном месте.

## Программы для работы со списками рассылки

Существует два пакета — Mailman и Sympa,— которые автоматизируют работу со списками рассылки. Они лучшие в своем роде. Эти пакеты обычно позволяют пользователям

получать информацию о списке и легко оформлять подписку или отказ от нее. Они облегчают управление списком, а также выполняют фильтрацию спама и вирусов. Каждый пакет предоставляет возможность использования разных языков (обычных, не языков программирования).

И пакет Mailman (`gnu.org/software/mailman`), написанный на языке программирования Python, и пакет Sympa (`sympa.org`), написанный на языке программирования Perl, имеют веб-интерфейс, позволяющий пользователям оформлять подписку и отказ от нее, не прибегая к помощи менеджера списка.

## 20.6. Сканирование содержимого: спам и вредоносные программы

В этом разделе рассматриваются общие вопросы, касающиеся борьбы со спамом и вирусами, включая использование внешнего антивирусного инструмента **`amavis-new`**. Детали, относящиеся к конкретным транспортным агентам, описываются в соответствующих разделах.

Прежде чем приступать к сканированию содержимого писем, следует ответить на следующие вопросы.

- Где производить сканирование: в зоне DMZ или во внутренней сети?
- Когда производить сканирование: при первом соединении или после получения сообщения?
- В каком порядке производить сканирование?
- Что делать с обнаруженными вирусами и спамом?

Входящая корреспонденция обычно сканируется на концентраторе поступающих сообщений в зоне DMZ. В идеале ее следовало бы сканировать оперативно, чтобы вредные сообщения можно было отбрасывать, пока исходное соединение SMTP остается разомкнутым. Исходящую почту можно сканировать на вирусы и спам на внутреннем специальном компьютере, выполняющем маршрутизацию всех сообщений.

Проверка корректности сообщения должна состоять из следующих действий.

- Проверяем, соответствует ли реализация протокола SMTP, по которому работает отправитель, документации RFC.
- Проверяем, существуют ли локальные получатели.
- Сравниваем адрес с черным списком IP-адресов.
- Проверяем репутацию.
- Верифицируем подпись DKIM и запись SPF.
- Фильтруем спам.
- Фильтруем вирусы.

Многие роботы для рассылки спама неточно следуют протоколу SMTP; они обычно посылают сообщение еще до того, как получат ответ EHLO. Небольшая задержка на вашем сервере может выявить эту "несдержанность". Эту информацию можно использовать для того, чтобы либо разорвать соединение, либо увеличить рейтинг спама для полученных сообщений.

Проверка существования локальных получателей имеет смысл, если при дальнейшей проверке вы не искажаете их адреса. Ранняя проверка минимизирует работу почтового

сервера с недоставленной почтой. Кроме того, она исключает много "беспорядочного" спама. Однако она открывает доступ в ваше адресное пространство для зонда отправителя.

Порядок остальных проверок, в основном, зависит от связанных с ними затрат. Быстрые и простые проверки следует выполнять до более продолжительных и сложных, чтобы в среднем некорректные сообщения отфильтровывались на как можно более ранних стадиях.

Что делать с обнаруженным некорректным сообщением? Отклонить его, выбросить в мусорную корзину, отправить в карантин, заархивировать? Если вы тестируете настройки, мы рекомендуем отправлять их в карантин и архивировать. Если же система уже настроена так, как надо, отклоняйте или выбрасывайте в корзину все письма с вирусами, а также отклоняйте или архивируйте спам в соответствии с установленными вами правилами. Удаляйте заархивированный спам, срок давности которого превышает месяц; за это время пользователи смогут разобраться с письмами, которые были распознаны как спам по ошибке.

## Спам

Спам — это жаргонное название макулатурной почты, которую также называют непрошеной коммерческой электронной почтой (unsolicited commercial email — UCE). Она стала серьезной проблемой, потому что, несмотря на невысокий процент отвечающих, стоимость ответа в расчете на доллар высока. (Список, состоящий из 30 миллионов адресов электронной почты, стоит около 40 долл.) Если бы это не было выгодно для спамеров, то проблема не достигла бы таких масштабов. Исследования показывают, что 95–98% всех почтовых отправлений являются спамом. Самые свежие данные на эту тему можно найти на специализированных сайтах, например `spamlinks.net`.

Для борьбы со спамом рекомендуем использовать превентивные меры и открытые черные списки, доступные для вас. Хорошим источником таких списков является сайт `zen.spamhaus.org`. Можно также перенаправлять входящую корреспонденцию аутсорсинговым компаниям по борьбе со спамом, таким как Postini (которая стала частью компании Google) или Message Labs (которая в настоящее время является частью компании Symantec). Однако это может повлечь за собой снижение производительности, уровня конфиденциальности и безопасности.

Посоветуйте своим пользователям просто удалять спам, который они получают. Многие сообщения, являющиеся спамом, содержат инструкции о том, как удалить свой адрес из списка рассылки. Если последовать этим инструкциям, спамеры могут действительно удалить ваш адрес из текущего списка, но они немедленно добавят его в несколько других списков с аннотацией "доставлено реальному человеку, прочитавшему это сообщение". С этого момента ваш электронный адрес будет стоить еще дороже.

Люди, продающие электронные адреса спамерам, для сбора адресов используют специальный вид атаки с помощью словаря. Начиная со списка распространенных фамилий, сканирующие программы добавляют к ним разные инициалы в надежде обнаружить реальный электронный адрес. Для того чтобы проверить этот адрес, программа связывается с почтовыми серверами, принадлежащими, например, пятидесяти крупнейшим интернет-провайдерам, и применяет команды VRFY, EXPN или RCPT к миллиардам адресов. Транспортные агенты могут блокировать команды VRFY и EXPN протокола SMTP, но не команду RCPT. Такие действия загружают ваш почтовый сервер и мешают быстро доставлять реальные почтовые сообщений. Для защиты от подобного рода атак транспортные агенты могут ограничивать количество выполнений команды RCPT, поступающих из одного источника.

## Подделки

Подделать электронное письмо очень просто; многие пользовательские агенты позволяют заполнять поле адреса отправителя чем угодно. Транспортные агенты могут использовать механизм аутентификацию по протоколу SMTP между локальными серверами, но они не могут этого сделать в масштабе всего Интернета. Некоторые транспортные агенты добавляют предупреждающие заголовки в исходящие локальные сообщения, которые могут быть подделаны.

В электронном сообщении личность отправителя может быть фальсифицирована. Будьте очень осторожны, если в вашей организации электронные сообщения используются в качестве средства авторизации, например, ключей от дверей, карточек доступа и денег. Вы должны предупредить об этом администраторов и полагаться на то, что они просматривают подозрительные письма, поступающие от авторитетных отправителей, и проверяют корректность сообщений. Следует быть еще более осторожным, если в сообщении предлагается предоставить необычной персоне непропорциональные привилегии.

## Конфиденциальность сообщений

🕮 Более подробно пакеты PGP и GPG описаны в разделе 22.10.

Если вы не используете какой-либо внешний пакет для шифрования, например Pretty Good Privacy (PGP), его клон для GNU (GPG) или S/MIME, то о конфиденциальности сообщений говорить не приходится. По умолчанию вся почта посылается незашифрованной. Шифрование требует специальной поддержки со стороны почтовых пользовательских агентов. Если ваши пользователи хотят сохранить свои сообщения в тайне, они должны применять собственное шифрование.

Пакеты S/MIME и PGP описаны в документах RFC, причем стандартом считается S/MIME. Однако мы предпочитаем пакеты PGP и GPG; они являются более доступными. Пакет PGP был разработан выдающимся криптографом Филом Циммерманом (Phil Zimmermann), которому мы доверяем.

Эти стандарты образуют основу для обеспечения конфиденциальности электронной почты, аутентификации, проверки целостности сообщений и гарантии сохранения авторства. Однако анализ трафика по-прежнему возможен, потому что заголовки и конверты пересылаются открытым текстом.

## Фильтрация спама

Проблема спама привела к борьбе между борцами со спамом, с одной стороны, и спамерами, с другой, причем обе стороны вооружены сложными технологиями. В настоящее время для текущего контроля используются следующие показатели.

- "Серые" списки: временная отсрочка (форма проверки на соответствие документам RFC).
- SpamAssassin: эвристический инструмент для распознавания спама на основе сравнения с образцами.
- "Черные" списки: списки известных спамеров; часто на основе системы DNS.
- "Белые" списки: списки разрешенных отправителей на основе системы DNS, чтобы избежать ложного распознавания спама.
- Почтовые фильтры, сканирующие заголовки и тело сообщения.

- Записи SPF и DKIM/ADSP для идентификации доменов отправителей и почтовых правил.
- Системы **amavisd-new** и MailScanner: антивирусные и антиспамовские фильтрующие системы.

Боле подробно эти инструменты будут рассмотрены немного позже.

## Когда следует фильтровать

Это фундаментальный вопрос, и на него нет однозначного ответа. Основная проблема заключается в том, следует ли выполнять фильтрацию одновременно с выполнением транзакции SMTP при соединении с отправителем или уже после получения сообщения. Эти схемы имеют как преимущества, так и недостатки. Преимущества оперативной фильтрации (до размещения в очереди почтовой системы) заключаются в следующем.

- Вы можете отказаться от приема почты и не принимать на себя ответственность за ее доставку. (В некоторых странах это даже является юридическим требованием!)
- Отправитель гарантированно уведомляется о причине, по которой его сообщение не могло быть доставлено. Вы не обязаны доверять отправителю сообщения; вы просто формулируете причину отказа и поручаете исходному почтовому серверу сообщить об этом отправителю. Это более аккуратный и надежный способ, чем прием почты и ее отправка обратно.

Однако у схемы обработки почты после ее размещения в очереди почтовой системы тоже есть свои преимущества.

- Производительность почтового сервера, имеющего выход в Интернет, не страдает от интенсивной проверки спама. Это особенно ценно, когда одновременно со спамом почтовый сервер загружен полезными сообщениями.
- Фильтрация после размещения сообщения в очереди почтовой системы проще и надежнее.

На первый взгляд, может показаться, что следует предпочесть фильтрацию после размещения сообщения в очереди почтовой системы. Она не снижает производительность работы почтового сервера и легче управляется системными администраторами. Однако обратная отправка сообщения, генерируемая системой фильтрации после размещения в очереди почтовой системы, сама становится разновидностью спама, если адрес отправителя был подделан, как это обычно бывает при рассылке спама.

Эта проблема называется "обратной рассылкой спама" ("backscatter spam"). Для борьбы с ней была изобретена система BATV (bounce address tag validation — проверка обратного адреса). Тем не менее проблема остается нерешенной. Система может определить корректность обратного адреса (адрес отправителя на конверте), если отправитель сообщения подписал адрес на конверте. Фильтры системы BATV могут помочь сайтам отправлять сообщения только по корректным обратным адресам.

Разумным компромиссом было бы выполнение фильтрации вирусов и спама до размещения сообщений в очереди почтовой системы, а затем выполнение дополнительного сканирования после размещения сообщений в очереди.

## "Серые" списки/DCC

"Серые" списки — это схема, в которой почтовый сервер конфигурируется так, чтобы при установлении всех соединений с новыми, нераспознанными, IP-адресами воз-

никала задержка, например, от 15 минут до часа. Сервер отказывается принимать почту, отправляя в ответ сообщение "повторите попытку позже". Реальные транспортные агенты, посылающие реальную почту реальных пользователей, подождут и затем повторят попытку; роботы для рассылки спама перейдут к следующему адресу в списке и не будут повторять попытку.

"Серые" списки были реализованы для компьютеров, на которых работают транспортные агенты; подробности можно найти на сайте `greylisting.org`. Они особенно эффективны в качестве компонентов системы борьбы со спамом под названием DCC (Distributed Checksum Clearinghouses; см. `rhyolite.com/dcc`), распознающей "массовость" сообщения путем вычисления нечеткой контрольной суммы и проверки, сколько почтовых серверов имеют ту же самую контрольную сумму. Эта система распознает не спам как таковой, а массовые рассылки. Если вы поместите в белый список всю массовую рассылку, которую хотите получать (например, списки рассылки, принадлежащие вам), то остальные распознанные сообщения образуют непрошеную почту, т.е. спам.

Система DCC также может работать с серыми списками; она используется как фильтр и может заносить в серые списки или отвергать письма до их постановки в очередь почтовой системы во время сеанса SMTP. Поскольку система DCC не выполняет сравнение с образцами, как инструменты типа SpamAssassin, ее невозможно обмануть, добавляя случайные слова в свои сообщения, как делают спамеры, пытаясь обмануть системы, сравнивающие образцы.

Эффективность серых списков снизилась (с более чем 97% до 90%), когда роботы для рассылки спама восприняли их всерьез и привели в порядок свою реализацию протокола SMTP. Однако они по-прежнему остаются эффективными, если используются в сочетании с черными списками, потому что система автоматической поддержки черных списков часто управляется сайтами, специализирующимися на борьбе со спамом, пока не истечет период для повторной попытки.

## Программа SpamAssassin

Программа SpamAssassin (`spamassassin.apache.org`) — это открытый модуль на языке Perl, написанный Хабибом Диу (Habeeb Dihu) и поддержанный Яном Джастманом (Ian Justman). Он прекрасно справляется с распознаванием спама. Эту программу можно вызвать с помощью фильтра и использовать во многих системах для борьбы со спамом.

Для распознавания спама программа SpamAssassin использует множество эмпирических правил. Эти правила постоянно обновляются, но в настоящее время они поддерживаются все менее активно. Программа SpamAssassin перехватывает практически весь спам, но иногда принимает "ложноположительные" решения, особенно если она настроена с включенной опцией "авто-Байес". При настройке программы SpamAssassin и выборе ее параметров внимательно изучите накопленный спам.

Программа SpamAssassin использует систему подсчета баллов для оценки сообщений. Если сообщение набирает слишком много баллов, количество которых зависит как от настроек сайта, так и от пользователя, то программа SpamAssassin помечает сообщение как спам. После этого подозрительные сообщения можно отправить в папку для спама, либо запустив на сервере фильтр, например фильтр **`sieve`** фирмы Cyrus, либо соответствующим образом настроив своего пользовательского агента. Программу SpamAssassin можно даже обучить распознавать хорошие и плохие сообщения ("ham" и "spam"), подключив байесовский фильтр.

## Черные списки

🕮 Более подробная информация о системе DNS приведена в главе 17.

Несколько организаций (например, `spamhaus.org`) составляют списки спамеров и публикуют их в системе DNS. Транспортные агенты можно настроить так, чтобы они проверяли эти черные списки (известные также как Realtime Black Lists, или RBL) и отвергали почту, приходящую с указанных адресов.

Они также составляют списки открытых почтовых станций, т.е. почтовых серверов, предназначенных для пересылки сообщений из Интернета пользователю, находящемуся вне локальной сети, без аутентификации сервера-отправителя. Спамеры используют открытые почтовые станции для маскировки источников сообщений и перепоручения рассылки огромного количества сообщений другим сайтам.

## Белые списки

Белые списки — это репутационные списки, основанные на системе DNS, которые, по существу, являются противоположностью черных списков, описанных выше. Они используются для сокращения количества "ложноположительных" ответов, генерируемых фильтрами спама. Один из белых списков, поддерживаемых на сайте `dnswl.org`, оценивает домены следующим образом.

- Высокий рейтинг — никогда не посылает спама.
- Средний рейтинг — редко посылает спам, исправляет проблемы, связанные со спамом, если они возникают.
- Низкий рейтинг — иногда посылает спам, медленно исправляет проблемы.
- Нет рейтинга — законный почтовый сервер, но может рассылать спам.

Белые списки рекомендуют пропускать часть обычной процедуры сканирования почты с учетом рейтинга отправителя в белом списке.

- Не искать в черном и сером списках домены, имеющие рейтинг.
- Не выполнять фильтрацию спама для доменов с высоким или средним рейтингом.
- Никогда не игнорировать сканирование на вирусы.

Веб-сайт `dnswl.org` содержит подробную информацию о том, как использовать белый список для каждого транспортного агента, рассмотренного в нашей книге. Анализ выполняется на основе системы DNS, как и для черных списков. Например, если вы хотите узнать рейтинг IP-адреса 1.2.3.4, должны выполнить DNS-запрос для псевдоузла `4.3.2.1.list.dnswl.org`. В ответ вы получите IP-адрес в форме `127.0.`*x*`.`*y*, где *x* — число, идентифицирующее общую категорию бизнеса для данного домена (например, финансовые услуги или маркетинг электронной почты), а *y* — рейтинг сайта в белом списке по шкале 0–3 (0 — нет, 3 — высокий).

Для ускорения работы белых списков можно загрузить данные для полного белого списка и ежедневно выполнять команду **`rsync`**; каждый час или каждые полчаса это делать не следует.

Можно также оценить рейтинг своей организации на веб-сайте `dnswl.org`. Ниже приводится типичный вывод для сайта `caida.org`, не рассылающего спам.

```
IP range          192.172.226.32/32
Domain/Hostname    jungle.caida.org
Score             med
```

```
IP range          192.172.226.36/32
Domain/Hostname    fido.caida.org
Score             med

IP range          192.172.226.78/32
Domain/Hostname    rommie.caida.org
Score             med
```

Домен `hotmail.com` порождает около десяти страниц, содержащих записи, которые имеют рейтинг "нет".

## Фильтрация почты

Разработчики программы **sendmail** создали прикладной интерфейс, позволяющий программам сторонних производителей фильтровать заголовки и содержимое почтовых сообщений так, будто они обрабатываются транспортным агентом.

Эти почтовые фильтры используются для борьбы со спамом, распознавания вирусов, статистического анализа, шифрования и многих других целей. Почтовые фильтры поддерживаются как утилитой **sendmail**, так и транспортными агентами сервера Postfix; вместо них сервер Exim использует обычные фильтры и списки контроля доступа. Каталог доступных почтовых фильтров, сопровождаемый рейтингами пользователей, лицензионной информацией и статистическими показателями о загрузках и обновлениях, находится на сайте `milter.org`.

Транспортные агенты применяют почтовые фильтры к входящим сообщениям, пока те еще остаются на связи с сайтом-отправителем. Почтовые фильтры могут распознать профиль вируса или спама и сообщить об этом транспортному агенту, игнорировать сообщение, создать записи в журнале или предпринять другое действие, которое вы сочтете приемлемым. Почтовые фильтры имеют доступ и к метаданным, и к содержимому сообщения.

Почтовые фильтры представляют собой потенциально мощные инструменты как для борьбы со спамом и вирусами, так и с попытками нарушить конфиденциальность переписки. Когда менеджеры узнают, что по электронной почте из их организации происходит утечка конфиденциальной информации, возникает щекотливая ситуация, потому что сотрудники полагают, будто их сообщения должны считаться приватными. В соглашениях с сотрудниками следует четко указывать, что вы имеете право выполнять любое сканирование их электронных сообщений.

## Технология SPF и спецификации Sender ID

Лучший способ борьбы со спамом — остановить работу его источника. Это звучит легко и просто, но в реальной жизни это практически невозможно сделать. Структура Интернета затрудняет отслеживание реального источника сообщения и проверку его аутентичности. Обществу необходим надежный способ верификации отправителя и его намерений.

Для решения этой проблемы было сделано много предложений, но технология SPF и спецификации Sender ID оказались наиболее удачными. Технология SPF (Sender Policy Framework — инфраструктура политики отправителей) описана организацией IETF в документе RFC4408. Протокол SPF определяет набор DNS-записей (см. раздел 17.8), с помощью которых организация может указать свои официальные почтовые серверы для исходящей корреспонденции. Транспортные агенты могут отвергать сообщения, исходящие из домена организации, если они не отправлены из указанных источников.

Разумеется, эта система работает хорошо только в том случае, когда большинство организаций публикует записи SPF. Для проверки записей SPF разработано несколько почтовых фильтров.

Спецификации Sender ID и технология SPF практически идентичны по форме и функциям. Однако ключевые части технологии Sender ID запатентованы компанией Microsoft, следовательно, являются предметом полемики. В момент написания книги компания Microsoft все еще пыталась склонить индустрию к принятию своих запатентованных стандартов. Организация IETF решила не делать однозначный выбор и опубликовала документы RFC4406 (о спецификациях Sender ID) и RFC4408 (о технологии SPF). Обе технологии названы экспериментальными, так что рынок сам должен решить, какая из них будет выбрана в качестве стандартной.

Отложенные сообщения не соответствуют ни спецификациям Sender ID, ни протоколу SPF, что является серьезным недостатком обеих систем. Получатель проверяет запись SPF об оригинальном отправителе, чтобы открыть его список авторизованных серверов. Однако эти адреса не соответствуют компьютерам, выполняющим задержку писем и задействованным в транспортировке данного сообщения. Следует быть осторожным, принимая решения на основе отказов системы SPF.

## Системы DomainKeys, DKIM и ADSP

🕮 Более подробно технология DKIM описана в главе 17.

DKIM (DomainKeys Identified Mail — почта, идентифицированная доменными ключами) — это система криптографических подписей для сообщений электронной почты. Она позволяет получателю верифицировать не только личность отправителя, но и тот факт, что сообщение не было подделано по пути. Система использует DNS-записи для публикации доменных криптографических ключей и правил подписи сообщений.

Оригинальная система DomainKeys, предшественница системы DKIM, обладавшая аналогичными функциональными возможностями, продвигалась компанией Yahoo!. Она по-прежнему используется. Системы DKIM и DomainKeys не конфликтуют, и организации могут верифицировать подписи обоих типов. И все же для подписи новых сообщений лучше использовать систему DKIM.

Записи ADSP DNS (Author Domain Signing Practice) позволяют отправителю объявлять свои правила подписи для каждого поддомена. Например, банк может объявить, что он подписывает всю почту, исходящую с сервера `transactions.mybank.com`. Любой адресат, получивший неподписанную почту (или почту, подпись на которой не была верифицирована), которая утверждает, что отправлена с указанного домена, должен либо отвергнуть ее, либо отправить в карантин. Однако сайт `marketing.mybank.com` может вообще не подписывать свои сообщения.

Некоторое время система ADSP называлась SSP (sender signing policy — политика подписи отправителя), так что ее можно было рассматривать как разновидность обработки записей DNS TXT. Технология DKIM поддерживалась всеми транспортными агентами, описанными в этой главе, но ее развертывание в реальном мире по неизвестным причинам оказалось медленным.

Даже если вы не хотите отвергать сообщения на основе верификации по технологиям DKIM или SPF, все еще можете использовать информацию, повышающую рейтинг спама, или изменять свою реакцию в соответствии с репутацией отправителя.

## Функциональные возможности транспортных агентов по борьбе со спамом

Каждый транспортный агент имеет настройки конфигурации, которые облегчают борьбу со спамом. Например, некоторые транспортные агенты могут распознать, что их просят выполнить миллиарды операций RCPT, и ввести 15-секундную задержку между выполнением команд RCPT для соединений, которые требуют их выполнить.

Мы рассмотрим настройки конфигурации, предназначенные для борьбы со спамом, вместе с остальными настройками транспортных агентов. Для утилиты **sendmail** см. раздел 20.8, для сервера Exim — раздел 20.13, а для сервера Postfix — раздел 20.15. Технологии DKIM и ADSP более подробно обсуждаются в разделах 20.16 и 17.8.

## Программа MailScanner

Эта программа (`mailscanner.info`), написанная Джулианом Филдом (Julian Field), представляет собой активно используемый, гибкий и открытый сканер концентраторов почты, который распознает спам, вирусы и попытки фишинга. Он написан на языке Perl и использует внешние антивирусные программы (в частности, ClamAV и 25 других инструментов), а также программное обеспечение для борьбы со спамом (SpamAssassin). Его компонент ScamNailer (`scamnailer.info`), предназначенный для предотвращения фишинга, является независимым и может использоваться самостоятельно.

Системный администратор может настроить конфигурацию программы MailScanner на уровне пользователей, доменов и IP-адресов.

Программа MailScanner — это не почтовый фильтр, а самостоятельная программа, работающая с очередями сообщений, принадлежащих транспортным агентам. Например, можно конфигурировать транспортный агент в своей зоне DMZ, чтобы принимать сообщения (после их проверки с помощью почтовых фильтров, черных списков и так далее, но до размещения в очереди), а затем размещать их в очереди входящих сообщений. Программа MailScanner считывает сообщения из очереди, проверяет их на спам, вирусы и фишинг, а затем передает в отдельную очередь исходящих сообщений. Другой экземпляр транспортного агента может обработать очередь входящих сообщений и доставить их адресату.

Недостатком этой системы является тот факт, что почта, отвергнутая программой MailScanner, создает рикошет сообщений и, следовательно, может порождать обратную рассылку спама.

Несмотря на то что программа MailScanner является свободной распространяемой, существует ее коммерческая версия. Кроме того, она поддерживает список рассылки для активных пользователей и выделенный канал IRC, который отслеживается круглосуточно. Она хорошо документирована как в Интернете, так и в книге *MailScanner: A User Guide and Training Manual*, написанной Джулианом Филдом. Файл конфигурации программы MailScanner содержит так много комментариев, что примитивы конфигурации трудно распознать; если вы эксперт, то, возможно, захотите удалить некоторые из этих формулировок.

Статистические показатели, определяемые программой MailScanner, можно получить от веб-интерфейса Mail-Watch. Графическое изображение данных можно создать с помощью программного обеспечения MRTG (рис. 20.3). Обратите внимание на огромный пик, возникший 19 сентября. Вероятно, в этот день имела место атака определенного типа.

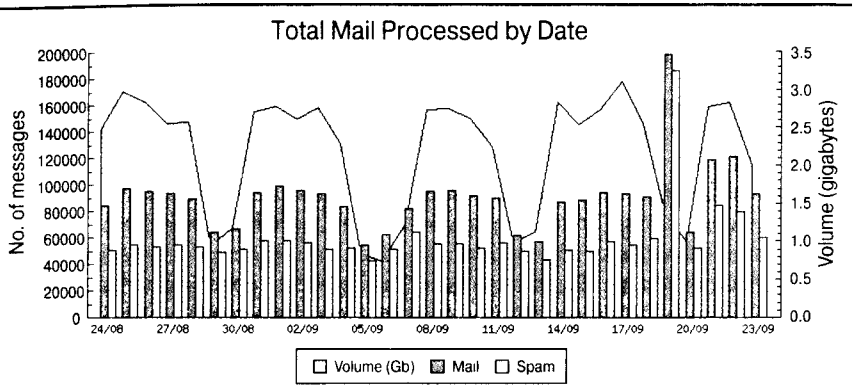



*Рис. 20.3. График почтового трафика, созданный с помощью программы MRTG*

## Интерфейс amavisd-new

Программа **amavisd-new** — это интерфейс между транспортными агентами и сканерами спама, такими как ClamAV и SpamAssassin. Изначально он был основан на сканере AMaViS (A Mail Virus Scanner), но в настоящее время имеет с оригиналом мало общего. Он написан на языке Perl Марком Мартинесом (Mark Martinec); веб-сайт: `ijs.si/software/amavisd`. Следуя соглашениям с организациями, поддерживающими эту программу, мы называем весь пакет (полужирным шрифтом) **amavisd-new**, хотя сам демон называется **amavisd**.

Программа **amavisd-new** взаимодействует с транспортным агентом через локальный домен или сокет TCP. Она может фильтровать письма либо до их размещения в очереди (с помощью почтового фильтра, до того как транспортный агент получит сообщение), либо после приема сообщения, но до доставки его получателю.

Зачем нужна еще одна программа, если транспортный агент и так выполняет сканирование с помощью почтовых фильтров и аналогичных инструментов? Ответ заключается в том, что конфигурацию всех фильтров удобно хранить в одном месте. Кроме того, если весь процесс фильтрации координируется одним интерфейсом, то легче отражать новую атаку или включать в линию обороны новый инструмент. Другое преимущество заключается в том, что сканер может работать на отдельном компьютере и тем самым облегчать работу по приему и обработке сообщений на загруженном сервере. Компромиссом является следующая схема: простые и быстрые проверки выполнять с омощью транспортного агента, а более сложные и продолжительные — с помощью инструмента типа **amavisd**.

Программа **amavisd** может взаимодействовать со многими пакетами, предназначенными для борьбы со спамом и вирусами. Это довольно мощный инструмент, хотя и обладает некоторыми недостатками.

- Документация немного разбросана, и не очень понятно, что является современным вариантом, а что устаревшим и не действующим.
- Сложная конфигурация; содержит множество параметров и их слабо различимых вариантов.

### Как работает программа amavisd

Программа **amavisd** занимает промежуточное место между транспортными агентами, хранящими сообщение, подлежащее проверке, и программным обеспечением, которое будет выполнять эту проверку. Программа **amavisd** прослушивает соединения с транспортным агентом на порту 10 024, отдает протоколам SMTP или LMTP приказ принять сообщения и возвращает ответы транспортному агенту на порту 10 025. Кроме того, если она и транспортный агент выполняются на одном и том же компьютере, эта программа использует сокет в локальном домене.

Если программа **amavisd** передает отсканированное сообщение и результат сканирования обратно транспортному агенту, от которого она получила данное сообщение, то можно выполнить фильтрацию до размещения сообщения в очереди и отвергнуть его во время первоначального сеанса SMTP, выполняемого транспортным агентом. Если же вместо этого программа **amavisd** размещает сообщения в очереди внутреннего концентратора почты, то фильтрация производится автономно и сообщения могут быть отвергнуты или отправлены обратно.

🕮 Информация о протоколе SNMP приведена в главе 21.

Пакет **amavisd** старается не потерять сообщения и не пропустить их без проверки, учитывает пожелания получателей и следует стандартам, описанным в разных документах RFC, посвященных электронной почте. Несмотря на то что программа **amavisd** написана на языке Perl, ее производительность довольно высока. Она сканирует каждое сообщение только один раз, независимо от того, сколько получателей с ним связано. Процесс сканирования выполняется довольно интенсивно, инструменты, входящие в дистрибутивный набор, могут отслеживать процесс фильтрации с помощью протокола SNMP. Программу **amavisd** не обязательно запускать как корневую; у нее хорошая репутация в области безопасности.

### Инсталляция пакета amavisd

Загрузите самую свежую версию программы с сайта `Download the latest ijs.si/software/amavisd` или выполните команду **grab** в пакете Linux и пройдите этапы, описанные в файле **INSTALL**, расположенном на вершине иерархии дистрибутива. Домашняя страница проекта имеет ссылки на заранее скомпилированные пакеты.

Программа **amavisd** должна запускаться как пользователь и группа vscan или amavis, поэтому проще всего создать этого пользователя и группу, а затем зарегистрироваться как vscan, чтобы получить программное обеспечение и инсталлировать его. Если после инсталляции программа **amavisd** работает правильно, измените регистрационное имя на **/bin/false** или на имя другой ограниченной оболочки.

Файл **amavisd.conf-default** содержит список всех возможных параметров конфигурации и их значения, заданные по умолчанию. Файл **amavisd.conf-sample** содержит типичный пример конфигурации. Кроме того, файл **amavisd.conf** представляет собой минимальную стартовую площадку (хотя и содержит более 750 строк!) с комментариями к некоторым переменным, которые необходимо изменить. Языком конфигурации является Perl.

### Базовая конфигурация пакета amavisd

Рассмотрим базовую конфигурацию пакета **amavisd** для узла `mail.example.com`, где транспортный агент и пакет amavisd выполняются на одном и том же компьютере, а для взаимодействия используют протоколы TCP и SMTP.

```
use strict;

$myhostname = 'mail.example.com';
@local_domains_maps = (['.example.com']);
@mynetworks = qw(127.0.0.0/8 192.168.0.0/16);

$forward_method = 'smtp:[127.0.0.1]:10025';
$enable_db = 1;
$enable_global_cache = 1;

$max_servers = 5;
$DO_SYSLOG = 1;
$SYSLOG_LEVEL = "mail.info";
$bypass_decode_parts = 1;
$final_virus_destiny = D_REJECT;
$final_banned_destiny = D_REJECT;
$final_bad_header_destiny = D_PASS;
$log_recip_templ = undef;

@av_scanners = (
  ['ClamAV-clamd',
     \&ask_daemon, ["CONTSCAN {}\n", "/var/run/clamav/clamd"],
     qr/\bOK$/m, qr/\bFOUND$/m,
     qr/^.*?: (?!Infected Archive)(.*) FOUND$/m ],
);
1;
```

В массиве стандартных конфигураций `av_scanners` перечислены более 40 программ для борьбы со спамом и вирусами; в этом фрагменте показана только программа ClamAV. Выражение `1;` в конце файла является специфичным для языка Perl. Оно гарантирует, что сам файл в любом контексте языка Perl при попытке чтения будет интерпретироваться как истинный.

### *Инструменты amavisd-new*

Дистрибутивный пакет **amavisd-new** содержит два полезных инструмента: **amavisd-nanny** и **amavisd-agent**. Инструмент **amavisd-nanny** следит за состоянием пакета **amavisd**, а программа **amavisd-agent** обеспечивает доступ в реальном времени к счетчикам и шкалам, похожим на те, которые используются в протоколе SNMP. Обе программы требуют наличия библиотеки Berkeley DB library.

Программа **amavisd-nanny** демонстрирует состояние всех процессов пакета **amavisd**, какие сообщения они в данный момент обрабатывают, что именно делают и как долго будут выполняться. Запустив эту программу с флагом **-h**, можно увидеть сообщение о ее использовании и список состояний, в которых может находиться пакет **amavisd**. Рассмотрим наиболее интересные состояния: `S` — фильтрация спама, `V` — сканирование вирусов и точка — простой. Буква, обозначающая состояние, выводится на экран ежесекундно вместе с двоеточием после каждых десяти букв, чтобы их легче было считать. Программу **amavisd-nanny** следует запускать иногда, просто чтобы увидеть, как система работает. Рассмотрим пример.

```
$ sudo amavisd-nanny
process-id task-id   elapsed in   elapsed-bar (dots indicate idle)
           or state idle or busy
PID 01422:           0:09:51      .........:.........:.........:.....
PID 26784: 26784-18  0:00:01      ==
```

```
PID 01422:          0:09:53      .........:.........:.........:.....
PID 26784:          0:00:03      ...
```

Программа **amavisd-agent** — это агент, похожий на агентов в протоколе SNMP, который собирает статистические данные от всех выполняющихся демонов и может выводить такую информацию, как количество обработанных сообщений, время обработки каждого сообщения, доля содержащихся вирусов, наиболее распространенные вирусы и т.д. Приведем усеченный пример.

```
$ sudo amavisd-agent
entropy         STR ipwvEIo5VA
sysContact      STR
sysDescr        STR amavisd-new-2.6.1 (20080629)
sysLocation     STR
sysObjectID     OID 1.3.6.1.4.1.15312.2.1
sysServices     INT 64
sysUpTime       Timeticks 111090596 (12 days, 20:35:05.96)
...
ContentVirusMsgs              1274     3/h       0.5%    (InMsgs)...
InMsgs                        247458   515/h     100.0%  (InMsgs)...
InMsgsRecips                  297574   619/h     120.3%  (InMsgs)...
InMsgsSize                    28728MB  60MB/h    100.0%  (InMsgsSize)...
TimeElapsedTotal              62518s   0.253s/msg        (InMsgs)...
virus.byname.W32/MyDoom-N     9        0/h       15.5%   (ContentVirusM...
virus.byname.Troj/BredoZp-H   8        0/h       13.8%   (ContentVirusM...
```

Первый столбец чисел — это количество элементов, за ним следует вычисленный уровень и доля по отношению к основному значению, выраженная в процентах.

В данном случае почтовый сервер обработал 247 458 сообщений за 12 дней со средним уровнем 1,2 получателей на сообщение (показатель `InMsgsRecips` равен 120,3% входящих сообщений (InMsgs)). Сервер обнаружил 1274 вируса, которые составили около 0,5% всего почтового трафика. Сканеры в среднем потребовали 0,253 секунды на обработку одного почтового сообщения. Наиболее часто встречающимися вирусами были MyDoom-N и BredoZp-H.

## Проверка эффективности сканирования с помощью транспортных агентов

Проверяя, правильно ли ваши транспортные агенты идентифицируют вирусы и другое вредоносное программное обеспечение, вы должны использовать реальные инфицированные сообщения, чтобы убедиться, что ваши меры оказались эффективными. Это не следует делать в промышленном окружении, где события могут выйти из-под контроля. Организуйте безопасную, физически отдельную тестовую лабораторию, не соединенную с промышленной сетью.

Исследователи антивирусных программ скомпилировали небольшой тестовый файл и предоставили его организации EICAR (European Expert Group for IT-Security (`eicar.org/anti_virus_test_file.htm`)) для распространения. На самом деле это не вирус, о последовательность байтов, которую антивирусные программы добавляют в свои базы данных как вирус (обычно под описательным именем, например, EICAR-AV-Test). Тестовый файл можно свободно пересылать по электронной почте, распространять среди коллег и воспроизводить, не беспокоясь о вирусной опасности. Группа EICAR предоставляет несколько вариантов этого файла, так что его можно даже протестировать в архивах, например ZIP.

Текст GTUBE, обобщенный текст для массива непрошеной почты, похож на предыдущий файл, но предназначен для тестирования фильтров спама. Он доступен на сайте `spamassassin.apache.org/gtube`.

Если вы тестируете и отлаживаете свой транспортный агент с помощью протокола SMTP, проверьте инструмент SWAKS (SWiss Army Knife SMTP, `jetmore.org/john/code/#swaks`), разработанный Джоном Джетмором (John Jetmore). Он написан на языке Perl и позволяет легко тестировать обмен сообщениями по протоколу SMTP с аргументами в командной строке. Документацию можно найти на справочной странице или выполнив команду **`swaks --help`**. Для аутентификации по протоколу SMTP этот инструмент требует наличия библиотек **`libnet-ssleay-perl`** и **`libnet-dns-perl`**. Это не высшая математика, но все же быстрее, чем набирать команды SMTP вручную.

# 20.7. Конфигурация электронной почты

Ядром системы электронной почты является ее транспортный агент. Программа **`sendmail`** сначала была транспортным почтовым агентом системы UNIX, написанной много лет тому назад выпускником университета Беркли (UC Berkeley) Эриком Аллманом (Eric Allman). С тех пор были разработаны другие транспортные агенты. Одни из них являются коммерческими продуктами, а другие — программами с открытым кодом. В этой главе мы рассмотрим три почтовых транспортных агента с открытым кодом: **`sendmail`**, Postfix, написанный Витцем Венема (Wietse Venema) из компании IBM Research, и Exim, написанный Филипом Хазелем (Philip Hazel) из Кембриджского университета.

После высокоуровневого проектирования почтовой системы, ее настройка является второй по важности заботой системного администратора. К счастью, образцы конфигурации или примеры настроек, поставляемые вместе с транспортными агентами, часто оказываются очень близкими к тому, что требуется для среднестатистического сайта. Конфигурацию транспортного агента не следует начинать с нуля.

Сайт SecuritySpace (`securityspace.com`) ежемесячно проводит исследования, чтобы определить рыночную долю разных транспортных агентов. В отчете за декабрь 2009 года было указано, что ответы получены от 1,7 миллионов из 2 миллионов опрошенных транспортных агентов и 950 000 из них были идентифицированы. Результаты этого исследования приведены в табл. 20.4 наряду с результатами исследований, проведенных в 2007 году, и некоторыми значениями, полученными в 2001 году.

**Таблица 20.4. Рыночная доля почтовых транспортных агентов**

| MTA | Источник | Рыночная доля, % | | |
|---|---|---|---|---|
| | | 2009 | 2007 | 2001 |
| Exim | exim.org | 30 | 20 | 8 |
| Postfix | postfix.org | 20 | 15 | 2 |
| MS Exchange | microsoft.com/exchange | 20 | 22 | 4 |
| **sendmail** | sendmail.org | 19 | 29 | 60 |
| Другие | - | <3% | <3% | <3% |

Очевиден переход лидерства от программы **`sendmail`** к пакетам Exim и Postfix, в то время как компания Microsoft сначала достигла лидерства на рынке транспортных агентов, а потом уступила его. Среди остальных операционных систем все варианты системы

UNIX поставляются вместе с программой **sendmail**. Система Ubuntu перешла от пакета Postfix к пакету Exim, SUSE поставляется с пакетом Postfix, а Red Hat включает все три варианта, но по умолчанию устанавливает программу **sendmail**.

Для каждого из упомянутых транспортных агентов мы приводим детали конфигурации, необходимой для выполнения их функций, включая следующие.

- Конфигурация простых клиентов.
- Конфигурация почтового сервера, имеющего выход в Интернет.
- Управление маршрутизацией входящей и исходящей корреспонденции.
- Штемпелевание почты, поступающей от центрального сервера или самого домена.
- Безопасность.
- Отладка.

Если вы реализуете почтовую систему с нуля и не придерживаетесь заранее установленных правил или предпочтений, вам может быть трудно выбрать транспортный агент. Программа **sendmail** и пакет Exim определенно являются самыми сложными и, вероятно, допускают самую точную настройку и самые мощные опции. Пакет Postfix проще, быстрее и основной целью считает безопасность. Если ваша организация или системные администраторы ранее работали с какими-то транспортными агентами, они, вероятно, не захотят переключаться на что-то другое, если необходимые вам функциональные возможности не реализованы в старых агентах.

Конфигурация агента **sendmail** описывается в следующем разделе. Конфигурации пакета Exim описаны в разделе 20.14, а конфигурация пакета Postfix — в разделе 20.15.

## 20.8. Почтовый агент SENDMAIL

Почтовый агент **sendmail** распространяется в виде открытого кода и доступен на сайте `sendmail.org`, но в настоящее время его редко приходится собирать с самого начала. При желании можно сослаться на файл верхнего уровня **INSTALL**. Если необходимо обойти определенные установки, сделанные по умолчанию, то можно найти их в файле **devtools/OS/***имя-вашей-OS*. Можно также добавить новые функциональные свойства, отредактировав файл **devtools/Site/site.config.m4**. Программа **sendmail** использует препроцессор макросов **m4** не только во время компиляции, но и для конфигурирования. Файл конфигурации **m4** обычно называется *имя_компьютера*.**mc**. Он компилируется из довольно понятного синтаксиса в совершенно невразумительный низкоуровневый язык и записывается в файл *имя_компьютера*.**cf**, который в свою очередь инсталлируется как **/etc/mail/sendmail.cf**.

Для того чтобы увидеть версию программы **sendmail**, инсталлированной на вашем компьютере, а также узнать, как она скомпилировалась, попытайтесь выполнить следующую команду.

```
linux$ /usr/sbin/sendmail -d0.1 -bt < /dev/null
Version 8.13.8
Compiled with: DNSMAP HESIOD HES_GETMAILHOST LDAPMAP LOG
   MAP_REGEX MATCHGECOS MILTER MIME7TO8 MIME8TO7 NAMED_BIND
   NETINET NETINET6 NETUNIX NEWDB NIS PIPELINING SASLv2 SCANF
   SOCKETMAP STARTTLS TCPWRAPPERS USERDB USE_LDAP_INIT
============ SYSTEM IDENTITY (after readcf) ============
    (short domain name) $w = ross
(canonical domain name) $j = ross.atrust.com
```

```
        (subdomain name) $m = atrust.com
             (node name) $k = ross.atrust.com
==========================================================
```

Эта команда переводит программу **sendmail** в режимы проверки адреса (**-bt**) и отладки (**-d0.1**), но не вводит никакого адреса для тестирования (**</dev/null**). Побочный эффект заключается в том, что программа **sendmail** сообщает номер своей версии и флаги компилятора, с которыми она компилировалась. Зная номер версии, вы можете зайти на веб-сайт `sendmail.org` и увидеть, существуют ли у этой версии программы скрытые недостатки.

Для того чтобы найти файлы программы **sendmail** в своей системе, посмотрите в начало инсталлированного файла **/etc/mail/sendmail.cf**. Комментарии, которые там находятся, содержат имя каталога, в котором была собрана данная конфигурация. Этот каталог, в свою очередь, приведет вас к файлу **.mc**, являющемуся исходным источником конфигурации.

Большинство поставщиков программы **sendmail** включают в дистрибутивный пакет не только исполняемый модуль, но и каталог **cf** из дистрибутивного дерева, который они скрывают среди файлов операционной системы. Найти его поможет табл. 20.5.

**Таблица 20.5. Местоположение каталога конфигурации программы sendmail**

| Система | Каталог |
|---|---|
| Ubuntu | **/usr/share/sendmail** |
| SUSE | **/usr/share/sendmail** |
| Red Hat | **/usr/share/sendmail-cf** |
| Solaris | **/etc/mail/cf** |
| HP-UX | **/usr/newconfig/etc/mail/cf** |
| AIX | **/usr/samples/tcpip/sendmail/cf** |

## Файл переключения

🕮 Переключение служб более подробно описано в главе 19.

В большинстве систем существует конфигурационный файл "переключения служб" **/etc/nsswitch.conf**, в котором перечисляются методы, удовлетворяющие разным стандартным запросам, например запросам поиска пользователя и компьютера. Если для данного типа запросов в файле указано несколько методов распознавания, то файл переключения служб также определит порядок, в котором будут проверяться эти методы.

Существование переключения служб в программном обеспечении обычно не скрывается. Однако программа **sendmail** осуществляет очень плотный контроль над своими процедурами поиска, поэтому в настоящее время она игнорирует системный файл переключения служб и использует свой собственный внутренний файл конфигурации служб (**/etc/mail/service.switch**).

На почтовую систему оказывают влияние два поля в файле переключения служб: `aliases` и `hosts`. Поле `hosts` может иметь значения `dns`, `nis`, `nisplus` и `files`, а поле `aliases` — `files`, `nis`, `nisplus` и `ldap`. Вспомогательные файлы для механизмов, которые вы используете (за исключением `files`), должны быть скомпилированы в программе **sendmail** до того, как данная служба будет использована.

## Запуск программы sendmail

Программа **sendmail** не должна контролироваться демонами **inetd** или **xinetd**, поэтому ее необходимо явно запускать во время загрузки. Подробности этой процедуры описаны в главе 3. Функционирование программы **sendmail** определяется флагами, с которыми она запускается. Программу **sendmail** можно запускать в нескольких режимах, выбранных с помощью флага **-b**, который означает слово "be" ("быть") или "become" ("становиться") и всегда используется в сочетании с другим флагом, определяющим роль программы **sendmail**. Допустимые значения флага **-b** и флага **-A**, который производит выбор между агентами MTA и MSA, приведены в табл. 20.6.

**Таблица 20.6. Флаги командной строки для основных режимов работы программы sendmail**

| Флаг | Описание |
|---|---|
| -Ac | Использует файл конфигурации `submit.cf` и действует как агент доставки |
| -Am | Использует файл конфигурации `sendmail.cf` и действует как транспортный агент |
| -ba | Функционирует в режиме ARPANET (ожидает управляющих символов CR/LF в конце строки) |
| -bd | Функционирует в режиме демона и прослушивает соединения на порту 25 |
| -bD | Функционирует в режиме демона, но явно, а не в фоновом режиме [a] |
| -bh | Просматривает информацию о последнем соединении (аналогично команде `hoststat`) |
| -bH | Стирает с диска копию информации об устаревших соединениях (аналогично команде `purgestat`) |
| -bi | Инициализирует хешированные псевдонимы (аналогично команде newaliases) |
| -bm | Функционирует как почтальон, доставляет почту как обычно (по умолчанию) |
| -bp | Выводит на печать очередь почтовых сообщений (аналогично команде mailq) |
| -bP | Выводит на печать количество записей в очередях, расположенных в совместно используемой памяти |
| -bs | Устанавливает режим сервера SMTP (для стандартного входа, но не для порта 25) |
| -bt | Устанавливает режим проверки адреса |
| -bv | Только проверяет почтовые адреса; не посылает почту |

[a] Этот режим используется для отладки, чтобы увидеть сообщения об ошибках.

Если вы конфигурируете сервер, который будет принимать почту из Интернета, запустите программу **sendmail** в режиме демона (**-bd**). В этом режиме программа **sendmail** прослушивает сетевой порт 25 и ожидает задания.[8] Обычно, кроме него, устанавливается флаг **-q**, задающий интервал времени, в течение которого программа **sendmail** обрабатывает почтовую очередь. Например, флаг **-q30m** заставляет программу проверять очередь каждые 30 минут, а **-q1h** — каждый час.

Программа **sendmail** обычно пытается доставить сообщение немедленно, сохраняя его в очереди только на короткий промежуток времени, чтобы гарантировать надежность работы. Однако если ваш узел перегружен или адресат недоступен, программа **sendmail** размещает сообщение в очереди и пытается послать его снова. Программа **sendmail** использует персистентные обработчики очереди (queue runners), которые обычно запускаются во время загрузки. Она выполняет блокировку, поэтому параллельное выполнение нескольких обработчиков очереди вполне безопасно. Функциональная возможность

[8] Порты, которые прослушивает программа ***sendmail***, определяются параметром *DAEMON_OPTIONS; порт 25 задается по умолчанию.*

"группы очередей" позволяет работать со списками рассылки и очередями. Более подробно она описывается в разделе 20.12.

Программа **sendmail** считывает свой конфигурационный файл **sendmail.cf** только в момент запуска. Следовательно, если вы изменили конфигурационный файл, то должны либо прекратить выполнение и запустить программу **sendmail** заново, либо послать сигнал HUP. Программа **sendmail** создает файл **sendmail.pid**, содержащий идентификатор процесса и команду его запуска. Вы должны запустить программу **sendmail**, используя полностью определенный путь, потому что она, получив сигнал HUP, применяет сама к себе команду **exec**. Файл **sendmail.pid** позволяет процессу принимать сигнал HUP с помощью следующей следующей команды.

```
$ sudo kill -HUP `head -1 sendmail.pid`
```

Местоположение файла PID зависит от операционной системы. Обычно он находится по адресу **/var/run/sendmail.pid** или **/etc/mail/sendmail.pid**, но его можно задать в конфигурационном файле с помощью параметра confPID_FILE.

```
define(confPID_FILE, `/var/run/sendmail.pid`)
```

## Почтовые очереди

Программа **sendmail** использует, по крайней мере, две очереди: **/var/spool/mqueue**, когда она функционирует как транспортный агент на порту 25, и **/var/spool/clientmqueue**, когда она функционирует как агент доставки на порту 587.[9] Все сообщения, по крайней мере, на короткое время оказываются в очереди, прежде чем отправиться в дальнейший путь.

Сообщения, находящиеся в очереди, сохраняются в виде фрагментов в разных файлах. Каждое имя файла имеет двухбуквенный префикс, идентифицирующий фрагмент, за ним следует случайный идентификатор, построенный на основе идентификатора процесса в программе **sendmail**. Шесть возможных фрагментов приведены в табл. 20.7.

**Таблица 20.7. Префиксы для файлов из очереди**

| Префикс | Содержимое файла |
|---|---|
| **qf** | Заголовок сообщения и управляющий файл |
| **df** | Тело сообщения |
| **tf** | Временная версия файла **df** в момент его обновления |
| **Tf** | Информация о том, что было выполнено более 32 неудачных попыток блокировки |
| **Qf** | Информация о том, что сообщение было отправлено обратно и не может быть восстановлено |
| **xf** | Временный файл транскрипции сообщений об ошибках, поступивших от программы для рассылки писем |

Если подкаталоги **qf**, **df** и **xf** в каталоге очереди уже существуют, то эти фрагменты сообщения помещаются в соответствующий подкаталог. Файл **qf** содержит не только заголовок сообщения, но и адреса, указанные на конверте, дату, при наступлении которой сообщение следует вернуть как недоставленное, приоритет сообщения в очереди и причину, по которой сообщение оказалось в очереди. Каждая строка начинается с однобуквенного кода, идентифицирующего остальную часть строки.

[9] Для повышения производительности работы программа **sendmail** может использовать несколько очередей в каталоге mqueue; см. раздел 20.12.

Каждое сообщение, помещенное в очередь, должно иметь файлы **qf** и **df**. Все остальные префиксы программа **sendmail** использует, пытаясь доставить сообщение. Если компьютер дает сбой и перезагружается, то последовательность загрузочных команд для программы **sendmail** должна удалить файлы **tf**, **xf** и **Tf** из всех очередей. Системный администратор, ответственный за почту, должен иногда проверять файлы **Qf**, если локальная конфигурация вызывает рикошет. Эпизодическая проверка каталогов очереди позволит исправить проблемы до того, как произойдет катастрофа.

Почтовая очередь создает несколько предпосылок для неприятностей. Например, файловая система может переполниться (избегайте размещения каталогов **/var/spool/mqueue** и **/var/log** в одном и том же разделе диска), очередь может "засориться", а беспризорные почтовые сообщения могут "застрять" в очереди. В конфигурации программы **sendmail** есть параметры, обеспечивающие высокую производительность на очень загруженных компьютерах; рассмотрим их в разделе 20.12.

# 20.9. Конфигурация программы sendmail

Действия программы **sendmail** определяются одним конфигурационным файлом, который обычно называется **/etc/mail/sendmail.cf**, если программа **sendmail** запускается как транспортный почтовый агент, или **/etc/mail/submit.cf**, если программа **sendmail** действует как агент доставки. Флаги, с которыми запускается программа **sendmail**, определяют, какой файл конфигурации используется: флаги **-bm**, **-bs** и **-bt** означают использование файла **submit.cf**, если он существует, а все другие режимы используют файл **sendmail.cf**. Эти имена можно изменить с помощью флагов в командной строке или параметров конфигурационного файла, но лучше этого не делать.

Формат конфигурационного файла был разработан для того, чтобы облегчить его разбор с помощью компьютера, а не человека. Исходный файл препроцессора **m4** (с расширением **.mc**), который порождает файл с расширением **.cf**, является усовершенствованием, но его капризный и строгий синтаксис очень неудобен. К счастью, многие парадигмы, которые вы, возможно, захотите использовать, уже были предусмотрены и реализованы в дистрибутивном пакете в качестве готовых решений.

Настройка конфигурации программы **sendmail** состоит из нескольких этапов.

- Выбор роли для компьютера, который вы конфигурируете: клиент, сервер, получатель почты через Интернет и т.д.
- Выбор функциональных возможностей, необходимых для реализации этой роли, и создание файла с расширением **.mc** для данной конфигурации.
- Компиляция файла с расширением **.mc** с помощью препроцессора **m4**, чтобы создать файл конфигурации **.cf**.

Мы опишем функциональные возможности, типичные для общедоступных в рамках организации серверов, имеющих выход в Интернет, и для небольших настольных клиентов. Более подробное описание можно найти в документации программы **sendmail**, в книге *Sendmail* Брайана Косталеса и его соавторов (Bryan Costales et al.), а также в файле **cf/README** из дистрибутивного пакета.

## Препроцессор m4

Изначально предназначенный для использования в сочетании с языками программирования, препроцессор **m4** позволяет пользователям писать более удобочитаемые (или,

наоборот, запутанные) программы. Это достаточно мощный инструмент, чтобы оказаться полезным во многих ситуациях, связанных с преобразованием входной информации, он прекрасно работает с конфигурационными файлами программы **sendmail**.

Макрос препроцессора **m4** имеет следующую форму.

```
name(arg1, arg2, ..., argn)
```

Между именем и открывающей скобкой не должно быть пробелов. Левая и правая одинарные кавычки обозначают строки в качестве аргументов. Соглашения о кавычках в макросах препроцессора **m4** являются странными, поскольку левая и правая кавычки — это одинаковые символы. Кроме того, кавычки могут быть вложенными.

Препроцессор **m4** имеет несколько встроенных макросов, и пользователи могут определять свои собственные макросы. Основные встроенные макросы, используемые в конфигурации программы **sendmail**, приведены в табл. 20.8.

**Таблица 20.8. Макросы препроцессора m4, часто используемые программой sendmail**

| Макрос | Функция |
|---|---|
| `define` | Определяет макрос с именем arg1 со значением arg2 |
| `undefine` | Отменяет предыдущее определение макроса с именем arg1 |
| `include` | Включает (интерполирует) файл с именем arg1 |
| `dnl` | Игнорирует все символы до начала новой строки |
| `divert` | Управляет выходными потоками |

## Фрагменты конфигурации программы sendmail

Дистрибутивный пакет программы **sendmail** содержит подкаталог **cf**, в котором находятся все фрагменты, необходимые для конфигурации препроцессора **m4**. Если вы не самостоятельно инсталлируете программу **sendmail** на основе исходного текста, а полагаетесь на поставщика, то используйте местоположение каталога **cf**, указанное в табл. 20.5. Документация о конфигурации программы **sendmail** записана в файле **README**. Подкаталоги, перечисленные в табл. 20.9, содержат примеры и фрагменты, которые можно включать в свою конфигурацию.

**Таблица 20.9. Подкаталоги конфигурации программы sendmail**

| Каталог | Содержимое |
|---|---|
| **cf** | Эталонные файлы .mc (мастера конфигурации) |
| **domain** | Эталонные файлы m4 для разных доменов в домене Berkeley |
| **feature** | Фрагменты, реализующие разные функциональные возможности |
| **hack** | Особые функциональные возможности, ценность или реализация которых сомнительна |
| **m4** | Основной файл конфигурации и другие важные файлы |
| **ostype** | Местоположение файлов, зависящее от операционной системы, и другая полезная информация |
| **mailer** | Файлы m4, описывающие основные программы для рассылки (агенты доставки) |
| **sh** | Сценарии оболочки, используемые в препроцессоре m4 |

Каталог **cf/cf** содержит примеры файлов с расширением **.mc**. Фактически он содержит так много примеров, что в них легко запутаться. Мы рекомендуем хранить свои файлы с расширением **.mc** отдельно от примеров, размещенных в каталоге **cf**. Для этого следует либо создать новый каталог для своего сайта (**cf/*имя_сайта***), либо разместить

каталог **cf** отдельно под именем **cf.examples** и создать новый каталог **cf**. Если вы сделаете это, скопируйте в свой новый каталог сценарии **Makefile** и **Build**, чтобы инструкции из файла **README** сохранили свою работоспособность. В качестве альтернативы можно скопировать все собственные файлы конфигурации с расширением **.mc** в центральное место хранения, а не оставлять их в дистрибутивном пакете программы **sendmail**. Сценарий **Build** использует относительные пути, поэтому, если вы хотите создать файл с расширением **.cf** из файла с расширением **.mc** и не использовать имена их дистрибутивной иерархии программы **sendmail**, придется их изменить.

Файлы из каталога **cf/ostype** конфигурируют программу **sendmail** для любой конкретной операционной системы. Многие из них являются заранее заданными, но если вы изменили настройки своей системы, то вам придется либо модифицировать их, либо создавать новые. Скопируйте тот из них, который лучше соответствует вашей системе, и присвойте ему новое имя.

Каталог **cf/feature** — это место, где вы будете искать все фрагменты конфигурации, которые вам нужны. Там найдутся все функциональные возможности, которые могут оказаться полезными для сайта, использующего программу **sendmail**.

Остальные подкаталоги каталога **cf** служат, скорее, как заглушки. Их не надо исследовать или анализировать — просто используйте их.

## Конфигурационный файл, построенный на основе эталонного файла с расширением .mc

Перед тем как погрузиться в детали различных конфигурационных макросов, свойств и параметров, которые могут использоваться при конфигурировании программы **sendmail**, создадим простую конфигурацию, тем самым проиллюстрировав весь процесс. Предметом нашего примера будет концевой узел `myhost.example.com`; главный конфигурационный файл называется **myhost.mc**. Рассмотрим полный файл с расширением **.mc**.

```
divert(-1)
#### basic .mc file for example.com
divert(0)
VERSIONID(`$Id$')
OSTYPE(`linux')
MAILER(`local')
MAILER(`smtp')
```

За небольшим исключением, каждая строка содержит вызов готового макроса. Первые четыре строки представляют собой заглушку; они вставляют комментарий в скомпилированный файл, чтобы указать версию программы **sendmail**, созданный каталог конфигурации и т.д. Макрос `OSTYPE` интерполирует файл **../ostype/linux.m4**. Строки, содержащие вызов макроса `MAILER`, разрешают локальную доставку сообщений (пользователям, имеющим учетные записи на сервере `myhost.example.com`), а также доставку интернет-сайтам.

Для того чтобы создать реальный конфигурационный файл, просто примените команду **Build** к новому скопированному вами каталогу **cf**.

```
$ ./Build myhost.cf
```

В заключение инсталлируйте файл **myhost.cf** в соответствующем месте —как правило, как файл **/etc/mail/sendmail.cf**, хотя некоторые поставщики могут поместить его в другое место. Чаще всего поставщики размещают его в каталоге **/etc** и **/usr/lib**.

Для более крупных сайтов можно создать отдельный файл препроцессора m4, чтобы хранить там значения, заданные по умолчанию; поместите его в каталог **cf/domain**. Отдельные узлы могут затем включать его содержимое, используя макрос DOMAIN. Отдельный файл конфигурации нужен не каждому компьютеру, но каждая группа похожих компьютеров (с одинаковой архитектурой и одинаковыми ролями: сервер, клиент и т.д.), вероятно, нуждается в собственной конфигурации.

Порядок следования макросов в файле с расширением **.mc** не случаен. Он должен быть таким.

```
VERSIONID
OSTYPE
DOMAIN
FEATURE
определения локальных макросов
MAILER
```

Даже имея простую систему конфигурации **m4** для программы **sendmail**, вам, возможно, придется принимать несколько разных решений, связанных с конфигурацией вашего сайта. Читая о функциональных возможностях, описанных ниже, подумайте о том, как их применять к вашей сети. Небольшая сеть, вероятно, будет иметь один концентратор и концевые узлы, а значит, только два варианта файла конфигурации. В более крупной сети могут понадобиться отдельные концентраторы для входящей и исходящей почты и, вероятно, отдельный сервер POP/IMAP.

Независимо от сложности вашей сети и ее представления во внешнем мире (открытой, за брандмауэром или виртуальной частной сети, например), скорее всего, каталог **cf** содержит готовые фрагменты, подходящие для настройки и использования.

# 20.10. Примитивы конфигурации программы sendmail

Команды конфигурации **sendmail** чувствительны к регистру клавиатуры. По общепринятому соглашению, имена заранее определенных макросов состоят только из прописных букв (например, OSTYPE), команды препроцессора **m4** — только из строчных букв (например, define). Названия параметров конфигурации начинаются с префикса conf, набранного в нижнем регистре, и заканчиваются именем переменной, набранным в верхнем регистре (например, confFALLBACK_MX). Макросы обычно ссылаются на файл препроцессора **m4** с именем *../имя_макроса/arg1*.**m4**. Например, ссылка OSTYPE(`linux') включает файл **../ostype/linux.m4**.

## Таблицы и базы данных

Перед тем как погрузиться в детали, связанные с конкретными примитивами конфигурации, сначала следует обсудить таблицы (иногда называемые ассоциативными массивами (maps) или базами данных), которые программа **sendmail** может использовать для маршрутизации почты или перезаписи адреса. Большинство из них используется в сочетании с макросом FEATURE.

Таблица — это кеш (как правило, текстовый файл), содержащий информацию о маршрутизации, псевдонимах, правилах, а также другие данные, которые можно преобразовать в формат базы данных с помощью команды **makemap**, а затем использовать как источник информации при выполнении одной или нескольких операций программы **sendmail**.

Хотя данные для программы **sendmail** обычно хранятся в текстовом файле, они могут также поступать из системы DNS, от сервера NIS, от протокола LDAP или из других источников. Использование централизованного сервера IMAP освобождает программу **sendmail** от выполнения рутинных операций, связанных с розыском пользователей и их устаревших таблиц.

Поддерживаются две библиотеки баз данных: библиотека dbm/ndbm — стандарт для большинства версий систем UNIX и Linux, а также Berkeley DB — более крупная библиотека, поддерживающая многочисленные схемы хранения. Мы рекомендуем использовать библиотеку BDB, если она уже установлена в вашей системе или вы можете ее установить. Она работает быстрее, чем база данных dbm, и создает файлы меньшего размера.

Программа **sendmail** предусматривает три типа ассоциативных массивов.

- `dbm` — использует расширяемые алгоритмы хеширования (**dbm/ndbm**)
- `hash` — использует стандартную схему хеширования (DB)
- `btree` — использует B-дерево (DB)

В большинстве случаев использования таблиц в программе **sendmail** наилучшим является формат базы данных `hash`, заданный по умолчанию. Для создания базы данных из текстового файла следует использовать команду **makemap**, задав тип и имя выходного файла базы данных. Текстовая версия базы данных должна передаваться на стандартный вход команды **makemap**. Рассмотрим пример.

```
$ sudo makemap hash /etc/mail/access < /etc/mail/access
```

На первый взгляд эта команда выглядит ошибкой, которая может вызвать перезапись входного файла пустым результирующим файлом. Однако команда **makemap** добавляет специальный суффикс, так что реальный результирующий файл будет называться **/etc/mail/access.db** и конфликт не возникнет. Каждый раз при изменении текстового файла файл базы данных необходимо создавать заново с помощью команды **makemap** (но программа **sendmail** не требует использования сигнала HUP).

В текстовых файлах, на основе которых создаются ассоциативные массивы, могут находиться комментарии. Они начинаются символом # и продолжаются до конца строки.

В большинстве случаев в качестве ключа базы данных используется самое длинное из возможных совпадений. Как и в любой другой хешированной структуре, порядок следования записей во входном текстовом файле не имеет значения. Примитивы `FEATURE`, ожидающие файл базы данных как параметр, по умолчанию предусматривают в качестве типа базы данных тип `hash`, а в качестве имени базы данных — имя **/etc/mail/***имя_таблицы*.**db**.

## Обобщенные макросы и функциональные возможности

В табл. 20.10 перечислены основные примитивы конфигурации с указанием степени их использования (да, нет, возможно) и кратким описанием их действий. Более подробно они будут рассмотрены ниже.

**Таблица 20.10. Основные примитивы конфигурации программы sendmail**

| Примитив | Используется? | Описание |
|---|---|---|
| `OSTYPE` | Да | Включает пути, специфичные для операционной системы, и флаги программы рассылки |
| `DOMAIN` | Нет | Включает детали конфигурации, специфичные для сайта |

*Окончание табл. 20.10*

| Примитив | Используется? | Описание |
|---|---|---|
| MAILER | Да | Допускает использование программ рассылки, обычно smtp и local |
| FEATURE | Возможно | Допускает использование набора функциональных возможностей программы sendmail |
| use_cw_file | Да (серверы) | Перечисляет узлы, на которые можно посылать сообщения |
| redirect | Возможно (серверы) | Перенаправляет письма при перемещении пользователей |
| always_add_domain | Да | Полностью определяет имена компьютеров, если сервер UA этого не сделал |
| access_db | Возможно (серверы) | Заставляет базу данных задерживать отправление почты |
| virtusertable | Возможно (серверы) | Включает псевдонимы доменов (виртуальные домены) |
| ldap_routing | Возможно (серверы) | Маршрутизирует входящую почту с помощью протокола LDAP |
| MASQUERADE_AS | Да | Маскирует всю почту так, будто она исходит из одного источника |
| EXPOSED_USER | Да | Перечисляет пользователей, которых не следует маскировать |
| MAIL_HUB | Возможно (серверы) | Задает почтовый сервер, предназначенный для входящей почты |
| SMART_HOST | Возможно (клиенты) | Задает почтовый сервер, предназначенный для исходящей почты |

### Макрос OSTYPE

Файл OSTYPE содержит разнообразную информацию от производителей, например ожидаемое местонахождение файлов, связанных с почтовой системой, пути к командам, которые необходимы для программы **sendmail**, флаги для программ рассылки и т.д. Список всех переменных, которые можно определить в файле OSTYPE, перечислены в файле **cf/README**.[10]

Каждая из рассмотренных нами операционных систем, за исключением системы SUSE, содержит соответствующий файл OSTYPE из дистрибутивного пакета **sendmail**. Вместо этого система SUSE содержит свой собственный файл **suse_linux.m4**.

Этот файл длинный (более 80 строк, по сравнению с пятью строками в сопоставимом файле **linux.m4**) и содержит многочисленные примитивы FEATURE наряду с другими макросами, которые обычно можно найти в главном конфигурационном файле сайта (файле с расширением **.mc**), но не в файле OSTYPE. Таким образом, реальная конфигурация скрыта от системного администратора — в этом есть и преимущества, и недостатки, но мы не рекомендуем эту практику.

[10] Так где же все-таки определен макрос *OSTYPE*? В одном из файлов в каталоге **cf/m4**, который таинственным образом добавляется к вашему конфигурационному файлу при запуске сценария **Build**.

### Макрос DOMAIN

Директива DOMAIN позволяет указывать всю общую информацию о сайте в одном месте (**cf/domain/*имя_файла*.m4**), а затем включать в каждый конфигурационный файл компьютера.

```
DOMAIN(`filename')
```

### Макрос MAILER

Этот макрос следует включать в каждый агент доставки, который вы хотите использовать. Полный список программ рассылки можно найти в каталоге **cf/mailers**, но обычно необходимы только local, smtp и, возможно, cyrus. Строки, содержащие макрос MAILER, обычно являются последними в файле **.mc**.

### Макрос FEATURE

Рассматриваемый макрос позволяет реализовать большое количество сценариев (по последним данным, 56!), включая файлы препроцессора **m4** из каталога **feature**. Его синтаксическая конструкция выглядит следующим образом.

```
FEATURE(keyword, arg, arg, ...)
```

Здесь аргумент *keyword* соответствует файлу *keyword*.**m4** в каталоге **cf/feature**. Количество остальных аргументов не должно превышать девяти.

### Функциональная возможность use_cw_file

Внутренний класс w программы **sendmail** (отсюда имя **cw**) содержит имена всех локальных компьютеров, к которым есть доступ из данного узла и может доставляться почта. Эта функциональная возможность позволяет доставлять почту узлам, построчно перечисленным в списке, находящемся в файле **/etc/mail/local-host-names**. Эту функциональную возможность активизирует конфигурационный файл.

```
FEATURE(`use_cw_file')
```

Клиентская машина на самом деле не нуждается в этой возможности, если у нее есть псевдоним. Файл **local-host-names** должен содержать имена всех локальных узлов и виртуальных доменов, которым можно доставлять почту, включая сайты, резервные записи MX которых ссылаются на ваш компьютер.

Если эта функциональная возможность не подключена, то программа **sendmail** выполняет локальную доставку почты, только если она адресована компьютеру, на котором выполняется сама программа **sendmail**.

Если вы добавили в сеть новый компьютер, то должны добавить его имя в файл **local-host-names** и послать сигнал HUP программе **sendmail**, чтобы изменения вступили в силу. К сожалению, программа **sendmail** считывает этот файл только при запуске.

### Функциональная возможность redirect

Когда люди покидают вашу организацию, вы обычно либо пересылаете им почту, либо возвращаете ее отправителю с сообщением об ошибке. Функциональная возможность redirect позволяет реализовать более элегантное решение для возврата почты.

Если Джо Смит окончил университет oldsite.edu (регистрационное имя smithj) и перешел в организацию newsite.com (регистрационное имя joe), то, включив функциональную возможность redirect с помощью примитива

```
FEATURE(`redirect')
```

и добавив строку

```
smithj: joe@newsite.com.REDIRECT
```

в файл **aliases** на сайте `oldsite.edu`, вы сможете возвращать почту, адресованную пользователю `smithj`, отправителю вместе с сообщением об ошибке, в котором указан новый адрес получателя `joe@newsite.com`. Само письмо автоматически не перенаправляется.

### *Функциональная возможность always_add_domain*

Эта функциональная возможность задает полное определение всех почтовых адресов. Ее следует использовать постоянно.

### *Функциональная возможность access_db*

С помощью этой функциональной возможности можно управлять задержкой и другими правилами работы с почтой. Обычно данные, которые управляют этой функциональной возможностью, либо поступают от сервера LDAP, либо хранятся в текстовом файле **/etc/mail/access**. Во втором случае текстовый файл должен быть преобразован в индексированный формат с помощью команды **makemap**, как показано выше. Для использования простого файла в конфигурационном файле следует указать инструкцию `FEATURE(`access_db')`; для работы с сервером LDAP следует использовать инструкцию `FEATURE(`access_db', `LDAP')`.[11]

Полем ключа для доступа к базе данных является IP-адрес сети или имя домена, сопровождаемое необязательным дескриптором, например `Connect:`, `To:` или `From:`. Поле значения указывает, что делать с сообщением.

Наиболее широко используются такие значения, как `OK` для приема сообщений, `RELAY` для разрешения задержки, `REJECT` для отказа с сообщением об ошибке общего характера или `ERROR:` "*код ошибки и сообщение*" для отказа с сообщением о конкретной ошибке. С помощью других значений можно установить более точный контроль. Рассмотрим фрагмент из файла **/etc/mail/access**.

```
localhost       RELAY
127.0.0.1       RELAY
192.168.1.1     RELAY
192.168.1.17    RELAY
66.77.123.1     OK
fax.com         OK
61              ERROR:"550 We don't accept mail from spammers"
67.106.63       ERROR:"550 We don't accept mail from spammers"
```

### *Функциональная возможность virtusertable*

Эта функциональная возможность позволяет задавать псевдонимы доменов для входящей корреспонденции, размещать несколько виртуальных доменов на одной машине и использовать их для веб-хостинга. Поле ключа таблицы содержит либо адрес электронной почты (*`user@host.domain`*), либо спецификацию домена (*`@domain`*). Поле значения содержит локальный или внешний адрес электронной почты. Если ключом является домен, то значение может либо передавать поле `user` как переменную `%1`, либо направлять почту другому пользователю. Рассмотрим несколько примеров.

[11] В этом примитиве используется схема LDAP, определенная в файле **cf/sendmail.schema**; если вы хотите использовать другой файл схемы, укажите дополнительные аргументы в инструкции *FEATURE*.

```
@appliedtrust.com %1@atrust.com
unix@book.admin.com sa-book-authors@atrust.com
linux@book.admin.com sa-book-authors@atrust.com
webmaster@example.com billy.q.zakowski@colorado.edu
info@testdomain.net ausername@hotmail.com
```

Все ключи в левой части отображения данных должны быть перечислены в файле **cw**, т.е. **/etc/mail/local-host-names**, или находиться в списке VIRTUSER_DOMAIN. Если это условие не выполняется, то программа **sendmail** не сможет принять локальную почту и попытается найти адресата в Интернете. Однако записи DNS MX вернут программу **sendmail** назад на тот же самый сервер, и в результате рикошета вы получите сообщение "local configuration error" ("ошибка локальной конфигурации"). К сожалению, программа **sendmail** не может сообщить, что это сообщение на самом деле означает "ключа виртуального пользователя в файле cw нет".

### *Функциональная возможность ldap_routing*

Протокол LDAP (Lightweight Directory Access Protocol — облегченный протокол доступа к каталогам) может быть источником данных о псевдонимах или информации о маршрутизации почты, а также общих данных, хранящихся в таблицах. Файл **cf/README** содержит большой раздел о протоколе LDAP с множеством примеров.

Для того чтобы использовать сервер LDAP указанным выше способом, необходимо скомпилировать программу **sendmail** так, чтобы она включала поддержку протокола LDAP. Для этого в файл **.mc** следует добавить такие строки.

```
define(`confLDAP_DEFAULT_SPEC', `-h server -b searchbase')
FEATURE(`ldap_routing')
LDAPROUTE_DOMAIN(`my_domain')
```

Эти строки означают, что вы хотите использовать базу данных протокола LDAP для маршрутизации входящей почты, адресованной указанному домену. Параметр LDAP_DEFAULT_SPEC идентифицирует сервер LDAP и базовое имя LDAP для поисков. Протокол LDAP использует порт 389, если вы не указали явным образом другой порт, добавив в инструкцию define параметр -p *порт_ldap*.

Программа **sendmail** использует значения двух дескрипторов в базе данных LDAP.

- mailLocalAddress — для адресата входящей почты.
- mailRoutingAddress — для пункта назначения, в который должна быть доставлена почта.

Кроме того, программа **sendmail** использует дескриптор mailHost, который направляет почту обработчику, заданному в поле MX для указанного узла. Адресом получателя остается значение дескриптора mailRoutingAddress.

Записи базы данных LDAP позволяют использовать шаблонный символ @*domain*, который перенаправляет почту, адресованную кому-либо в указанном домене (как и параметр virtusertable).

По умолчанию почта, адресованная получателю user@host1.mydomain, сначала направляется пользователю адресата user@host1.mydomain. Если такого адресата нет, программа **sendmail** может попытаться выполнить поиск в домене @host1.mydomain, а не использовать адрес user@mydomain. Включив строку

```
LDAPROUTE_EQUIVALENT(`host1.mydomain')
```

можно также проверить ключи user@mydomain и @mydomain. Эта функциональная возможность допускает использование единой базы данных для маршрутизации почты

на сложных сайтах. Вы также можете использовать записи для разделов LDAPROUTE_EQUIVALENT из файла, что делает это свойство еще более полезным.

Синтаксис этой инструкции выглядит следующим образом.

```
LDAPROUTE_EQUIVALENT_FILE('filename')
```

Кроме того, свойство ldap_routing позволяет более подробно описать схему LDAP, чем в разделе *+detail*, и точнее управлять обработкой имен адресатов. Как всегда, более подробную информацию следует искать в файле **cf/README**.

### Маскирующие функциональные возможности

Адрес электронной почты часто используется в качестве имени пользователя, узла и домена, но многие организации не хотят демонстрировать имена своих узлов в Интернете. Макрос **MASQUERADE_AS** позволяет скрыть все машины за единственной сущностью. Вся появляющаяся почта будет исходить от указанного компьютера или домена. Это очень удобно для обычных пользователей, но системные пользователи, такие как корневой пользователь, не должны участвовать в маскировке.

Например, последовательность инструкций

```
MASQUERADE_AS(`atrust.com')
EXPOSED_USER(`root')
EXPOSED_USER(`Mailer-Daemon')
```

проштамповывает почту так, будто она исходит от адреса user@atrust.com, если она не была послана корневым пользователем или почтовой системой; в этих случаях почта должна иметь истинный адрес отправителя. Инструкция MASQUERADE_AS — это верхушка айсберга, за которой скрываются десятки вариантов и исключений. Функциональные возможности allmasquerade и masquerade_envelope (в сочетании с MASQUERADE_AS) скрывают требуемую часть локальной информации. Детали см. в файле **cf/README**.

### Макросы MAIL_HUB и SMART_HOST

Маскировка изменяет заголовки и, возможно, конверт, чтобы вся почта выглядела исходящей от одного компьютера или домена. Однако большинство организаций *действительно* желают, чтобы вся почта исходила от одного компьютера или поступала на один компьютер, чтобы можно было контролировать поток вирусов, спама и секретов компании. Этот контроль можно обеспечить с помощью сочетания записей MX в системе DNS, макроса **MAIL_HUB** для входящей почты и макроса **SMART_HOST** для исходящей почты.

🕮 Более подробная информация о записях DNS MX приведена в разделе 17.8.

Например, в архитектурной диаграмме из раздела 20.4 записи MX могут направлять входящую электронную почту из Интернета транспортному агенту в демилитаризованной зоне. После проверки того факта, что полученная почта не содержит вирусы и спам и была направлена правильному локальному пользователю, почту можно передать с помощью инструкции define внутреннему транспортному агенту, выполняющему маршрутизацию.

```
define(`MAIL_HUB', `smtp:routingMTA.mydomain')
```

🕮 Функциональная возможность nullclient рассматривается в следующем разделе.

Аналогично клиентские компьютеры могут направлять свою почту компьютеру SMART_HOST, назначенному в конфигурации с помощью свойства nullclient. Компью-

тер SMART_HOST затем может профильтровать вирусы и спам, чтобы почта из вашей сети не просочилась в Интернет.

Синтаксис инструкции SMART_HOST аналогичен синтаксису инструкции MAIL_HUB, причем агентом доставки тоже является relay.

```
define(`SMART_HOST', `smtp:outgoingMTA.mydomain')
```

Ту же самую машину можно использовать как сервер для входящей и исходящей почты. Инструкции SMART_HOST и MAIL_HUB должны разрешать передачу: первая — от клиентов внутри вашего домена, а вторая — от транспортного агента в зоне DMZ.

## Конфигурация клиентов

Если ваша организация следует принципам, описанным в разделе 20.4, то большинство ваших компьютеров необходимо конфигурировать как клиентов, желающих посылать исходящую почту, генерируемую пользователями, и не получать почту вообще. Одна из функциональных возможностей программы **sendmail** под названием nullclient предназначена именно для таких ситуаций. Она создает конфигурационный файл, который перенаправляет всю почту на центральный концентратор через сервер SMTP. Содержание всего конфигурационного файла, следующее за строками с макросами VERSIONID и OSTYPE, сводится к следующим строкам.

```
FEATURE(`nocanonify')
FEATURE(`nullclient', `mailserver')
EXPOSED_USER(`root')
```

Здесь *mailserver* — это имя центрального концентратора. Параметр nocanonify сообщает программе **sendmail**, что она не должна выполнять поиск в системе DNS или перезаписывать адреса с полностью определенными именами доменов. Всю эту работу будет выполнять почтовый сервер. Этот параметр похож на параметр SMART_HOST и предполагает, что клиент применяет к почтовому серверу инструкцию MASQUERADE_AS. Инструкция EXPOSED_USER исключает корень из маскировки и открывает возможности для отладки.

Компьютер *mailserver* должен разрешать ретрансляцию от своих нулевых клиентов. Это разрешение предоставляется примитивом access_db, описанным выше. Нулевой клиент должен иметь соответствующую запись MX, ссылающуюся на компьютер *mailserver*, а также содержаться в файле **cw** на компьютере *mailserver* (который обычно называется **/etc/mail/local-host-names**). Эти настройки позволяют компьютеру *mailserver* принимать почту для клиента.

Если пользовательские агенты на клиентской машине могут использовать порт 587 для отправки почтовых сообщений, то программа **sendmail** должна выполняться как агент доставки (без флага **-bd**). В противном случае вы можете выполнить программу **sendmail** в режиме демона (**-bd**), задав параметр конфигурации DAEMON_OPTIONS так, чтобы прослушивать соединения только в интерфейсе замыкания (loopback interface).

В операционной системе SUSE есть эталонный файл **/etc/mail/linux.nullclient.mc** для нулевого клиента. Запишите в него имя своего почтового сервера, создайте файл **sendmail.cf**, и готово!

## Параметры конфигурации

Параметры файла конфигурации задаются с помощью команды define, относящейся к препроцессору **m4**. Полный список параметров, доступных как переменные препро-

цессора **m4** (вместе с их значениями, заданными по умолчанию), приведен в файле **cf/README**.

Значения по умолчанию подходят для типичной организации, не страдающей от паранойи и не слишком обеспокоенной вопросами производительности. Параметры, заданные по умолчанию, предназначены для защиты от спама: они отключают ретрансляцию и требуют, чтобы адреса были полностью определенными, а домены отправителей соответствовали их IP-адресам. Если ваш почтовый концентратор слишком загружен и обслуживает слишком много списков рассылки, может потребоваться более тонкая настройка.

Параметры, которые приходится настраивать чаще всего, перечислены в табл. 20.11 (это лишь около 10% из почти 175 параметров конфигурации). В скобках указаны их значения по умолчанию. В целях экономии места префикс `conf` в названиях параметров опущен. Например, параметр `FALLBACK_MX` в действительности называется `confFALLBACK_MX`. Таблица разделена на части, в соответствии с тем, для каких целей применяются те или иные параметры: общих, связанных с ресурсами, производительностью, безопасностью, спамом и т.д. Некоторые параметры попадают сразу в несколько категорий, однако мы указали их только в одном месте.

**Таблица 20.11. Основные параметры конфигурации**

| | Параметр | Описание (значение по умолчанию) |
|---|---|---|
| Ресурсы | `MAX_DAEMON_CHILDREN` | Максимальное количество порожденных процессов [a] (ограничений нет) |
| | `MAX_MESSAGE_SIZE` | Максимальный размер в байтах одного сообщения (ограничений нет) |
| | `MIN_FREE_BLOCKS` | Минимальное пространство в файловой системе для хранения поступающей почты (100) |
| | `TO_`*имя* | Время тайм-аута для разных ситуаций (варьируется) |
| Производительность | `DELAY_LA` | Показатель средней загруженности, при котором доставка почты замедляется (0 — нет ограничений) |
| | `FALLBACK_MX` | См. описание в разделе 20.12 (значение по умолчанию отсутствует) |
| | `FAST_SPLIT` | Подавляет поиск записей MX при сортировке адресов получателей и распределении их по очередям (1 — включено) |
| | `HOST_STATUS_DIRECTORY` | См. описание в разделе 20.12 (значение по умолчанию отсутствует) |
| | `MCI_CACHE_SIZE` | Количество открытых кешированных TCP-соединений (2) |
| | `MCI_CACHE_TIMEOUT` | Время, в течение которого кешированные соединения остаются открытыми (5m) |
| | `MIN_QUEUE_AGE` | Минимальный период времени, в течение которого сообщение должно находиться в очереди; позволяет улучшить обработку очереди на перегруженном компьютере (0) |
| | `QUEUE_LA` | Показатель средней загруженности, при котором сообщения помещаются в очередь, а не отправляются немедленно (8*`число_процессоров`) |
| | `REFUSE_LA` | Показатель средней загруженности, при котором почта перестает приниматься (12*`число_процессоров`) |

*Окончание табл. 20.11*

| | Параметр | Описание (значение по умолчанию) |
|---|---|---|
| Безопасность/спам | AUTH_MECHANISMS | Список механизмов аутентификации по протоколу SMTP для библиотеки Cyrus SASL [6] |
| | CONNECTION_RATE_THROTTLE | Предотвращает атаки типа "отказ от обслуживания", ограничивая скорость, при которой разрешается устанавливать почтовое соединение (ограничений нет) |
| | DONT_BLAME_SENDMAIL | Отменяет ряд функций программы **sendmail**, связанных с безопасностью и проверкой файлов; не изменяйте значение этой опции без особой необходимости (`safe`) |
| | MAX_MIME_HEADER_LENGTH | Максимальный размер заголовков MIME (ограничений нет) [a] |
| | MAX_RCPTS_PER_MESSAGE | Препятствует отправке спама; обработка избыточного числа получателей приостанавливается и генерируется временное сообщение об ошибке (число получателей не ограничено) |
| | PRIVACY_FLAGS | Налагает ограничения на информацию, выдаваемую по протоколу SMTP (`authwarnings`) |
| Разное | DOUBLE_BOUNCE_ADDRESS | Перехватывает множество спама; некоторые организации используют каталог /dev/null, но это может привести к серьезным проблемам (почтовый сервер) |
| | LDAP_DEFAULT_SPEC | Спецификация базы данных LDAP, включая имя компьютера, на котором запущен сервер, и номер порта (не определена) |

[a] Точнее говоря, это максимальное количество дочерних процессов, которые можно выполнить одновременно. При достижении предельного значения программа **sendmail** отказывается принимать запросы. Эта опция позволяет предотвратить атаки типа "отказ от обслуживания".

[б] Значение по умолчанию таково: "EXTERNAL GSSAPI KERBEROS_V4 DIGEST-MD5 CRAM-MD5". Не добавляйте "PLAIN LOGIN", поскольку пароль передается открытым текстом. Если соединение также не защищено с помощью криптографического протокола SSL, то изнутри все может выглядеть благополучно, а со стороны Интернета — нет.

[в] Эта опция защищает буфер пользовательского агента от переполнения. Рекомендуемое значение — "256/128", что означает "256 байт на заголовок и 128 байт на каждый параметр заголовка".

## Средства программы sendmail для борьбы со спамом

Программа **sendmail** имеет разнообразные возможности для борьбы со спамом и вирусами.

- *Правила управления открытой ретрансляцией*, под которой понимается использование почтового сервера для отправки сообщения одним сторонним пользователем другому, тоже стороннему. Спамеры часто прибегают к ретрансляции, пытаясь таким способом замаскировать действительного адресата сообщения и тем самым избежать обнаружения со стороны своих собственных провайдеров. Это также позволяет им перекладывать финансовую и вычислительную нагрузку на чужие плечи.
- *База доступа*, позволяющая фильтровать почту согласно адресам, содержащимся в базе. Это напоминает применение брандмауэра для электронной почты.
- *Черные списки* открытых ретрансляторов и известных спамеров, проверяемые программой **sendmail**.
- *Дроссели, которые могут замедлять прием почты при обнаружении неправильного поведения.*

- *Проверка заголовков* и *фильтрация входящей почты* посредством универсальной библиотеки фильтров **libmilter**. Эта библиотека позволяет сканировать заголовки и содержимое сообщений и отбрасывать сообщения, удовлетворяющие определенным критериям.

Эти средства в сочетании с другими технологиями, такими как серые списки, сканирование содержимого с помощью интерфейса **amavisd-new** и новые записи DNS для аутентификации электронной почты, описанными в разделе 20.6, создают эффективную защиту от спама.

## Ретрансляция

Программа **sendmail** получает входящую почту, просматривает адреса конверта, принимает решение о том, куда переслать сообщение, после чего отправляет его в соответствующий пункт назначения. Это может быть не только локальный получатель, но и другой транспортный агент, расположенный далее по цепи доставки. Когда входящее сообщение не имеет локального получателя, транспортный агент, обрабатывающий его, играет роль *ретранслятора* (relay).

До появления программы **sendmail** версии 8.9 по умолчанию включался режим "беспорядочной" рассылки (известный как *открытая ретрансляция*). Программа **sendmail** принимала сообщения через порт 25 и пыталась осуществить доставку оптимальным способом. В те времена считалось, что Интернет — дружественная среда.

Только узлам, помеченным ключевым словом RELAY в базе доступа (см. выше), а также узлам, перечисленным в файле **/etc/mail/relay-domains**, разрешается предоставлять почту для ретрансляции. Впрочем, некоторые типы ретрансляции необходимы. Как определить, какое сообщение следует ретранслировать, а какое — отклонить? Ретрансляция нужна лишь в двух случаях.

- *Когда транспортный агент выступает в качестве шлюза для узлов, недоступных никаким другим способом.* Это могут быть периодически выключаемые компьютеры (PPP-узлы, персональные компьютеры, работающие под управлением Windows) и виртуальные узлы. В этом случае все получатели, которым требуется переслать сообщение, должны находиться на одном домене.
- *Когда транспортный агент является сервером исходящей почты для других узлов.* В этом случае имена и IP-адреса всех узлов-отправителей являются локальными (или, по крайней мере, заданы явно).

Любые другие случаи ретрансляции, скорее всего, свидетельствуют о плохой конфигурации сервера (исключение можно сделать в отношении мобильных пользователей). Чтобы избежать ретрансляции в первом из упомянутых вариантов, организуйте прием почты на центральном сервере (с использованием протокола POP или IMAP для клиентского доступа). Второй вариант должен быть всегда допустим, но только для локальных узлов. Лучше проверять IP-адреса, чем имена компьютеров, так как последние проще подделать; впрочем, программа **sendmail** следит за этим.

В программе **sendmail** ретрансляция запрещена по умолчанию, но с помощью специальных средств ее можно разрешить либо в полном объеме, либо в ограниченном и контролируемом режиме. Эти средства перечислены ниже. Их можно использовать, но мы рекомендуем проявлять максимум осторожности и не злоупотреблять ими. Безопаснее всего организовать ограниченную ретрансляцию с помощью средства access_db, рассматриваемого в следующем подразделе.

- FEATURE(`relay_entire_domain') — разрешает ретрансляцию для узлов текущего домена.
- RELAY_DOMAIN(`*домен*,...') — дополнительно разрешает ретрансляцию для указанных доменов.
- RELAY_DOMAIN_FILE(`*имя_файла*') — то же самое, но список доменов берется из файла.
- FEATURE(`relay_hosts_only') — влияет на работу макроса RELAY_DOMAIN и средства access_db.

Если благодаря макроконстантам SMART_HOST и MAIL_HUB почта направляется на конкретный почтовый сервер, придется сделать исключение. Этот сервер должен быть настроен на ретрансляцию всей почты, поступающей от локальных компьютеров.

```
FEATURE(`relay_entire_domain')
```

Если вы рассматриваете возможность разрешения ретрансляции в какой-то форме, обратитесь к документации программы **sendmail** в файле **cf/README**, чтобы ненароком не стать подручным спамеров. Если решите включить ретрансляцию, то обратитесь к одному из специализированных сайтов с просьбой проверить, что вы случайно не создали открытую ретрансляцию, в частности, можно обратиться на сайт spamhelp.org.

### *Использование черных списков*

Если необходимо блокировать почту каких-либо локальных пользователей или узлов, воспользуйтесь средством

```
FEATURE(`blacklist_recipients')
```

которому в базе доступа соответствуют записи следующих типов.

```
To:nobody@                  ERROR:550 Mailbox disabled for this user
To:printer.mydomain         ERROR:550 This host does not accept mail
To:user@host.mydomain       ERROR:550 Mailbox disabled for this user
```

Эти записи блокируют почту, приходящую на адрес пользователя **nobody** (на любом узле), а также адресованную сетевому принтеру и одному из пользователей конкретного компьютера. Дескриптор To: разрешает этим пользователям посылать сообщения (подобной возможностью обладают некоторые принтеры), но не принимать их.

Для того чтобы подключить черные DNS-списки для входящей почты, используется параметр dnsbl.

```
FEATURE(`dnsbl')
FEATURE(`enhdnsbl')
```

Эта функциональная возможность заставляет программу **sendmail** отклонять почту, приходящую от узла, IP-адрес которого указан в списке известных спамеров (SBL, XBL и PBL), поддерживаемых сайтом spamhaus.org. В других списках указаны адреса спамеров, работающих по коммутируемым линиям, и узлов, поддерживающих открытую ретрансляцию. Черные списки распространяются через службу DNS с помощью особого приема. Более подробное описание работы этой системы приведено в разделе 20.6.

Средство dnsbl может принимать третий аргумент, который задает требуемое сообщение об ошибке. Если третий аргумент пропущен, будет выдано фиксированное сообщение (из базы данных DNS, содержащей проверяемые записи).

Функциональную возможность dnsbl можно включать несколько раз, чтобы проверять разные списки злоумышленников.

## Дроссели, скорость и ограничения на количество соединений

В табл. 20.12 перечислено несколько команд программы **sendmail**, которые могут замедлить обработку почты, если поведение клиентов покажется подозрительным.

**Таблица 20.12. Основные параметры конфигурации**

| Примитив | Описание |
|---|---|
| BAD_RCPT_THROTTLE | Замедляет коллекционирование адресов спамерами |
| MAX_RCPTS_PER_MESSAGE | Откладывает доставку, если сообщение имеет слишком много получателей |
| ratecontrol | Ограничивает скорость входящих соединений |
| conncontrol | Ограничивает количество одновременных соединений |
| greet_pause | Задерживает ответ HELO, требует точного подчинения протоколу SMTP |

После того как счетчик no-such-login достигает предельного значения, заданного параметром BAD_RCPT_THROTTLE, программа **sendmail** выполняет односекундную задержку после получения каждой команды RCPT, замедляя сбор адресов спамерской программой. Для того чтобы задать это предельное значение равным 3, используйте следующую команду.

```
define(`confBAD_RCPT_THROTTLE', `3')
```

Параметр MAX_RCPTS_PER_MESSAGE заставляет отправителя поставить дополнительных получателей в конец очереди. Это простая форма серого списка для сообщений, имеющих подозрительно много получателей.

Параметры ratecontrol и conncontrol позволяют установить для каждого компьютера или каждой сети скорость приема входящих соединений и количество одновременно устанавливаемых соединений соответственно. Оба параметра можно задать в файле **/etc/mail/access**, чтобы указать пределы и домены, к которым они относятся. Первый параметр сопровождается дескриптором ClientRate: в поле ключа, а второй — дескриптором ClientConn:. Для того чтобы включить режим контроля скорости, вставьте следующие строки в свой файл с расширением **.mc**.[12]

```
FEATURE(`ratecontrol', `nodelay',`terminate')
FEATURE(`conncontrol', `nodelay',`terminate')
```

Затем добавьте в свой файл **/etc/mail/access** список компьютеров или сетей, подлежащих контролю, а также соответствующие предельные значения. Например, строки

```
ClientRate:192.168.6.17 2
ClientRate:170.65.3.4 10
```

ограничивают узлы 192.168.6.17 и 170.65.3.4 двумя новыми соединениями в минуту и десятью новыми соединениями в минуту соответственно. Строки

```
ClientConn:192.168.2.8 2
ClientConn:175.14.4.1 7
ClientConn: 10
```

устанавливают следующие пределы: два соединения для узла 192.168.2.8, семь — для узла 175.14.4.1 и десять — для одновременных соединений со всеми другими узлами.

Еще одним интересным параметром является greet_pause. Когда удаленный транспортный агент соединяется с вашим сервером **sendmail**, протокол SMTP заставляет

[12] Наряду с инструкцией FEATURE(`access_db').

его подождать и поприветствовать ваш сервер, прежде чем начать обмен сообщениями. Однако программы рассылки спама обычно мгновенно выдают команду EHLO/HELO. Это поведение частично объясняется плохой реализацией протокола SMTP в инструментах для рассылки спама, но это может экономить время для спамера. Как бы то ни было, такое поведение является подозрительным и называется "навязыванием" ("slamming").

Параметр `greet_pause` заставляет программу **sendmail** подождать указанный период времени в начале соединения, прежде чем поздороваться со новообретенным другом. Если удаленный транспортный агент не останавливается, чтобы правильно поздороваться и обменяться командами EHLO или HELO в назначенный момент времени для знакомства, программа **sendmail** регистрирует ошибку и отказывается от выполнения последующих команд, поступающих от удаленного транспортного агента.

Вы можете задать паузу для приветствия с помощью следующей записи в файле с расширением **.mc**.

```
FEATURE(`greet_pause', `700')
```

Эта строка устанавливает задержку в 700 миллисекунд в начале каждого нового соединения. Вы можете устанавливать задержки для узлов или сетей с помощью префикса `GreetPause:` в базе доступа, но большинство сайтов для этого использует общее значение для этого параметра.

## Конфигурирование почтовых фильтров в программе sendmail

Почтовые фильтры в общих чертах были описаны в разделе 20.5. Этот раздел посвящен вопросам конфигурирования почтовых фильтров в программе **sendmail**. Почтовые фильтры управляются директивами `INPUT_MAIL_FILTER` и `MAIL_FILTER` и контролируются опциями `MILTER_MACROS_*`, которые позволяют указывать, какие фильтры должны применяться на той или иной стадии SMTP-диалога. Например, инструкция

```
INPUT_MAIL_FILTER('filtername',
                  'S=mailer:/var/run/filtername.socket')
```

перенаправляет все входящие сообщения программе **/etc/mail/filtername** через сокет, указанный во втором аргументе.

Ниже приведен боле реалистичный пример использования почтовых фильтров для соединения с программой SpamAssassin через локальный доменный сокет и для проверки подписей DKIM с помощью программы **dkim-filter** через сокет TCP на порту 8699.

```
dnl # Enable SpamAssassin
INPUT_MAIL_FILTER(`spamassassin',
`S=local:/var/run/spamass-milter.sock, F=, T=C:15m;S:4m;R:4m;E:10m')
dnl # Enable DomainKeys and DKIM
INPUT_MAIL_FILTER(`dkim-filter', `S=inet:8699@127.0.0.1, T=R:2m')
define(`confMILTER_MACROS_CONNECT', `j, {daemon_name}')
define(`confMILTER_MACROS_ENVFROM', `i, {auth_type}')
```

Две последние инструкции устанавливают параметры, передаваемые почтовым фильтрам в начале сеанса связи и после команды MAIL FROM соответственно.

Для получения более подробной информации обратитесь к документации в файле **libmilter/README** или HTML-файлам в каталоге **libmilter/docs** дистрибутивного пакета **sendmail**. Файл **README** содержит обзор и простой пример фильтра, регистрирующего сообщения в файле. Файлы в каталоге **docs** описывают интерфейс библиотеки и содержат информацию о том, как использовать разные вызовы для сборки своих соб-

ственных фильтрующих программ. Отличным источником информации является также сайт milter.org.

## Соединение сканера amavisd и программы sendmail

**Amavisd** — это внешний коммерческий сканер вирусов и спама, описанный в разделе 20.6. В этом разделе показано, как его использовать в сочетании с программой **sendmail**.

Легче всего соединить программу **sendmail** и сканер **amavisd** с помощью двух почтовых серверов: один получает почту из Интернета и передает ее сканеру **amavisd**, а второй, работающий только в режиме постановки в очередь, получает отсканированные сообщения от сканера **amavisd** и передает их дальше либо для локальной доставки, либо в Интернет. Сканер **amavisd** в этой схеме расположен посредине, действуя как концентратор MAIL_HUB для входящей почты и узел SMART_HOST для исходящей.

К сожалению, эта схема сканирует сообщения после постановки писем в почтовую очередь, после того как программа **sendmail** уже приняла почту для доставки. Процедура использования сканера **amavisd** до постановки писем в почтовую очередь описана в файле **README.milter** в документации дистрибутивного пакета **amavisd-new**.

Основные строки конфигурации сервера, взаимодействующего с Интернетом, — это те строки, которые передают всю почту на прослушку порта 10024 с помощью сканера **amavisd**.

```
FEATURE(`stickyhost')
define(`MAIL_HUB', `esmtp:[127.0.0.1]')
define(`SMART_HOST', `esmtp:[127.0.0.1]')
define(`confDELIVERY_MODE', `q')
define(`ESMTP_MAILER_ARGS', `TCP $h 10024')
DAEMON_OPTIONS(`Name=receivingMTA')
```

Последняя строка намного облегчает отладку конфигурации, поскольку мы можем сказать, какой процесс (получения писем программой **sendmail**, передачи писем программой **sendmail** или процесс сканирования, выполняемый программой **amavisd**) зарегистрировал сообщения.

После сканирования программа **amavisd** передает сообщения процессу прослушивания порта 10025 (а не порта 25, как обычно) с помощью программы **sendmail**, работающей только в режиме постановки в очередь. С этого момента обработчик очереди либо выполнит полную локальную доставку, либо вернет сообщения в Интернет.

На передаточном сервере инструкция

```
DAEMON_OPTIONS(`Addr=127.0.0.1, Port=10025, Name=transmittingMTA')
```

сообщает программе **sendmail**, что она должна прослушивать порт 10025 в ожидании сообщений, поступающих от сканера **amavisd**. Эта программа должна регистрировать любую информацию или сообщение об ошибке, содержащее имя transmittingMTA, чтобы отличать его от сообщений, содержащих имя receivingMTA.

Существует много настроек (например, пределы, регулирующие производительность), позволяющих обеспечить согласованную работу двух экземпляров программы **sendmail**. Для того чтобы точно решить, какие проверки следует осуществить, какой процесс выполнить и в каком порядке они должны следовать друг за другом, необходимо хорошенько подумать.

Полезным источником информации является файл **README_FILES/README.sendmail-dual** в дистрибутивном пакете **amavisd-new**.

# 20.11. Безопасность и программа sendmail

С началом лавинообразного роста Интернета программы наподобие **sendmail**, принимающие произвольные входные данные и направляющие их локальным пользователям, в файлы или в интерпретаторы команд, превратились в излюбленные мишени для хакеров. В настоящее время программа **sendmail**, наряду со службой DNS и даже протоколом IP, начала "обзаводиться" встроенными средствами аутентификации и шифрования, позволяющими решать основные проблемы безопасности.

Программа **sendmail** поддерживает систему SMTP-аутентификации и шифрование по протоколу TSL (Transport Layer Security — безопасность транспортного уровня), ранее известному как протокол SSL (Secure Sockets Layer — протокол защищенных сокетов). Протокол TSL содержит шесть новых конфигурационных параметров, связанных с файлами сертификатов и ключей. Кроме того, в базе доступа появились новые операции сравнения, связанные с аутентификацией.

В этом разделе мы опишем модель безопасности программы **sendmail** и систему для защиты конфиденциальности. Затем кратко рассмотрим протоколы TLS и SASL (Simple Authentication and Security Layer — простой протокол аутентификации и защиты) и их применение в сочетании с программой **sendmail**.

Прежде чем довериться содержимому, скажем, файлов **.forward** или **aliases**, программа **sendmail** тщательно проверяет права доступа к ним. Как правило, подобные меры предосторожности действительно необходимы, но иногда все же требуется их ослабить. С этой целью в программе появился параметр `DontBlameSendmail`, название которого ("не осуждай **sendmail**") подсказывает системным администраторам, что после изменения опции действия программы станут небезопасными.

У параметра `DontBlameSendmail` очень много возможных значений (по последним подсчетам, 55). По умолчанию он равен `safe`. Полный список значений можно найти в файле **doc/op/op.ps** дистрибутива **sendmail** или в книге О'Рейли, посвященной программе **sendmail**. Впрочем, лучше всего оставить этот параметр заданным по умолчанию.

## Владельцы файлов

Программа **sendmail** различает трех специальных пользователей: `DefaultUser`, `RunAsUser` и `TrustedUser`.

По умолчанию все агенты доставки работают от имени пользователя `DefaultUser`, если не установлены специальные флаги. При наличии в файле **/etc/passwd** учетной записи **mailnull**, **sendmail** или **daemon** именно эта запись будет соответствовать пользователю `DefaultUser`. В противном случае пользователю будут соответствовать значения UID и GID, равные 1. Мы рекомендуем применять учетную запись **mailnull**. Добавьте ее в файл **/etc/passwd** со звездочкой в поле пароля, с пустым полем идентификатора команд, с пустым полем начального каталога и группой **mailnull**. Понадобится также добавить запись **mailnull** в файл **/etc/group**. Этой учетной записи не должны принадлежать никакие файлы. Если программа **sendmail** не выполняется с правами суперпользователя, для агентов доставки должен быть установлен бит setuid.

Если задан пользователь `RunAsUser`, программа **sendmail** работает от его имени, игнорируя пользователя `DefaultUser`. Если программа запускается с установленным битом setgid (идентификатор группы меняется на **smmsp**), то агент подачи почты (программа **sendmail**) передает транспортному агенту (другой экземпляр **sendmail**) сообщения по протоколу SMTP. Транспортный агент **sendmail** не имеет установленного бита setuid, но вызывается с правами корневого пользователя из сценариев запуска системы.

Пользователю RunAsUser соответствует значение UID, которое присваивается программе **sendmail** при открытии сокета, подключенного к порту 25. Привилегированные порты (их номера не превышают 1024) могут открываться только суперпользователем, следовательно, программа **sendmail** первоначально должна работать от имени пользователя **root**. Тем не менее по завершении указанной операции программа может взять себе другое значение UID. Это уменьшает вероятность возможных злоупотреблений, если программа **sendmail** запускается обманным путем. Не применяйте средство RunAsUser на компьютерах, где есть другие пользовательские учетные записи и службы; оно предназначено для брандмауэров и бастионных узлов.[13]

По умолчанию программа продолжает выполняться от имени пользователя **root**. Если же учетная запись RunAsUser не эквивалентна суперпользователю, придется многое изменить. Пользователю RunAsUser должна принадлежать очередь почтовых сообщений, он должен иметь возможность читать все подключаемые файлы и файлы баз данных, запускать нужные программы и т.д. На поиск всех файлов и каталогов, владелец которых должен быть изменен, может уйти несколько часов.

Пользователь TrustedUser может владеть файлами баз данных и псевдонимов. Ему разрешается запускать демон и перестраивать файл **aliases**. Этот пользователь необходим для поддержки графических надстроек к программе **sendmail**, которые вынуждены предоставлять ограниченный административный контроль определенным пользователям. Обязательно контролируйте учетную запись, соответствующую пользователю TrustedUser, поскольку через нее можно легко получить права суперпользователя. Пользователь TrustedUser не соответствует классу TRUSTED_USERS, который определяет, кто имеет право менять заголовок "From" сообщений[14].

## Права доступа

Для безопасности программы **sendmail** большое значение имеют права доступа к определенным файлам и каталогам. Установки, приведенные в табл. 20.13, считаются безопасными.

**Таблица 20.13. Владельцы и права доступа для каталогов, связанных с программой sendmail**

| Каталог | Владелец | Режим | Назначение/содержимое |
|---|---|---|---|
| **/var/spool/clientmqueue** | **smmsp:smmsp** | 770 | Почтовая очередь для начальной подачи почты |
| **/var/spool/mqueue** | RunAsUser | 700 | Очередь почтовых сообщений |
| **/, /var, /var/spool** | **root** | 755 | Родительские каталоги подкаталога **mqueue** |
| **/etc/mail/*** | TrustedUser | 644 | Таблицы, конфигурационный файл, файлы псевдонимов |
| **/etc/mail** | TrustedUser | 755 | Родительский каталог для разных файлов |
| **/etc** | **root** | 755 | Путь к каталогу **mail** |

Программа **sendmail** не будет читать файлы с "ослабленными" правами доступа (например, файлы, открытые для записи, а также файлы, находящиеся в каталогах, которые открыты для записи). Программа **sendmail** тщательно проверяет путь к любому фай-

[13] Бастионные узлы — это специально укрепленные узлы, предназначенные для отражения атаки и размещенные в зоне DMZ или за пределами брандмауэра.

[14] Средство TRUSTED_USERS обычно применяется для поддержки программ, работающих со списками рассылки.

лу псевдонимов, что иногда сказывается на управлении списками рассылки. Для того чтобы проверить настройки, запустите команду **sendmail -v -bi**. Флаг **-bi** заставляет программу проинициализировать базу данных псевдонимов и выдать предупреждение, если права доступа не подходят.

Операционная система Solaris имеет удобную программу **check-permissions**, которая соответствует стандартам безопасности программы **sendmail** и сообщает об опасных путях и файлах. Она прослеживает директивы include в файлах псевдонимов и **.forward**. Кроме того, она может проверять либо вызывающих пользователей, либо всех пользователей, в зависимости от флагов командной строки.

Программа **sendmail** больше не читает файл **.forward**, если число ссылок на него больше единицы, а путь к каталогу небезопасен (права доступа ослаблены). На этом не так давно попалась Эви Немет, когда ее файл **.forward**, представляющий собой жесткую ссылку либо на файл **.forward.to.boulder**, либо на файл **.forward.to.sandiego**, перестал поддерживать пересылку почты от небольшого узла, с которого ей иногда приходили сообщения. Прошли месяцы, прежде чем Эви поняла, что это происходит по ее вине и фраза "Я никогда не получала от вас почту" не является оправданием.

Отключить многие ограничения на доступ к файлам можно с помощью опции `DontBlameSendmail`. Но не забывайте о безопасности.

## Безопасная пересылка почты в файлы и программы

В качестве агента программной доставки мы рекомендуем применять утилиту **smrsh**, а не **/bin/sh**; агентом локальной доставки лучше назначить утилиту **mail.local**, а не **/bin/mail**. Обе утилиты входят в дистрибутив **sendmail**. Для их активизации необходимо добавить в **mc**-файл следующие директивы.

```
FEATURE(`smrsh', `каталог')
FEATURE(`local_lmtp', `каталог')
```

Если каталоги не указаны, по умолчанию принимается каталог **/usr/libexec**. Можно воспользоваться опцией `confEBINDIR`, чтобы изменить стандартный каталог исполняемых файлов. Найти агентов доставки вам поможет табл. 20.14.

**Таблица 20.14. Местоположение ограниченных агентов доставки программы sendmail**

| Операционная система | smrsh | mail.local | sm.bin |
|---|---|---|---|
| **sendmail** | **/usr/libexec** | **/usr/libexec** | **/usr/adm** |
| Ubuntu | **/usr/lib/sm.bin** | **/usr/lib/sm.bin** | **/usr/adm** |
| SUSE | **/usr/lib/sendmail.d/bin** | **/usr/lib/sendmail.d/bin** | - |
| Red Hat | **/usr/sbin** | - | **/etc/smrsh** |
| Solaris | **/usr/lib** | **/usr/lib** | **/var/adm** |
| HP-UX | **/usr/sbin** | - | **/var/adm** |
| AIX | **/usr/sbin** | - | **/var/adm** [a] |

[a] Этот каталог не существует ни в системе HP-UX, ни в системе AIX, поэтому его необходимо создать.

Утилита **smrsh** — это ограниченная оболочка, запускающая программы, находящиеся только в одном каталоге (по умолчанию — **/usr/adm/sm.bin**). Она игнорирует заданные пользователями имена каталогов и деревьев, выполняя поиск требуемых команд

в собственном безопасном каталоге. Утилита **smrsh** также блокирует некоторые метасимволы интерпретатора команд, в частности '<' — оператор переадресации входного потока. В каталоге **sm.bin** допускается наличие символических ссылок, поэтому создавать копии программ не потребуется. Программа **vacation** — подходящий кандидат для включения в каталог **sm.bin**. Не размещайте в нем программу **procmail**; она небезопасна.

Приведем несколько примеров команд вместе с их возможными интерпретациями утилитой **smrsh**.

```
vacation eric                 # Выполняется команда /usr/adm/sm.bin/vacation
cat /etc/password             # Отклоняется, т.к. cat нет в каталоге sm.bin
vacation eric < /etc/password # Отклоняется, т.к. оператор < не разрешен
```

Опция `SafeFileEnvironment` программы **sendmail** контролирует, куда можно записывать сообщения, если в файле **aliases** или **.forward** задано перенаправление почты в файл. Это опция заставляет программу осуществить системный вызов **chroot**, вследствие чего корневым каталогом файловой системы станет не **/**, а **/safe** или любой другой указанный в опции каталог. Например, если в файле псевдонимов задается перенаправление почты в файл **/etc/passwd**, то на самом деле сообщения будут записываться в файл **/safe/etc/passwd**.

Опция `SafeFileEnvironment` защищает также файлы устройств, каталоги и другие специальные файлы, позволяя записывать почту только в обычные файлы. Это полезно не только в плане улучшения безопасности, но и с точки зрения борьбы с ошибками пользователей. В некоторых системах значение этой опции задается равным **/home**, что открывает доступ к начальным каталогам пользователей, но оставляет системные файлы вне досягаемости.

Агенты доставки тоже могут запускаться в виртуальном каталоге.

## Опции безопасности

В программе **sendmail** есть опции безопасности, которые определяют следующее:

- какие сведения о системе можно узнать из внешнего мира по протоколу SMTP;
- что требуется от узла на противоположном конце SMTP-соединения;
- могут ли пользователи просматривать или обрабатывать очередь почтовых сообщений.

В табл. 20.15 приведены текущие значения опций безопасности. Самую последнюю информацию можно узнать в файле **doc/op/op.ps** дистрибутива.

**Таблица 20.15. Значения переменной PrivacyOption**

| Значение | Интерпретация |
|---|---|
| `public` | Проверка конфиденциальности/безопасности не осуществляется |
| `needmailhelo` | Требуется SMTP-команда HELO (идентифицирует удаленный узел) |
| `noexpn` | Не допускается SMTP-команда EXPN |
| `novrfy` | Не допускается SMTP-команда VRFY |
| `needexpnhelo` | Не допускается раскрытие адреса (команда EXPN) без команды HELO |
| `needvrfyhelo` | Не допускается проверка адреса (команда VRFY) без команды HELO |
| `noverb`[a] | Не допускается "многословный" режим команды EXPN |
| `restrictmailq` | Только пользователи группы, которой принадлежит каталог **mqueue**, могут просматривать очередь сообщений |

*Окончание табл. 20.15*

| Значение | Интерпретация |
|---|---|
| `restrictqrun` | Только владелец каталога **mqueue** может обрабатывать очередь сообщений |
| `restrictexpand` | Ограничивается объем информации, выдаваемой при наличии флагов **-bv** и **-v**[6] |
| `noetrn`[в] | Не допускается асинхронная обработка очереди |
| `authwarnings` | К сообщениям добавляется заголовок "Authentication-Warning" (это установка по умолчанию) |
| `noreceipts` | Запрещается выдача кодов состояния доставки |
| `nobodyreturn` | При выдаче кода состояния доставки не возвращается тело сообщения |
| `goaway` | Отменяются все статусные SMTP-запросы (EXPN, VRFY и т.д.) |

[а] В "многословном" режиме команда EXPN отслеживает переадресацию в файле **.forward** и выдает дополнительную информацию о местонахождении пользовательской почты. Необходимо включать опцию `noverb` или, еще лучше, `noexpn` на любом компьютере, имеющем выход во внешний мир.

[б] Если только программа не выполняется от имени пользователя **root** или **TrustedUser**.

[в] ETRN — это команда протокола ESMTP, которая используется узлами с коммутируемым доступом. Она запрашивает обработку очереди только для сообщений данного узла.

Мы рекомендуем сделать в **mc**-файле "консервативные" установки.

```
define(`confPRIVACY_FLAGS', ``goaway, authwarnings,
     restrictmailq, restrictqrun'')
```

По умолчанию установлен лишь параметр `authwarnings`. Обратите внимание на двойные кавычки: некоторые версии препроцессора **m4** требуют этого для предотвращения ненужной интерпретации запятых в списке значений. В системах Red Hat, Solaris и AIX по умолчанию принята установка `authwarnings`, в системах SuSE и Ubuntu — установки `authwarnings`, `needmailhelo`, `novrfy`, `noexpn` и `noverb`, а в системе HP-UX — `restrictqrun`, `goaway` и `authwarnings`. Самыми безопасными являются установки системы HP.

## Выполнение программы sendmail в виртуальном каталоге (для настоящих параноиков)

Те, кого беспокоит влияние программы **sendmail** на файловую систему, могут запускать ее в виртуальной среде. Создайте в этом каталоге минимальную файловую систему, включив в нее файл **/dev/null**, важные файлы каталога **/etc** (**passwd**, **group**, **resolv.conf**, **sendmail.cf**, таблицы баз данных, **mail/***), необходимые программе **sendmail** совместно используемые библиотеки, исполняемый файл **sendmail**, каталог очереди почтовых сообщений и журнальные файлы. Вероятно, вы захотите поэкспериментировать со списком. Для запуска программы **sendmail** в виртуальном окружении используйте команду **chroot**.

```
# sudo chroot /jail /usr/sbin/sendmail -bd -q30m
```

## Отражение атак типа "отказ от обслуживания"

Атаки типа "отказ от обслуживания" невозможно предотвратить, потому что нельзя заранее предсказать, является ли сообщение оружием атаки или нет. Злоумышленники могут поступать по-разному, например переполнять SMTP-порт ложными запросами на подключение, "забивать" разделы диска гигантскими сообщениями, засорять вы-

ходные соединения, бомбардировать систему почтовыми сообщениями и т.д. В программе **sendmail** имеются конфигурационные параметры, позволяющие ограничить влияние этих атак на систему, но в результате может пострадать и легитимная почта. Определенную помощь может оказать и новая библиотека функций фильтрации почты.

Параметр `MaxDaemonChildren` ограничивает число процессов **sendmail**. Он не позволяет программе захватить все вычислительные ресурсы системы, но зато дает возможность хакерам очень легко одолеть службу SMTP. Опция `MaxMessageSize` защищает каталог очереди от переполнения, но если задать значение опции слишком маленьким, жертвами могут оказаться обычные сообщения. (Следует уведомить пользователей о существующем пределе на размер сообщения, чтобы они не удивлялись, если письмо вдруг вернется обратно. Этот предел обычно устанавливается достаточно большим.) Параметр `ConnectionRateThrottle` задает предельно допустимое число соединений в секунду. Это может привести к небольшому замедлению работы программы **sendmail**. Наконец, опция `MaxRcptsPerMessage` определяет максимальное число получателей сообщения.

Программа **sendmail** всегда умела отказываться от приема сообщений (опция `REFUSE_LA`) и ставить сообщения в очередь (опция `QUEUE_LA`) в зависимости от уровня загруженности системы. Дополнительный параметр `DELAY_LA` позволяет обработку почты, но с меньшей скоростью. Подробности изложены в разделе 20.12.

Несмотря на все меры предосторожности, трудно предотвратить ситуацию, когда кто-то начинает бомбардировать систему письмами. Это может стать очень неприятной проблемой.

## SASL: простой протокол аутентификации и защиты

Программа **sendmail** поддерживает систему SMTP-аутентификации, которая определена в документе RFC2554. Она основана на технологии SASL (Simple Authentication and Security Layer — простой протокол аутентификации и защиты), представляющей собой систему совместного использования секретного ключа между двумя узлами (см. документы RFC4422 и RFC4752). В ней нужно явно задавать пары серверов, осуществляющих взаимную аутентификацию. Обычно она используется между пользовательскими агентами и агентами доставки или между агентами доставки и транспортными агентами внутри сайта.

SASL — это базовый механизм аутентификации, который может быть встроен в ряд других протоколов. Механизм SASL (это библиотека) основан на двух концепциях: авторизации идентификатора (например, регистрационного имени) и аутентификации идентификатора (например, пароля). Она может применять их для распределения прав доступа к файлам, паролям, сертификатам протокола Kerberos и т.д. Библиотека функций SASL состоит из двух частей: модуля аутентификации (например, регистрационного имени) и модуля шифрования (например, пароля). Для использования библиотеки SASL в сочетании с программой **sendmail** необходимо получить копию библиотеки Cyrus SASL по адресу: `ftp://ftp.andrew.cmu.edu/pub/cyrus-mail`.

## TLS: безопасность транспортного уровня

Еще одна система аутентификации и шифрования — TLS — определена в документе RFC3207. Она реализована в программе **sendmail** в виде расширения протокола SMTP под названием STARTTLS. Можно даже одновременно использовать протоколы SASL и TLS.

Протокол TLS труднее настроить, так как требуется поддержка со стороны центра сертификации. Можно заплатить компании VeriSign, которая выдаст вам необходимые

сертификаты (подписанные открытые ключи, идентифицирующие их предъявителя), или организовать собственный центр сертификации. В процессе ретрансляции почты и приема запросов от других узлов аутентификация осуществляется не на основе имени узла или его IP-адреса, а путем обмена ключами. Записи вида

```
TLS_Srv:secure.example.com ENCR:112
TLS_Clt:laptop.example.com PERM+VERIFY:112
```

в базе доступа указывают на то, что используется механизм STARTTLS. Почта, направляемая в домен `secure.example.com`, должна шифроваться, как минимум, 112-битовыми ключами. Почта, поступающая от узла `secure.example.com`, будет приниматься лишь в том случае, если клиент пройдет процедуру аутентификации.

Грег Шапиро (Greg Shapiro) и Клаус Ассман (Claus Assmann) из компании Sendmail, Inc., ведут прекрасные веб-страницы, посвященные настройке протоколов SASL и TLS в программе **sendmail**: `sendmail.org/~gshapiro` и `www.sendmail.org/~ca`. Ссылка **index** на сайте `~ca` особенно полезна.

# 20.12. Производительность программы sendmail

Программа **sendmail** имеет ряд конфигурационных опций, предназначенных для повышения производительности. Мы упоминали их ранее в главе, а теперь попытаемся более подробно описать наиболее важные из них. Эти опции и средства необходимо учитывать при развертывании крупномасштабной почтовой системы. Но, конечно, если пропускная способность системы должна составлять порядка миллиона сообщений в час, имеет смысл приобрести коммерческую версию программы **sendmail**, предлагаемую компанией Sendmail, Inc.

## Режимы доставки

Программа **sendmail** имеет четыре режима доставки: фоновый, интерактивный, с постановкой в очередь и отложенный. Каждый из них представляет собой определенный компромисс между временем задержки и пропускной способностью. В фоновом режиме сообщение доставляется немедленно, но для этого программа должна породить новый процесс. В интерактивном режиме тоже поддерживается немедленная доставка, но ее осуществляет тот же процесс, поэтому удаленный узел вынужден дожидаться результатов. В режиме очереди поступившее сообщение помещается в буфер, откуда позднее его извлекает обработчик очереди. Режим отложенной доставки аналогичен предыдущему, но в очередь заносятся также все операции поиска в таблицах, в DNS, в файлах **aliases** и **.forward**. Интерактивный режим применяется редко. Фоновый режим обеспечивает меньшее время задержки, а в режиме отложенной доставки и режиме очереди выше пропускная способность. Режим доставки задается с помощью опции `confDELIVERY_MODE`. По умолчанию принят фоновый режим.

## Группы очередей и разбивка конвертов

Группы очередей позволяют создавать несколько очередей для исходящей почты и управлять атрибутами каждой группы по отдельности. Группа очередей используется для разбивки конвертов, позволяя распределить конверт с множеством получателей (например, сообщение, направленное списку рассылки) между несколькими группами очередей. Группа очередей имеет несколько примитивов конфигурации. Подробности можно

найти в книге О'Рейли *Sendmail* или в файле **cf/README**. Это помогает решить проблему, связанную с нахождением слишком большого числа файлов в одном почтовом каталоге.

## Обработчики очередей

Программа **sendmail** порождает копии самой себя при транспортировке сообщений. Можно задавать, сколько копий программы должно выполняться в любой момент времени и даже сколько из них должно быть связано с каждой группой очередей. Это позволяет системным администраторам перераспределять нагрузку между программой **sendmail** и операционной системой на крупных почтовых концентраторах.

Число демонов **sendmail**, обрабатывающих каждую очередь, определяется тремя опциями.

- MAX_DAEMON_CHILDREN — задает общее число демонов **sendmail**, которые могут выполняться одновременно, включая те, что обрабатывают очередь, и те, что принимают входящую почту.
- MAX_QUEUE_CHILDREN — задает общее число одновременно выполняющихся обработчиков очереди.
- MAX_RUNNERS_PER_QUEUE — задает стандартный предел числа обработчиков одной очереди, если в определении группы отсутствует параметр Runners= (или R=).

## Контроль средней загруженности

Программа **sendmail** всегда умела отказываться от приема сообщений или помещать сообщения в очередь, когда показатель средней загруженности системы становится очень высоким. К сожалению, измерение средней загруженности осуществляется раз в минуту, что не позволяет с достаточной степенью точности контролировать потребление программой **sendmail** ресурсов системы. Новая опция DELAY_LA позволяет задать показатель средней загруженности, при котором программа должна замедлить свою работу: она будет делать секундную паузу между выполнением SMTP-команд текущего соединения и приемом нового запроса на подключение. По умолчанию опция равна нулю, т.е. режим отключен.

## Обработка недоставленных сообщений

Недоставленные сообщения, остающиеся в почтовой очереди, способны существенно снизить производительность почтового сервера. У программы **sendmail** есть ряд средств для борьбы с такими сообщениями. Наиболее эффективным из них является опция FALLBACK_MX, задающая пересылку сообщений на другой компьютер, если их не удалось доставить с первой попытки. Эта опция позволяет главному серверу заниматься доставкой почты с правильными адресами, перепоручив "проблемную" почту резервному серверу. Вторая опция, HOST_STATUS_DIRECTORY, позволяет сохранять информацию о состоянии удаленных узлов между запусками обработчиков очереди.

Опция FALLBACK_MX позволяет существенно повысить производительность узла с большими списками рассылки, где неизбежно попадаются временно или постоянно недоступные адреса. Значением опции должен быть адрес компьютера, обрабатывающего отложенную почту. Например, команда

```
define(`confFALLBACK_MX', `mailbackup.xor.com')
```

задает пересылку всех сообщений, которые не удалось доставить с первого раза, на центральный сервер `mailbackup.xor.com` для последующей обработки. Разрешается иметь несколько резервных серверов, если указанному узлу соответствует несколько записей MX в базе данных DNS.

Параметр `TO_INCONNECT` задает паузу перед первой попыткой связи и отправки сообщения. Если эта пауза установлена короткой, то транспортный агент будет перегружен излишней работой. Однако она позволяет главному серверу обработать большой список рассылки за одну попытку в течение рекордно короткого времени.

На резервном сервере можно воспользоваться опцией `HOST_STATUS_DIRECTORY` для решения проблемы повторных сбоев. Эта опция предписывает программе **sendmail** хранить статусный файл для каждого компьютера, на который посылается почта. На основании этой информации определяются приоритеты просмотра узлов при очередной обработке очереди. Это позволяет реализовать схему отрицательного кеширования и хранить статусную информацию в промежутках между запусками обработчиков очередей. Результатом становится повышение производительности серверов, обрабатывающих списки рассылки с большим числом неправильных адресов, хотя это и затратно с точки зрения файлового ввода-вывода.

В следующем примере для хранения статусной информации выделяется каталог **/var/spool/mqueue/.hoststat** (его нужно предварительно создать).

```
define(`confHOST_STATUS_DIRECTORY', `/var/spool/mqueue/.hoststat')
```

Если указать относительный путь к каталогу **.hoststat**, то он будет создан в каталоге очереди. Программа **sendmail** формирует собственную иерархию подкаталогов, основываясь на имени узла-получателя.

Например, если почту по адресу `evi@anchor.cs.colorado.edu` не удается доставить, то статусная информация будет занесена в файл **anchor** каталога **/var/spool/mqueue/.hoststat/edu./colorado./cs./**, так как наиболее приоритетная запись MX узла **anchor** ссылается на него самого. Если бы записи MX задавали перенаправление почты узла **anchor** на узел **foo**, то имя файла было бы **foo**, а не **anchor**.

Еще одна возможность повышения производительности связана с заданием минимального времени пребывания сообщения в очереди. Сообщение, которое не удалось доставить с первого раза, помещается в очередь и остается там до тех пор, пока не будет осуществлена повторная попытка. Обычно это делается в сочетании с флагами командной строки, задающими более частую обработку очереди (например, **-q5m**). Если обработчик очереди "зависает" на каком-то сообщении, то через 5 минут запускается другой. Вся очередь обрабатывается блоками, формируемыми в зависимости от того, какие сообщения превысили минимальное время пребывания в очереди. Таким образом, запуск программы **sendmail** с флагами **-bd -q5m** и включение опции

```
define(`confMIN-QUEUE_AGE', `27m')
```

в конфигурационном файле позволяют организовать более оперативную обработку почты.

## Настройка ядра

Если UNIX- или Linux-систему планируется использовать в качестве крупного почтового сервера, придется модифицировать ряд параметров сетевой конфигурации ядра. В табл. 20.16 приведены параметры, которые необходимо изменить в системе Linux на загруженном почтовом сервере, а также их рекомендуемые значения и значения, заданные по умолчанию.

**Таблица 20.16. Параметры ядра, которые требуется менять на крупных почтовых серверах**

| Переменная (относительно каталога /proc/sys) | По умолчанию | Рекомендуется |
|---|---|---|
| net/ipv4/tcp_fin_timeout | 180 | 30 |
| net/ipv4/tcp_keepalive_time | 7200 | 1800 |
| net/core/netdev_max_backlog | 300 | 1024 |
| fs/file_max | 4096 | 16384 |
| fs/inode_max | 16384 | 65536 |

Для того чтобы настроить параметры для системы Linux, примените команду оболочки **echo** к соответствующей переменной в файловой системе **/proc**. Общее описание этой процедуры приведено в разделе 14.11. Эти изменения можно сделать постоянными, выполнив команду **sysctl** или поместив соответствующие команды **echo** в сценарий оболочки, который выполняется во время загрузки.

Например, для того чтобы изменить значение тайм-аута FIN в протоколе TCP, можно выполнить следующую инструкцию.

```
linux$ sudo sh -c "echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout"15
```

Для настройки сетевых параметров в системах Solaris и HP-UX используется команда **ndd**. Реализация системы HP-UX хорошо задокументирована, и команда **ndd -h** (справка) дает точное описание каждой переменной, диапазона ее значений и значения, заданного по умолчанию. В системе Solaris команда **ndd** интерпретирует знак вопроса как попытку обратиться к документации, но перед знаком вопроса необходимо поставить обратную косую черту (**ndd \?**), чтобы защитить эту команду от оболочки. К сожалению, команда **ndd** в системе Solaris просто перечисляет настраиваемые переменные, а не описывает их.

Например, чтобы изменить значение тайм-аута FIN в системе HP-UX с помощью команды **ndd**, необходимо выполнить следующую команду.

```
hp-ux$ sudo ndd -set tcp_fin_what_ timeout 30000
```

Единица тайм-аута для команды **ndd** задается в миллисекундах, поэтому указано число 30000, а не 30. В системе Solaris эта переменная называется **tcp_fin_wait_2_flush_interval**. Интересно, что в справочной системе не указано никаких единиц измерения... но поисковая машина Google знает о них! Фактически этими единицами измерения являются миллисекунды, причем значение по умолчанию равно 675000. Более подробно команда **ndd** описана в разделах 14.13 (для системы Solaris) и 14.14 (для системы HP-UX).

Система AIX для настройки параметров сети использует команду **no**. Инструкция **sudo no -L** перечисляет настраиваемые переменные с указанием их минимального, максимального и текущего значений наряду с единицами их измерения. Если вы хотите, что изменения стали постоянными, используйте команду **no** с флагом **-p**. В этом случае ваши изменения сохранятся после перезагрузки.

[15] Если вы попытаетесь выполнить эту команду в виде ***sudo echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout***", то просто получите сообщение "permission denied", потому что оболочка попытается открыть результирующий файл до выполнения команды ***sudo***. Команда ***sudo*** должна применяться и к команде ***echo***, и к перенаправлению. Следовательно, необходимо создать корневую подоболочку, в которой выполняется вся команда: ***sudo sh -c "echo..."***.

Например, для задания параметра FIN_WAIT протокола TCP используется такая команда.

```
aix# sudo no -p -o tcp_finwait2=60
```

Единицей измерения этого параметра является половина секунды, поэтому значение 60 соответствует рекомендуемым 30 секундам. Команда **no** описана в разделе 14.15.

# 20.13. Сбор статистических данных, тестирование и отладка

Конфигурация, основанная на применении препроцессора **m4**, в определенной степени тестируется автоматически. Низкоуровневая отладка, в основном, не требуется. Единственное, что невозможно контролировать при помощи флагов отладки, — это смысловые ошибки. В процессе подготовки главы мы обнаружили несколько подобного рода ошибок в конфигурационных файлах. Иногда, к примеру, вызывалось средство без сопутствующего ему макроса (в частности, активизировалось средство masquerade_envelope, но не включался режим маскирования адресов с помощью макроса MASQUERADE_AS). Или конфигурация программы **sendmail** не соответствовала настройкам брандмауэра, который определял, какую почту разрешается пропускать.

Нельзя проектировать почтовую систему саму по себе. Ее необходимо согласовать с имеющимися записями MX базы данных DNS и настройками брандмауэра.

## Мониторинг очереди

С помощью команды **mailq** (эквивалент команды **sendmail -bp**) можно просматривать состояние сообщений, помещенных в очереди. Сообщение могут помещаться в очередь после доставки или при неудачной попытке доставки.

Команда **mailq** выводит на печать отчет о файлах **/var/spool/mqueue** в любой заданный момент. Результат позволяет определить, почему сообщение могло быть задержано. Если выясняется, что в системе увеличивается поток почтовых сообщений, системный администратор может отследить попытки программы **sendmail** разобрать этот завал.

По умолчанию существует две очереди: для сообщений, полученных через порт 25, и для сообщений, полученных через порт 587 (очередь подачи для клиентов). Клиентскую очередь можно увидеть с помощью команды **mailq -Ac**.

Ниже приведен типичный результат работы команды **mailq**. В отчете указано, что доставки ожидают три сообщения.

```
$ sudo mailq
/var/spool/mqueue (3 requests)
-----Q-ID----- --Size-- -----Q-Time----- ------------Sender/Recipient-------
k623gYYk008732    23217       Sat Jul 1 21:42          MAILER-DAEMON
      8BITMIME (Deferred: Connection refused by agribusinessonline.com.)
                                                 <Nimtz@agribusinessonline.com>
k5ULkAHB032374    279         Fri Jun 30 15:46 <randy@atrust.com>
            (Deferred: Name server: k2wireless.com.: host name lookup fa)
                                                 <relder@k2wireless.com>
k5UJDm72023576 2485 Fri Jun 30 13:13 MAILER-DAEMON
            (reply: read error from mx4.level3.com.)
                                                 <lfinist@bbnplanet.com>
```

Если вы считаете, что разбираетесь в ситуации лучше, чем программа **sendmail**, или просто хотите, чтобы программа **sendmail** попыталась немедленно повторить попытку доставки сообщений, попавших в очередь, можете заставить обработчик очереди выполнить команду **sendmail -q**. Если вы используете команду **sendmail -q -v**, то программа **sendmail** показывает результаты для каждой попытки доставки, что часто оказывается полезным для отладки.

Программа **sendmail** повторяет доставку каждому обработчику очереди по истечении заданного интервала (как правило, каждые 30 минут).

## Журнальная регистрация

Программа **sendmail** регистрирует сообщения об ошибках и своем состоянии через систему Syslog. Это делается от имени средства `mail` на уровнях от `debug` до `crit`. Все сообщения помечаются строкой "sendmail". Изменить эту установку можно с помощью флага командной строки **-L**, что довольно удобно, если отлаживается одна из копий программы, в то время как остальные копии продолжают нормально работать.

🕮 Подробнее о системе Syslog рассказывалось в главе 11.

Опция `confLOG_LEVEL`, заданная в командной строке или в файле конфигурации, определяет уровень важности, который программа **sendmail** примет в качестве порогового. Высокие пороговые значения соответствуют низким уровням важности и обеспечивают регистрацию большего объема информации.

В табл. 20.17 показано приблизительное соответствие между уровнями журнальной регистрации в программе **sendmail** и уровнями важности Syslog.

**Таблица 20.17. Соотношение между пороговыми уровнями программы sendmail и уровнями важности Syslog**

| Порог | Уровень важности |
|---|---|
| 0 | Регистрация отключена |
| 1 | `alert` или `crit` |
| 2 | `crit` |
| 3 | `err` или `warning` |
| 4 | `notice` |
| 5–11 | `info` |
| >=12 | `debug` |

Напомним, что сообщение, зарегистрированное в системном журнале на конкретном уровне, относится как к этому уровню, так и ко всем находящимся выше уровням. Файл **/etc/syslog.conf** определяет окончательный пункт назначения каждого сообщения. Местоположения, заданные по умолчанию, показаны в табл. 20.18.[16]

Несколько программ могут обрабатывать регистрационные файлы **sendmail**, создавая конечные продукты, которые варьируются от простых количественных и текстовых таблиц (**mreport**) до сложных веб-страниц (Yasma). У системного администратора может возникнуть необходимость или желание ограничить доступ к этим данным или, по крайней мере, проинформировать пользователей о сборе данных о них. Например, про-

[16] Было бы отлично, если бы кто-нибудь предпринял попытку стандартизации этих беспорядочных и, на первый взгляд, бессмысленных различий в сценариях, чтобы обеспечить их переносимость.

грамма Yasma (Yet Another Sendmail Log Analizer) позволяет скрывать имя пользователя, которому адресована почта, попавшая в отчет.

**Таблица 20.18. Местоположения регистрационных журналов программы sendmail**

| Система | Местоположение регистрационного файла |
|---|---|
| Ubuntu | **/var/log/mail.log** |
| SUSE | **/var/log/mail** |
| Red Hat | **/var/log/maillog** |
| Solaris | **/var/log/syslog и /var/adm/messages** |
| HP-UX | **/var/adm/syslog/mail.log** |
| AIX | **/var/log/mail** |

# 20.14. Почтовый агент Exim

Агент транспорта и доставки почты Exim был написан в 1995 году Филипом Хейзелом из Кембриджского университета и распространен под универсальной общедоступной лицензией GNU. Текущий выпуск, версия 4.71 Exim, появился в конце 2009 года.

В интерактивном доступе находятся тонны документации о программе, кроме того, было опубликовано несколько книг, написанных автором программного обеспечения.

Запросы в поисковой системе Google о почтовом сервере Exim часто приводят к устаревшим, а иногда и вообще неприемлемым ответам, поэтому в первую очередь следует обращаться к официальной документации. В дистрибутивный пакет входит документ о спецификации и конфигурации (**doc/spec.txt**), состоящий из более чем 400 страниц. Этот документ также доступен на сайте `exim.org` в виде файла PDF. Он представляет собой исчерпывающий справочник по программе Exim, который неукоснительно обновляется с каждым новым выпуском.

Существует два подхода к конфигурации постового сервера Exim: Debian и альтернативный. Операционная система Debian использует препроцессор **m4** и управляет своим собственным набором списков рассылки для поддержки пользователей. Мы не будем рассматривать расширения конфигурации, характерные для системы Debian.

Начиная с выпуска 4.70 почтовый сервер Exim прекратил поддержку технологии DomainKeys (предшественницы DKIM) и по умолчанию в настоящее время предусматривает поддержку технологии DKIM. В реальных условиях обе системы могут и должны сосуществовать, но система DKIM является стандартом IETF и в конце концов заменит систему DomainKeys.

Почтовый сервер Exim похожа на программу **sendmail** тем, что он реализован как единственный процесс, выполняющий все почтовые операции. Однако его авторы отказались от исторического багажа программы **sendmail** (поддержки устаревших форматов адресов, необходимости получать почту для узлов, не находящихся в Интернете, и т.д.). После компиляции в сочетании с инструментом для сканирования содержимого, постовый сервер Exim взаимодействует с популярными сканерами спама и вирусами, такими как SpamAssassin и ClamAV. Контроль правил реализован с помощью списков доступа, которые могут принимать или отклонять сообщения или передавать их внешнему программному обеспечению для сканирования. Доступ к фильтрам пользователей обеспечивается посредством записей в пользовательских файлах **.forward**. Многие аспекты функционирования почтового сервера Exim определяются во время компиляции, при этом ориентирами являются база данных почтового сервера Exim и форматы хранения сообщений.

Основные рабочие инструменты в почтовом сервере Exim называются маршрутизаторами и транспортными средствами. Оба они относятся к общей категории "драйверов". Маршрутизаторы решают, как сообщения должны быть переданы, а транспортные средства выбирают способ доставки. Маршрутизаторы — это упорядоченный список адресов, которые следует проверить, а транспортные средства — неупорядоченный набор способов доставки.

## Инсталляция почтового сервера EXIM

Последнюю версию дистрибутивного пакета можно загрузить с сайта `exim.org` или, если ваш сайт работает под управлением системы Linux, из вашего любимого репозитария пакетов. Место, где следует инсталлировать программу, идентификаторы пользователей и другие параметры указаны в файлах **README** и **src/EDITME**. Файл **EDITME** содержит более 1000 строк, которые по большей части представляют собой комментарии к процессу компиляции; требуемые изменения помечены. После редактирования сохраните файл как **../Local/Makefile** или **../Local/Makefile**-имя_ОС (если в одном и том же каталоге вы создаете конфигурации для разных операционных систем), а затем выполните команду **make**.

Ниже перечислены несколько важных переменных (по нашему мнению) и предлагаемые значения (по мнению разработчиков почтового сервера Exim) из файла **EDITME**. Первые пять строк являются обязательными, остальные только рекомендуются.

```
BIN_DIRECTORY=/usr/exim/bin              # Место для исполняемого модуля exim
SPOOL_DIRECTORY=/var/spool/exim          # Каталог для почтовой очереди
CONFIGURE_FILE=/usr/exim/configure       # Файл конфигурации программы Exim
SYSTEM_ALIASES_FILE=/etc/aliases         # Место для файлов псевдонимов
EXIM_USER=ref:exim                       # Пользователь программы EXIM

ROUTER_ACCEPT=yes                        # Драйверы маршрутизатора
ROUTER_DNSLOOKUP=yes
ROUTER_IPLITERAL=yes
ROUTER_MANUALROUTE=yes
ROUTER_QUERYPROGRAM=yes
ROUTER_REDIRECT=yes
TRANSPORT_APPENDFILE=yes                 # Драйверы транспорта
TRANSPORT_AUTOREPLY=yes
TRANSPORT_PIPE=yes
TRANSPORT_SMTP=yes
SUPPORT_MAILDIR=yes                      # Разрешенные форматы почтовых ящиков
SUPPORT_MAILSTORE=yes
SUPPORT_MBX=yes
LOOKUP_DBM=yes                           # Системы управления базами данных
LOOKUP_LSEARCH=yes                       # Метод последовательного поиска
LOOKUP_DNSDB=yes                         # Разрешает почти все поиски в DNS
LOOKUP_CDB=yes                           # Константа Дана Бернштейна для поиска
USE_DB=yes                               # Использовать Berkeley DB (из README)
DBMLIB=-ldb                              # (из файла README)
WITH_CONTENT_SCAN=yes                    # Включает сканирование содержимого
                                         # через списки ACL
EXPERIMENTAL_SPF=yes                     # Включает поддержку SPF, нужна libspf2
CFLAGS += -I/usr/local/include           # Из www.libspf2.org
LDFLAGS += -lspf2
LOG_FILE_PATH=/var/log/exim_%slog        # Файлы журнала: file, syslog или оба
```

```
LOG_FILE_PATH=syslog
LOG_FILE_PATH=syslog:/var/log/exim_%slog
EXICYCLOG_MAX=10                        # Количество сохраненных файлов журнала
                                        # равно 10
```

Если вы собираетесь использовать маршрутизаторы и транспортные средства, то их следует скомпилировать в код.

В настоящее время, которое характеризуется большими объемами доступной памяти, их можно использовать все вместе. Некоторые пути, заданные по умолчанию, являются нестандартными: например, бинарный файл в каталоге **/usr/exim/bin** и файл PID в каталоге **/var/spool/exim**. Эти параметры можно изменить в соответствии со своим инсталлированным программным обеспечением.

Существует около десяти доступных систем управления базами данных, включая MySQL, Oracle, CDB[17] и LDAP. Если вы хотите выбрать систему LDAP, то должны указать переменную `LDAP_LIB_TYPE`, которая сообщит программе Exim о том, что вы используете библиотеку LDAP (вариантами являются системы Netscape, Solaris и разные версии системы OpenLDAP). Кроме того, вы должны указать путь для включения файлов и библиотек системы LDAP.

В файле **EDITME** хранится информация о любых зависимостях, которые могут понадобиться при выборе базы данных. В комментариях о зависимостях, которые сопровождают каждую запись, есть фраза "(from README)", в которой указан не файл **src/EDITME**, а именно **README**.

Файл **EDITME** имеет много дополнительных параметров безопасности, которые позволяют активизировать дополнительные возможности, такие как SMTP AUTH, TLS, SASL, PAM, а также параметры управления владением файлом и разрешениями доступа к нему. Некоторые возможности программы Exim можно отключить на этапе компиляции, чтобы ограничить последствия ее взлома злоумышленниками.

До завершения инсталляции мы рекомендуем прочитать файл **EDITME** полностью. Это даст вам приятное ощущение, что вы управляете выполнением программы посредством конфигурационного файла. Высокоуровневый файл **README** содержит много информации о деталях, специфичных для операционных систем, которые иногда полезно включать в файл **EDITME**.

Модифицировав файл **EDITME** и инсталлировав его в каталог **Local/Makefile**, выполните команду **make** на вершине дистрибутивной иерархии, а затем инструкцию **sudo make install**. На следующем этапе необходимо протестировать исполняемый модуль **exim** и убедиться, что он доставляет почту так, как ожидается. Полезная тестовая документация содержится в файле **doc/spec.txt**.

Убедившись, что почтовый сервер Exim работает правильно, свяжите модуль **/usr/sbin/sendmail** с модулем **exim**, чтобы почтовый сервер Exim мог эмулировать традиционный интерфейс командной строки для обмена информацией с почтовой системой, используемой многими почтовыми агентами. Кроме того, следует сделать так, чтобы модуль **exim** выполнялся во время загрузки.

## Загрузка почтового сервера Exim

На компьютере, выполняющем роль концентратора почты, модуль **exim** обычно запускается во время загрузки в режиме демона и работает непрерывно, прослушивая порт 25 и принимая сообщения в рамках протокола SMTP.

[17] CDB — это неизменяемая база данных Дана Бернштейна; она хорошо масштабируется.

Детали загрузки операционной системы описаны в разделе 3.6. Как и программа **sendmail**, почтовый сервер Exim может иметь несколько режимов работы, и при запуске с разными флагами она выполняет разные функции. Флаги режима почтового сервера Exim похожи на флаги режима программы **sendmail**, потому что разработчики модуля **exim** очень стараются обеспечить совместимость при вызове почтовых агентов и других инструментов. Некоторые из наиболее распространенных флагов перечислены в табл. 20.19.

**Таблица 20.19. Распространенные флаги командной строки модуля exim**

| Флаг | Смысл |
|---|---|
| `-bd` | Запускается в режиме демона и прослушивает соединения на порту 25 |
| `-bf` или `-bF` | Запускается в пользовательском режиме или в режиме тестирования системного фильтра |
| `-bi` | Перестраивает хешированные псевдонимы (эквивалентен команде **newaliases**) |
| `-bp` | Выводит на печать почтовую очередь (эквивалентен команде **mailq**) |
| `-bt` | Включает режим тестирования адресов |
| `-bV` | Проверяет синтаксические ошибки в конфигурационном файле |
| `-d+`***категория*** | Запускается в режиме отладки; очень гибкая конфигурация, учитывающая категории |
| `-q` | Запускает обработчик очереди (эквивалентен команде **runq**) |

Любые ошибки, существующие в конфигурационном файле, можно обнаружить на этапе синтаксического анализа с помощью команды **exim -bV**, но некоторые ошибки можно выявить только на этапе выполнения. Распространенная ошибка — неправильно расположенные скобки.

Все нюансы, связанные с флагами и параметрами команды **exim**, включая обширную информацию об отладке, можно найти на соответствующей справочной странице.

## Утилиты почтового сервера Exim

Дистрибутивный пакет программы Exim включает в себя набор утилит, облегчающих слежение, отладку и проверку инсталляции. Ниже приведен список с кратким описанием каждой из этих утилит. Более подробная информация о них изложена в документации.

- **exiwhat** — перечисляет выполняемые процессы программы Exim
- **exiqgrep** — выполняет поиск в очереди
- **exiqsumm** — создает отчет об очереди
- **exigrep** — просматривает основной журнал
- **exipick** — ищет сообщения по заданным критериям
- **exicyclog** — чередует регистрационные файлы
- **eximstats** — извлекает статистические показатели из журнала
- **exim_checkaccess** — проверяет доступность заданного IP-адреса
- **exim_dbmbuild** — создает файл DBM
- **exinext** — извлекает информацию о повторных попытках
- **exim_dumpdb** — распечатывает базу подсказок
- **exim_tidydb** — очищает базу подсказок
- **exim_fixdb** — исправляет базу подсказок

- **exim_lock** — блокирует файл почтового ящика
- **exilog** — визуализирует регистрационные файлы на нескольких серверах

Еще одна утилита, которая входит в пакет Exim, называется **eximon**. Она представляет собой приложение для системы X Windows, которое выводит на экран состояние программы Exim, состояние ее очереди и остаток регистрационного файла. Как и основной дистрибутивный пакет, она создается с помощью редактирования хорошо комментированного файла **EDITME** в каталоге **exim_monitor** и выполнения команды **make**. Параметры, заданные по умолчанию для утилиты **eximon**, подобраны очень удачно, поэтому конфигурирование этого приложения обычно не вызывает затруднений. Кроме того, конфигурирование и управление очередью можно осуществлять с помощью графического интерфейса утилиты **eximon**.

## Язык конфигурации программы Exim

Рассматриваемый здесь язык конфигурации программы Exim (точнее, языки: один для фильтров, другой для регулярных выражений и так далее) напоминает довольно старый (разработанный в 1970-х годах) язык Forth.[18] При первом чтении конфигурации программы иногда трудно отличить ключевые слова и имена параметров, которые являются фиксированными в языке Exim, от имен переменных, определенных системными администраторами. Для того чтобы решить эту проблему, мы поставили перед именем каждой переменной префикс `my_`.

Несмотря на то что программа Exim рекламируется как легкая для конфигурирования и хорошо документированная, ее довольно сложно освоить. Раздел спецификации "Как программа Exim получает и доставляет почту" требует от новичков больших усилий. Тем не менее в ней изложены все основные концепции, лежащие в основе системы.

После присвоения стандартной переменной конкретного значения язык Exim иногда активизирует определенные действия. В нем есть около 120 стандартных переменных, значения которых можно изменять в результате одного из действий. Эти переменные можно включать в условные выражения.

Выражение для вычисления инструкции `if` и подобных ей может напомнить обратную польскую запись, характерную для времен расцвета калькуляторов Hewlett-Packard. Рассмотрим простой пример. В строке

```
acl_smtp_rcpt = ${if ={25}{$interface_port} \
{acl_check_rcpt} {acl_check_rcpt_submit} }
```

установка параметра `acl_smtp_rcpt` вызывает реализацию списка управления доступом для каждого пользователя (т.е. выполнение команды SMPT RCPT). Значение, присвоенное этому параметру, может быть равным `acl_check_rcpt` или `acl_check_rcpt_submit`, в зависимости от того, имеет ли переменная `$interface_port` значение 25.

Мы не будем углубляться в детали языка конфигурации программы Exim, лишь посоветуем читателям обратиться к обширной документации. В частности, обратите внимание на раздел продолжения строки в спецификации программы Exim.

## Файл конфигурации программы Exim

Поведением программы Exim во время выполнения управляет файл конфигурации, который обычно называется **/usr/exim/configure**. Его имя представляет собой одну

[18] Для экспертов в компьютерных науках заметим, что этот язык является полным по Тьюрингу; в переводе для простых смертных это значит "мощный и сложный".

из обязательных переменных, задаваемых в файле **EDITME** и компилируемых в бинарный код.

Файл конфигурации **src/configure.default**, поставляемый по умолчанию, сопровождается подробными комментариями и хорошо подходит для новичков. Мы рекомендуем не слишком далеко отклоняться от этого файла, пока вы не освоитесь с парадигмой Exim и не научитесь создавать более сложные конфигурации для специальных ситуаций. Разработчики программы Exim хорошо продумали поддержку наиболее распространенных ситуаций и создали вполне разумные конфигурации, предусмотренные по умолчанию.

Рекомендуем также не изменять имена переменных, использованных в файле конфигурации, заданном по умолчанию, поскольку именно их ожидают увидеть в списках рассылки люди, которые будут отвечать на ваши вопросы о конфигурации.

Программа **exim** выводит сообщения в поток и прекращает работу, если в вашем файле конфигурации есть синтаксическая ошибка. Однако она не выявляет все синтаксические ошибки немедленно, поскольку не обращается к переменным, пока в них нет потребности.

Порядок записей в файле конфигурации не совсем произвольный: раздел глобальных параметров конфигурации должен существовать и быть первым. Все остальные разделы являются необязательными и могут следовать в любом порядке.

Перечислим некоторые разделы.

- Глобальные параметры конфигурации (обязательны)
- `acl` — списки управления доступом, фильтрующие адреса и сообщения
- `authenticators` — раздел для параметров команды SMPT AUTH или протокола аутентификации TLS
- `routers` — упорядоченная последовательность, предназначенная для определения места назначения сообщения
- `transports` — определения драйверов, предназначенных для фактической доставки
- `retry` — настройки правил для решения проблем, связанных с сообщениями
- `rewrite` — правила перезаписи глобальных адресов
- `local_scan` — ловушка для спама

Каждый раздел, за исключением первого, начинается с инструкции `begin` *имя-раздела*, например `begin acl`. Инструкции `end` *имя-раздела* не существует; признаком конца раздела является инструкция `begin` следующего раздела. Отступы между разделами облегчают чтение, но для программы Exim значения не имеют.

Некоторые инструкции конфигурации присваивают имена объектам, которые впоследствии будут использованы для управления потоками сообщений. Эти имена должны начинаться с буквы и содержать только буквы, цифры и символ подчеркивания. Если первым символом, отличающимся от разделителей строки, является символ #, то вся остальная часть строки считается комментарием. Это значит, что вы не можете поместить комментарий в строку, в которой помещается инструкция; он не будет распознан как комментарий, поскольку его первый символ не является символом #.

Программа Exim позволяет включать файлы в любое место файла конфигурации. Существует две формы включения файлов.

```
.include absolute-path
.include_if_exists absolute-path
```

Первая форма генерирует ошибку, если файла нет. Несмотря на то что включение файлов позволяет сохранять небольшой объем конфигурационного файла, за время существования сообщения они считываются несколько раз, поэтому может быть лучше просто включить их содержимое прямо в файл конфигурации.

## Глобальные параметры

В разделе глобальных параметров задается множество данных, включая операционные параметры (пределы, размеры, тайм-ауты, свойства почтового сервера на указанном узле), определения списков (локальных узлов, локальных узлов для перенаправления, удаленных доменов для перенаправления) и макросы (имя узла, контакт, местоположение, сообщения об ошибках, банер SMTP).

### *Параметры*

Параметры устанавливаются с помощью следующей синтаксической конструкции.

```
имя_параметра = значение[я]
```

Здесь *значения* могут быть булевыми или строчными, целыми или десятичными числами, а также временными интервалами. Допускается также несколько значений, в этом случае они разделяются двоеточиями.

Использование двоеточия в качестве разделителя создает проблему, потому что в адресах IPv6 оно является частью адреса. Решить эту проблему можно путем удваивания количества двоеточий, но намного проще и удобнее для чтения переопределить символ разделения и задать его равным < во время присвоения параметру его значения. Например, в следующих строках задаются значения параметра `localhost_interfaces`, содержащие IPv4- и IPv6-адреса локального узла.

```
local_interfaces = 127.0.0.1 : ::::1
local_interfaces = <; 127.0.0.1 ; ::1
```

Вторая форма, в которой в качестве разделителя используется точка с запятой, является более удобной для чтения и менее уязвимой для ошибок.

Существует огромное количество параметров — в предметном указателе документации их более 500. А мы говорили, что программа **`sendmail`** была слишком сложной! Большинство параметров имеет вполне разумные значения по умолчанию, и все параметры имеют информативные имена. Для того чтобы легко находить новый параметр, целесообразно скопировать файл **`doc/spec.txt`** из дистрибутивного пакета в свой привычный текстовый редактор. Мы не собираемся описывать все параметры, а упомянем лишь те из них, которые встречаются в наших примерах.

### *Списки*

Программа Exim использует четыре вида списков, которые объявляются с помощью ключевых слов `hostlist`, `domainlist`, `addresslist` и `localpartslist`. Ниже приведены два примера использования ключевого слова `hostlist`.

```
hostlist my_relay_list = 192.168.1.0/24 : myfriend.example.com
hostlist my_relay_list = /usr/local/exim/relay_hosts.txt
```

Члены списка могут быть перечислены непосредственно в строке или загружены из файла. Если они указываются непосредственно в строке, то разделяются двоеточиями. Существует до 16 именованных списков каждого типа. В примерах, приведенных выше,

мы включили все компьютеры в локальной сети /24 и указали конкретное имя локального компьютера.

Символ @ может быть членом списка; он означает имя локального узла и помогает написать единственный обобщенный конфигурационный файл, который будет применяться на всех компьютерах вашей сети. Не менее полезным является обозначение @[], которое означает все IP-адреса, прослушиваемые программой Exim; иначе говоря, все IP-адреса локального узла.

Для того чтобы сослаться на список, следует просто поставить символ + перед его именем, если хотите сравнивать его члены, или !+, если хотите сравнивать элементы, не являющиеся членами списка; например, +my_relay_list. Между знаком + и именем списка пробела быть не должно.

Списки могут содержать ссылки на другие списки, а знак ! означает отрицание. Списки, содержащие ссылки на переменные (например, $variable_name), замедляют обработку, потому что по умолчанию программа Exim не может кешировать результаты сравнений по списку.

### Макрос

Макросы можно использовать для определения параметров, сообщений об ошибках и т.д. Грамматический разбор макросов очень примитивен, поэтому вы не можете определить макрос, имя которого является подмножеством другого макроса, не получив непредсказуемых результатов.

Синтаксис макроса выглядит следующим образом.

```
MACRO_NAME = остальная часть строки
```

Например, в первой из приведенных ниже строк определяется макрос с именем ALIAS_QUERY, который выполняет поиск записи о псевдониме пользователя в базе данных MySQL. Вторая строка демонстрирует использование макроса для выполнения реального поиска, результат которого хранится в переменной с именем data.

```
ALIAS_QUERY = \
   select mailbox from user where login = '${quote_mysql:$local_part}';
data = ${lookup mysql{ALIAS_QUERY}}
```

Имена макросов не обязательно набирать только прописными буквами, но они должны начинаться с прописной буквы. Однако соглашение, по которому макросы состоят только из прописных букв, повышают ясность текста. Файл конфигурации может содержать директиву ifdef, вычисляющую макрос. Система использует ее, чтобы определить, включать или нет фрагмент конфигурационного файла. Поддерживаются все возможные формы инструкции ifdef; все они начинаются с точки.

### ACL (списки управления доступом)

Списки управления доступом фильтруют адреса входящих сообщений и либо принимают их, либо отклоняют. Программа Exim разделяет адреса входящих сообщений на локальную часть, соответствующую пользователю, и доменную, соответствующую домену получателя.

Списки управления доступом можно применять на разных стадиях обмена информацией по протоколу SMTP: HELO, MAIL, RCPT, DATA и т.д. Обычно список управления доступом требует строгого следования протоколу SMTP на стадии HELO, проверяет отправителя и его домен на стадии MAIL, проверяет получателя на стадии RCPT и сканирует содержимое сообщения на стадии DATA.

Для того чтобы определить, какой список управления доступом должен применяться после каждой команды в протоколе SMTP, используется множество параметров с именами, имеющими формат `acl_smtp_`*`команда`*. Например, параметр `acl_smtp_rcpt` означает, что список управления доступом должен применяться к каждому адресу, указанному как получатель данного сообщения. Списки управления доступом можно задать в разделе `acl` в файле конфигурации, в файле, на который ссылается параметр `acl_smtp_`*`команда`*, или в командной строке.

Ниже приведен пример списка управления доступом `my_acl_check_rcpt`. Его можно вызвать, присвоив это имя параметру `acl_smtp_rcpt` в разделе глобальных параметров в файле конфигурации. Если этот список управления доступом отвергает адрес, указанный в команде RCPT, то сервер отправителя должен отступить и больше не пытаться отправлять сообщения на этот адрес. Другой параметр списка управления доступом — `acl_smtp_data` — применяется к сообщению после его получения, например для сканирования его содержимого.

```
begin acl
    my_acl_check_rcpt:
        accept   hosts = :
                 control = dkim_disable_verify

        deny     message = Restricted characters in address
                 domains = +local_domains
                 local_parts = ^[.] : ^.*[@%!/|]

        deny     message = Restricted characters in address
                 domains = !+local_domains
                 local_parts = ^[./|] : ^.*[@%!] : ^.*/\\.\\./

        accept   local_parts = postmaster
                 domains = +local_domains

        require  verify = sender

        accept   hosts = +relay_from_hosts
                 control = submission
                 control = dkim_disable_verify

        accept   authenticated = *
                 control = submission
                 control = dkim_disable_verify

        require  message = Relay not permitted
                 domains = +local_domains : +relay_to_domains

        require verify = recipient
        accept
```

Этот список управления доступом, заимствованный из документации к программе Exim, по умолчанию заканчивается командой `accept`; вы можете изменить свое решение и сделать так, чтобы список управления доступом по умолчанию отклонял сообщение, как это обычно делают брандмауэры. Этому списку управления доступом по умолчанию назначено имя `acl_check_rcpt`; вероятно, это имя изменять не следует (тем не

менее мы изменили его, чтобы продемонстрировать, что оно задается вами, а не фиксируется заранее параметрами конфигурации программы Exim).

Первая строка, содержащая раздел `accept` и одно двоеточие, представляет собой пустой список. Пустой список удаленных узлов применяется в ситуациях, когда локальный пользовательский почтовый агент (MUA) подает сообщение на стандартный вход транспортного агента. Если проверяемый адрес соответствует данному условию, список управления доступом принимает адрес и отключает проверку подписи DKIM, которая по умолчанию включена. Если же адрес не соответствует данному разделу `address`, то управление переходит к следующему разделу в определении списка.

Первая строка конфигурации `deny` предназначена для сообщений, поступающих на ваши локальные домены. Она отклоняет любой адрес, в котором локальная часть (имя пользователя) начинается с точки или содержит специальные символы @, %, !, / или |. Вторая строка `deny` применяется к сообщениям, посланным во внешний мир вашими пользователями. Она также не допускает использование некоторых специальных символов и их комбинаций в качестве части адреса, поскольку это свидетельствует о том, что ваши компьютеры были заражены вирусом или другими вредоносными программами. В прошлом такие адреса использовались спамерами для запутывания списков управления доступом или для создания проблем в системе безопасности.

В целом, если вы хотите использовать выражения `$local_parts` (например, в качестве пользовательского имени получателя) в имени каталога (для хранения почты или просмотра файла автоответчика, например), будьте очень осторожны, чтобы ваши списки управления доступом, отфильтровывающие любые специальные символы, не привели к непредвиденным последствиям. (В этом примере выполняется поиск последовательности символов `/../`, которые могут создать проблемы, если имя пользователя является частью имени пути.)

Следующий раздел `accept` гарантирует, что почта, посланная администратору электронной почты, всегда будет доставляться, если она была послана в локальный домен; это облегчает отладку.

Строка `require` проверяет, можно ли вернуть сообщение о недоставке, но проверка относится только к домену отправителя.[19] Если пользовательское имя отправителя было подделано, то сообщение о недоставке может породить новое сообщение о недоставке. В этом месте проверку можно усилить, вызвав другую программу, но некоторые сайты рассматривают такие вызовы как злонамеренные действия и могут внести ваш почтовый сервер в черный список или в список серверов с плохой репутацией.

Следующий раздел `accept` проверяет узлы, через которые разрешается выполнять ретрансляцию, т.е. локальные узлы, подающие почту в систему. Эта строка означает, что программа **`exim`** должна действовать как агент представления почтовых сообщений и исправлять любые недостатки в заголовках при поступлении сообщений от пользовательского агента. Адрес отправителя не проверяется, потому что многие пользовательские агенты сбиваются с толку ошибочными возвратами. (Это приемлемо только для локальных машин, которые выполняют ретрансляцию на интеллектуальный узел, а не во внешние домены.) Верификация подписи DKIM отключена, поскольку эти сообщения отправляются вашими пользователями или их коллегами по ретрансляции.

Последний раздел `accept` относится к локальным узлам, выполняющим аутентификацию с помощью протокола SMTP AUTH. Эти сообщения снова обрабатываются как представления от пользовательских агентов.

[19] Раздел `require` означает "`deny`, если не совпало".

Затем мы проверяем домен адресата, который указан в заголовке письма и должен содержаться либо в нашем списке `local_domains`, либо в списке `relay_to_domains`, в котором перечислены домены, имеющие право на ретрансляцию. (Списки этих доменов определяются в другом месте.) Письмо с адресом, который не входит ни в один из этих списков, порождает особое сообщение об ошибке. Проверка подписи DKIM снова отключена.

Итак, если все наши сформулированные выше предположения были удовлетворены и ни одно из более тонких правил `accept` или `deny` не было нарушено, мы верифицируем получателя и принимаем сообщение. В эту категорию попадает большинство сообщений, поступающих из Интернета.

В приведенный выше пример мы не включили сканирование черных списков. Для доступа к черным спискам следует использовать один из примеров, описанных в файле конфигурации, предоставленном по умолчанию, или что-нибудь наподобие следующего кода.

```
deny    condition = ${if isip4{$sender_host_address}}
        !authenticated = *
        !hosts = +my_whitelist_ips
        !dnslists = list.dnswl.org
        domains = +local_domains
        verify = recipient
        message = You are on RBL $dnslist_domain: $dnslist_text
        dnslists = zen.spamhaus.org
        logwrite = Blacklisted sender [$sender_host_address] \
               $dnslist_domain: $dnslist_text
```

В переводе на обычный язык этот фрагмент означает, что если сообщение соответствует *всем* следующим критериям, то оно отклоняется с сообщением о пользовательской ошибке и регистрируется в журнале (также вместе с сообщением о пользовательской ошибке).

- Сообщение поступило с IPv4-адреса (некоторые списки неправильно обрабатывают IPv6-адреса).
- Сообщение не ассоциировано с аутентифицированным сеансом SMTP.
- Сообщение поступило от отправителя, которого нет в локальном белом списке.
- Сообщение поступило от отправителя, которого нет в глобальном (из Интернета) белом списке.
- Сообщение адресовано корректному локальному получателю.
- Узел отправителя занесен в черный список сайта `zen.spamhaus.org`.

Переменные `dnslist_text` и `dnslist_domain` задаются путем присваивания параметру `dnslists`, который выполняет переключение поиска по черным спискам. Этот раздел `deny` должен размещаться сразу после проверки необычных символов в адресах.

Рассмотрим еще один пример списка управления доступом, который отклоняет сообщение, поступившее от удаленного сайта, неправильно ответившего на команду HELO.

```
acl_check_mail:
    deny message = 503 Bad sequence of cmds - must send HELO/EHLO first
         condition = ${if !def:sender_helo_name}
    accept
```

Программа Exim решает проблему "болтуна" (более специальный случай сайта, "неправильно отвечающего на команду HELO") с помощью параметра `smtp_enforce_sync`, который включается по умолчанию.

## Сканирование содержимого на этапе применения списков управления доступом

Программа Exim поддерживает мощное сканирование содержимого в нескольких точках на пути сообщения внутри почтовой системы: на этапе применения списков управления доступом (после команды SMTP DATA), во время доставки с помощью параметра `transport_filter` или функции `local_scan` после выполнения всех проверок списков управления доступом. Для поддержки сканирования необходимо выполнить компиляцию данного модуля в систему Exim с помощью переменной `WITH_CONTENT_SCAN` в файле **EDITME**; по умолчанию она закомментирована. Этот параметр наделяет списки управления доступом дополнительными мощью и гибкостью, добавляя два новых параметра конфигурации: `spamd_address` и `av_scanner`.

Сканирование на этапе проверки списков управления доступом позволяет отклонять сообщения на лету с помощью обмена информацией между транспортным агентом и узлом отправителя. Сообщение никогда не будет принято к доставке, поэтому сообщение о недоставке не генерируется. Это очень хороший способ отклонения сообщений, поскольку он позволяет избежать обратной рассылки спама, вызванного сообщениями о недоставке, по поддельным адресам отправителей.

### *Сканирование вирусов*

Для сканирования вирусов сначала следует задать тип сканера и его параметры в переменной `av_scanner` в разделе глобальных параметров в файле конфигурации. В табл. 20.20 перечислены сканеры, допустимые в текущей версии программы Exim, и их соответствующие спецификации `av_scanner`.

**Таблица 20.20. Антивирусные сканеры, известные программе exim**

| Спецификация сканера | Демон или служба |
|---|---|
| `aveserver` *путь-к_сокету* | Демон сканера Касперского 5 |
| `clamd:` *ip-порта или путь-к-сокету* | ClamAV по протоколу TCP или локальный сокет |
| `cmdline:` путь `found-regex name-regex` | Обобщенный интерфейс командной строки |
| `drweb:` *ip-адрес порта или путь-к-сокету* | Демон сканера DrWeb (`said.com`) |
| `fsecure:` путь-к-`.fsav`-файлу | Демон сканера F-Secure (`f-secure.com`) |
| `kavdaemon:` *путь-к-сокету* | Демон сканера Касперского, версия 4 |
| `sophie:` *путь-к-сокету* | Интерфейс Sophie для сканера Sophos [a] |

[a] См. `clanfield.info/sophie`. Этот сканер задан по умолчанию (путь **/var/run/sophie**).

Задав переменную `av_scanner`, можно использовать условие `malware` в списке управления доступом, которое проверяет содержимое сообщений после выполнения команды DATA в ходе обмена информацией по протоколу SMTP. Рассмотрим пример, взятый из документации к программе Exim.

```
deny message = This message contains malware ($malware_name)
     demime = *
     malware = *
```

Раздел `malware` вызывает сканер вирусов, если передаваемое значение равно `true` (что в данном примере выполняется всегда). Если включен параметр `demime` и сообщение содержит приложение, закодированное по стандарту MIME, то программа **exim** декодирует приложение, чтобы сделать его доступным для антивирусного сканера.

Однако большинство сканеров может выполнить это декодирование самостоятельно. Для того чтобы избежать дублирования при решении этой задачи, раздел `demime` следует включать, только если это необходимо для антивирусного сканера. Такое дублирование не вызывает ошибки, но расходует ресурсы и замедляет обработку почты.

### Сканирование спама

Для сканирования спама агент Exim использует программу SpamAssassin, которая обычно принимает сообщения на TCP-порт 783, но может также использовать сокет локального домена. Задавая параметры соединения в файле конфигурации программы **exim**, присвойте какое-нибудь значение переменной `spamd_address`. Этим значением может быть либо IP-адрес и номер порта, разделенные пробелом, либо абсолютный путь сокета локального домена. Можно также одновременно задавать до 32 пар "адрес/порт", чтобы использовать несколько экземпляров утилиты SpamAssassin.

Присвоив имя пользователя переменной `spam`, вызовите утилиту SpamAssassin из списка управления доступом, связанного с командой SMTP DATA. Если в качестве пользователя указано имя `nobody`, то программа SpamAssassin использует универсальный профиль сканирования, в противном случае — профиль, связанный с указанным вами пользователем (если такой профиль существует).[20] Строки

```
deny    message = This message was classified as spam
        spam = nobody
```

означают использование системного профиля спама, установленного по умолчанию. Просто поместить инструкцию `spam` в файл конфигурации недостаточно; необходимо присвоить определенное значение переменной `spam`.

Поскольку программа SpamAssassin работает медленно, а большинство сообщений спама являются короткими, необходимо проверять размер и сканировать только небольшие сообщения. Рассмотрим пример.

```
deny    message = This message was classified as spam
        condition = ${if < {$message_size}{10K}}
        spam = nobody
```

С помощью программы SpamAssassin можно решать более сложные задачи; все детали описаны в спецификации программы Exim.

## Аутентификаторы

Аутентификаторы (authenticators) — это драйверы, взаимодействующие с последовательностью "вызов/ответ" команд SMTP AUTH и идентифицирующие механизм аутентификации, приемлемый для клиента и сервера. Программа Exim поддерживает четыре механизма.

- `AUTH_CRAM_MD5` (RFC2195).
- `AUTH_CYRUS_SASL` для использования программного обеспечения Cyrus IMAP.
- `AUTH_PLAINTEXT`, включающий макросы PLAIN и LOGIN.
- `AUTH_SPA`, поддерживающий механизм аутентификации Secure Password Authentication компании Microsoft.

[20] На первый взгляд, требование задавать имя пользователя обеспечивает гибкость. Однако, поскольку сканирование осуществляется после выполнения команды DATA, а не RCPT, сообщение уже связано со своим получателем. Если же получателей несколько, то неизвестно, какой профиль спама следует использовать.

Когда агент **exim** получает электронную почту, он действует как сервер SMTP AUTH. Когда он посылает электронную почту, является клиентом. Параметры, появляющиеся в определениях экземпляров аутентфикатора, сопровождаются префиксами `server_` или `client_`, позволяющими задавать разные конфигурации, в зависимости от роли, которую играет агент Exim.

Аутентификаторы используются в списках управления доступом, как показано в одном из примеров, приведенных выше.

```
accept authenticated = *
```

Ниже приведен пример, демонстрирующий клиентский и серверный механизмы LOGIN. В этом простом примере используются фиксированные имя пользователя и пароль, что приемлемо для небольших организаций, но, вероятно, это не следует рекомендовать для крупных сетей.

```
begin authenticators

   my_client_fixed_login:
      driver = plaintext
      public_name = LOGIN
      client_send = : myusername : mypasswd

   my_server_fixed_login:
      driver = plaintext
      public_name = LOGIN
      server_advertise_condition = ${if def:tis_cipher}
      server_prompts = User Name : Password
      server_condition = ${if and {{eq{$auth1}{имя_пользователя}} \
          {eq{$auth2}{пароль}}}}
      server_set_id = $auth1
```

Данные для аутентификации могут поступать из многих источников: LDAP, PAM, **/etc/passwd** и т.д. Раздел `server_advertise_condition` в указанном фрагменте кода предотвращает открытую пересылку по линии связи паролей почтовых клиентов по требованию протокола TLS (через STARTTLS или SSL). Если вы хотите, чтобы программа **exim** действовала аналогично, будучи клиентский системой, используйте параметр `client_condition` в разделе `client`, тоже вместе с параметром `tis_cipher`.

Детальное описание всех возможных параметров аутентификации программы Exim и примеры их использования можно найти в документации.

## Маршрутизаторы

Маршрутизаторы обрабатывают адреса электронной почты, либо перезаписывая их, либо переприсваивая транспортному агенту и посылая в пункт назначения. Конкретный маршрутизатор может иметь несколько экземпляров с разными параметрами.

Вы указываете последовательность маршрутизаторов. Сообщение начинает свой путь с первого маршрутизатора и проходит по списку, пока сообщение не будет принято или отклонено. Принимающий маршрутизатор обычно передает сообщения драйверу транспорта. Маршрутизаторы обрабатывают как входящую, так и исходящую корреспонденцию. Их можно считать чем-то вроде подпрограмм в языке программирования.

Маршрутизатор может возвращать любые из следующих уведомлений о статусе сообщения.

- `accept` — маршрутизатор принимает адрес и передает его драйверу транспорта.

- `pass` — данный маршрутизатор не может обработать указанный адрес; управление передается следующему маршрутизатору.
- `decline` — маршрутизатор решает не обрабатывать адрес; следующий маршрутизатор, пожалуйста!
- `fail` — адрес является неправильным; маршрутизатор ставит его в очередь для генерации сообщения о недоставке.
- `defer` — оставляет сообщение в очереди для последующей обработки.
- `error` — обнаружена ошибка в спецификации маршрутизатора; сообщение откладывается.

Если сообщение получило от всех маршрутизаторов в очереди уведомления `pass` или `decline`, оно возвращается отправителю как письмо, имеющее немаршрутизируемый адрес.

Если сообщение удовлетворяет всем предусловиям для маршрутизатора и маршрутизатор закончил работу, выполнив инструкцию `no_more`, то это сообщение не передается никаким дополнительным маршрутизаторам, независимо от того, какой статус оно получило у текущего маршрутизатора. Например, если ваш удаленный SMTP-маршрутизатор выполнил предусловие `domains = !+local_domains` и установил параметр `no_more`, то следующему маршрутизатору в последовательности отправляются только сообщения, адресованные локальным пользователям (т.е. сообщения, нарушающие предусловие домена).

Маршрутизаторы имеют много параметров; наиболее распространенными являются параметры предусловий или условий отказов, возвращаемых сообщений об ошибках и используемых драйверов транспорта.

В следующих разделах подробно описываются маршрутизаторы `accept`, `dnslookup`, `manualroute` и `redirect`. Снипеты конфигурации основаны на предположении, что агент **`exim`** выполняется на локальном компьютере в домене `example.com`. Все эти снипеты довольно просты; если вы захотите использовать более изощренные маршрутизаторы, обратитесь к документации.

## Маршрутизатор accept

Этот маршрутизатор помечает адрес как приемлемый и передает соответствующее сообщение драйверу транспорта. Ниже приведен пример экземпляров маршрутизатора `accept` с именами `localusers` (для доставки локальной почты) и `save_to_file` (для отправления сообщения в архив).

```
localusers:
    driver = accept
    domains = example.com
    check_local_user
    transport = my_local_delivery
save_to_file:
    driver = accept
    domains = dialup.example.com
    transport = batchsmtp_appendfile
```

Экземпляр маршрутизатора с именем `localusers` проверяет, совпадает ли доменная часть адреса доставки с именем домена `example.com` и является ли локальная часть адреса регистрационным именем локального пользователя. Если оба условия выполнены, маршрутизатор передает сообщение экземпляру драйвера транспорта с именем `my_local_delivery`, определенному в разделе транспорта. Экземпляр `save_to_file`

предназначен для абонентов автоматической телефонной линии; он добавляет сообщение в файл, указанный в определении транспорта `batchsmtp_appendfile`.

## Маршрутизатор dnslookup

Как правило, маршрутизатор `dnslookup` используется для исходящих сообщений. Он ищет запись MX о домене отправителя и передает сообщение драйверу транспорта SMTP для доставки. Ниже приведен экземпляр маршрутизатора с именем `remoteusers`.

```
remoteusers:
   driver = dnslookup
   domains = !+example.com
   transport = my_remote_delivery
```

🕮 Более подробно пространство частных адресов RFC1918 описано в разделе 14.4.

Код `dnslookup` ищет записи MX об адресате. Если их нет, он проверяет запись A. В итоге, данный экземпляр маршрутизатора может запрещать доставку сообщений на определенные IP-адреса; например, частные адреса RFC1918 не могут маршрутизироваться в Интернет. Больше информации об этом можно найти в описании параметра `ignore_target_hosts option`.

### Маршрутизатор manualroute

Гибкий драйвер `manualroute` может посылать электронную почту куда угодно. Информация о маршрутизации может быть представлена в виде таблицы правил соответствия для домена получателя (`route_list`) или отдельного правила, которое применяется для всех доменов (`route_data`). Ниже приведены два экземпляра маршрутизатора `manualroute`. Первый экземпляр реализует концепцию "интеллектуального узла", в которой все исходящие сообщения посылаются центральному "интеллектуальному" узлу для обработки. Этот экземпляр называется `smarthost` и применяется ко всем доменам получателей, не указанным (используется символ !) в списке `local_domains`.

```
smarthost:
   driver = manualroute
   domains = !+local_domains
   transport = remote_snmp
   route_data = smarthost.example.com
```

Экземпляр маршрутизатора `firewall`, приведенный ниже, использует протокол SMTP для отправки входящих сообщений узлам, находящимся внутри брандмауэра (возможно, после сканирования их на спам и вирусы). Он ищет данные о маршрутизации для каждого домена получателя в базе данных CDB, содержащей имена локальных узлов. (Для того чтобы иметь возможность использовать базу данных CDB, сборку программы Exim следует проводить с параметром `LOOKUP_CDB`.)

```
firewall:
   driver = manualroute
   transport = remote-smtp
   route_data = ${lookup{$domain} cdb {/internal/host/routes}}
```

### Маршрутизатор redirect

Драйвер `redirect` перезаписывает адрес, используя псевдонимы из системного файла **`aliases`** или пользовательского файла **`~/.forward`**. Обычно он не присваивает пере-

записанный адрес транспорту, а оставляет эту работу другим маршрутизаторам, находящимся в цепочке.

Первый экземпляр system_aliases, приведенный ниже, ищет псевдонимы методом последовательного перебора (lsearch) в файле **/etc/aliases**. Это удобно, если файл **aliases** небольшой, но если он огромный, то метод последовательного перебора следует заменить поиском в базе данных. Второй экземпляр с именем forwardfile сначала проверяет, что почта адресована локальному пользователю, а затем просматривает пользовательский файл **.forward**.

```
system_aliases:
   driver = redirect
   data = ${lookup{$local_part} lsearch {/etc/aliases}}

user_forward:
   driver = redirect
   check_local_user
   file = $home/.forward
   no_verify
```

Параметр check_local_user гарантирует, что получатель является корректным локальным пользователем. Параметр no_verify означает, что проверка правильности адреса, на который перенаправляется сообщение, не нужна, следует просто выполнить поставку.

### *Фильтрация пользователей с помощью файлов .forward*

Программа Exim позволяет не только перенаправлять сообщения, используя файлы **.forward**, но и фильтровать их, основываясь на содержимом пользовательского файла **.forward**. Она поддерживает свою собственную фильтрацию, а также фильтрацию методом решета (Sieve filtering), являющуюся стандартом IETF. Если первая строка пользовательского файла **.forward** выглядит как

```
#Exim filter
```

или

```
#Sieve filter
```

то последующие команды фильтрации (их около 15) можно использовать для выработки решения, куда следует доставить сообщение. Фильтрация на самом деле не подразумевает доставки сообщения, она просто выясняет пункт назначения. Рассмотрим пример.

```
#Exim filter
if  $header_subject: contains SAGE or $header_subject: contains sysadmin
then
    save $home/mail/sage-sysadmin
endif
```

Для того чтобы управлять тем, что пользователи могут делать в своих файлах **.forward**, а чего они делать не могут, используется множество параметров **.forward**. Имена этих параметров начинаются с префиксов forbid_ или allow_. Важно предотвратить запуск пользователями оболочек, загрузку библиотек в бинарные коды или использование встроенного интерпретатора языка Perl, если они не должны этого делать. Производя обновление почтовой системы, проверьте новые параметры forbid_*, чтобы убедиться, что пользователи не имеют возможности экспериментировать со своими файлами **.forward**.

## Транспортные механизмы

Маршрутизаторы решают, куда должны следовать сообщения, а транспортные механизмы выполняют их фактическое перемещение в пункт назначения. Локальные транспорты обычно добавляются в файл, передаются локальной программе или обмениваются информацией по протоколу LMTP с серверами IMAP. Удаленные транспорты обмениваются информацией со своими партнерами в Интернете по протоколу SMTP.

В агенте Exim существует пять транспортных механизма: `appendfile`, `lmtp`, `smtp`, `autoreply` и `pipe`; мы опишем транспортные механизмы `appendfile` и `smtp`. Транспорт `autoreply` обычно используется для отправки сообщений автоответчика, а транспорт `pipe` передает сообщения на вход команд через канал UNIX. Как и при работе с маршрутизаторами, необходимо определить экземпляры транспортов, причем желательно иметь несколько экземпляров транспорта одного и того же типа. Порядок важен для маршрутизаторов, но не для транспортных механизмов.

### *Транспортный механизм appendfile*

Драйвер `appendfile` хранит сообщения в формате **`mbox`**, **`mbx`**, **`Maildir`** или **`mailstore`** в заданном файле или каталоге. Компилируя программу Exim, необходимо включать соответствующие форматы почтового ящика; по умолчанию они закомментированы в файле **`EDITME`**. В следующем примере определен транспорт `my_local_delivery` (экземпляр транспорта `appendfile`), который упоминался в определении экземпляра маршрутизатора `localusers` в одном из предыдущих примеров.

```
my_local_delivery:
    driver = appendfile
    file = /var/mail/$local_part
    delivery_date_add
    envelope_to_add
    return_path_add
    group = mail
    mode = 0660
```

Строки `*_add` добавляют заголовки в сообщение. Разделы `group` и `mode` гарантируют, что транспортный агент может записывать данные в файл.

### *Транспортный механизм smtp*

Это основной компонент всей почтовой системы. Здесь мы определили два экземпляра — для стандартного SMPT-порта (25) и для порта подачи почты (587).

```
my_remote_delivery:
   driver = smtp

my_remote_delivery_port587:
  driver = smtp
  port = 587
  headers_add = X-processed-by: MACRO_HEADER port 587
```

Второй экземпляр, `my_remote_delivery_port587`, задает порт и указывает, что в сообщение должен быть добавлен заголовок, включающий индикацию исходящего порта. Кроме того, где-то в файле конфигурации должен быть определен макрос `MACRO_HEADER`.

## Конфигурация retry

В файле конфигурации должен существовать раздел `retry`, иначе программа Exim никогда не будет повторять попытку доставить почту, которая не была доставлена с первого раза. Можно указать три временных интервала, каждый из которых длиннее предыдущего. После истечение последнего интервала сообщения будут отправлены обратно отправителю как недоставленные.

В инструкциях `retry` для обозначения минут, часов, дней и недель используются суффиксы `m`, `h`, `d` и `w`. Для разных узлов и доменов можно задавать разные интервалы.

Вот как выглядит раздел `retry`.

```
begin retry
   *     *    F, 2h, 15m;    F, 24h, 1h;    F, 4d, 6h
```

Этот пример означает следующее: "Для любого домена доставку почты на временно недоступный адрес следует повторять каждые 15 минут в течение двух дней, каждые десять часов в течение следующих 24 часов, а затем каждые 6 часов через 4 дня и в заключение отправить сообщение обратно как недоставленное".

## Конфигурация перезаписи

Раздел перезаписи в файле конфигурации начинается словами `begin rewrite`. Он используется для исправления адресов, а не для перенаправления сообщений. Например, его можно использовать для исходящих адресов.

- Для того чтобы точно указать, что сообщение исходит из вашего домена, а не от индивидуальных узлов.
- Для отображения имен пользователей в стандартном формате, например Имя. Фамилия.

Перезапись не следует применять для адресов входящей почты.

## Функция локального сканирования

Если вы хотите еще более точно настроить программу **exim**, например, чтобы фильтровать самые современные и самые распространенные вирусы, можете написать функцию на языке C, реализующую ваш собственный алгоритм сканирования, и инсталлировать ее в разделе `local_scan` файла конфигурации. Детали и примеры описаны в документации программы Exim.

## Сочетание программ amavisd и Exim

Для того чтобы настроить программу **exim** так, чтобы посылать всю почту, предназначенную для вашего домена, программе **amavisd** сканирования на вирусы и спам, создайте первый маршрутизатор в файле конфигурации, как показано ниже.

```
amavis:
   no_verify_recipient
   driver = manualroute
   condition = ${if or {eq {$interface_port} {10025} \
     {eq {$received_protocol} {scanned-ok} } } {0} {1} }
   domain = local_domains
   transport = amavis
   route_list = * localhost byname
   self = send
```

Строка `condition` означает, что сообщения, исходящие из порта 10025, не должны перенаправляться программе **amavisd**. Это порт, на который возвращаются сообщения от программы **amavisd** после сканирования, поэтому эти сообщения не должны отправляться обратно программе **amavisd**, чтобы не возник замкнутый круг. Транспортный механизм `amavis` настраивается следующим образом.

```
amavis:
  driver = smtp
  port = 10024
  allow_localhost
```

Кроме того, в начало файла конфигурации необходимо добавить строку

```
local_interfaces = 0.0.0.0.25 : 127.0.0.1.10025
```

чтобы приказать программе **exim** принимать сообщения с любых адресов, если они поступили через порт 25, и от локальных узлов, если они поступили через порт 10025, который является портом возврата сообщений для программы **amavisd**.

Эта конфигурация переводит все сканирование в автономный режим, поэтому сообщения о недоставке в ответ на выявление вирусов и спама сами по себе могут создать обратную рассылку спам. Вероятно, лучше поместить инструкцию `amavis` чуть ниже в упорядоченном списке маршрутизаторов и предоставить программе Exim возможность использовать богатый язык списков управления доступом для проверки относительно коротких сообщений. Еще лучше использовать конструкцию `${run...}`, предусмотренную в программе **exim**, чтобы заставить сканер **amavisd** выполнять проверку в оперативном режиме. Кроме того, встроенные возможности сканирования в программе **exim** несколько снижают конкурентоспособность программы **amavisd**.

## Регистрация

По умолчанию программа Exim записывает три разных регистрационных файла: главный журнал, журнал отказов и журнал тревоги. Каждая запись в журнале содержит время ее создания. Местоположение регистрационных файлов задается в файле **EDITME** (до компиляции программы **exim**) или в файле конфигурации времени выполнения с помощью параметра `log_file_path`. По умолчанию файлы регистрации хранятся в каталоге **/var/spool/exim/log**.

Параметр `log_file_path` может принимать до двух значений, разделенных двоеточием. Каждое значение должно быть либо ключевым словом `syslog`, либо абсолютным путем со встроенными символами %, вместо которых подставляются имена **main**, **reject** и **panic**. Например, инструкция

```
log_file_path = syslog : /var/log/exim_%s
```

устанавливает, что в качестве регистрационных файлов должны использоваться системный журнал (с функцией "mail") и файлы **exim_main**, **exim_reject** и **exim_panic** в каталоге **/var/log**. Программа Exim передает записи журнала **main** в системный журнал в качестве первоочередной информации, записи **reject** — в качестве первоочередных сообщений, а записи **panic** — в качестве первоочередных сигналов.

Журнал **main** содержит по одной строке для поступления и доставки каждого сообщения. Отчет об информации, содержащейся в этом журнале, можно сгенерировать с помощью сценария **eximstats** на языке Perl, включенного в дистрибутивный пакет Exim. Записи из журнала **reject** хранят информацию о сообщениях, которые были отклонены из-за нарушения правил, — злонамеренные программы, спам и пр. Этот журнал содержит итоговую строку для сообщения из журнала **main**, а также исходные

заголовки отклоненных сообщений. Если правила были изменены, проверьте журнал **reject**, чтобы убедиться, что все в порядке.

Журнал **panic** предназначен для регистрации серьезных ошибок программного обеспечения; программа **exim** записывает в него информацию непосредственно перед тем, как завершить работу. Если проблемы не возникли, журнал **panic** существовать не должен. Попросите утилиту **cron** проверить, существует ли файл **panic**, устраните проблему, вызвавшую тревогу, а затем удалите этот файл. Программа **exim** заново создаст его, если снова возникнет тревожная ситуация. При отладке можно увеличить объем и расширить набор типов регистрируемых данных с помощью параметра `log_selector`. Рассмотрим пример.

```
log_selector = +smtp_connection +snmp_incomplete_transaction +…
```

Категории регистрируемых данных могут затем включаться или исключаться с помощью механизма `log_selector`, описанного в спецификации программы Exim в разделе "Log files". Существует примерно 35 возможностей, включая `+all`, которая реально переполнит информацией ваши диски!

Кроме того, программа **exim** хранит временный регистрационный файл для каждого обработанного сообщения. Он называется по идентификатору сообщения и находится в каталоге **/var/spool/exim/msglog**. Если возникают проблемы с конкретным пунктом назначения, следует проверить информацию в этом каталоге.

## Отладка

Программа Exim имеет мощные средства для отладки. Вы можете управлять объемом информации, которую хотите видеть для каждой настраиваемой возможности. Команда **exim -d** заставляет программу **exim** переключиться в режим отладки, в котором она получает высокий приоритет и не позволяет делать свои копии. К параметру **-d** можно добавить дополнительные категории отладки, вставляя символы + или - перед названием категории. Например, параметры **-d+expand+acl** предусматривают обычный вывод информации об отладке, который сопровождается дополнительными деталями, касающимися расширений строк и интерпретации списков управления доступом. (Эти две категории являются самыми проблемными.) Для отладочной информации можно настроить более 30 категорий; их полный список находится на соответствующей справочной странице.

При отладке почтовой системы обычно запускают почтовый транспортный агент на нестандартном порту, а затем обращаются к нему с помощью команды **telnet**. Например, для того чтобы запустить программу **exim** в режиме демона, прослушивающего порт 26, и включить вывод информации об отладке, можно использовать следующую инструкцию.

```
$ sudo exim -d -oX 26 -bd
```

Затем можно применить команду **telnet** к порту 26 и набрать команды SMTP, пытаясь воспроизвести проблемы, решение которых следует отладить.

В качестве альтернативы можно использовать команду **swaks**, которая вступит в обмен информацией с протоколом SMTP. Эта команда запускает сценарий на языке Perl, ускоряющий и облегчающий отладку протокола SMTP. Инструкция **swaks --help** выводит на экране короткое описание, а полную информацию можно найти на сайте `jetmore.org/john/code/#swaks`.

Если ваши регистрационные файлы делают тайм-аут примерно каждые 30 секунд, следует подозревать проблемы в системе DNS. Тайм-ауты длительностью 5 секунд могут

быть тайм-аутами запроса **identd**. (Демон **identd** в прошлом предназначался для идентификации реального отправителя почтового сообщения, однако поскольку его было очень легко обмануть, он больше не используется.)

# 20.15. Почтовый агент Postfix

Постфикс — популярная альтернатива программе **sendmail**. Вице Венема (Wietse Venema) начал работу над проектом Postfix в 1996 году, когда проводил годичный творческий отпуск в Исследовательском центре им. Т. Дж. Уотсона компании IBM (T. J. Watson Research Center), и до сих пор активно работает над ним. Целью разработки программы Postfix является не только безопасность (хотя это ее первоочередная и самая главная цель!), но и реализация политики распространения открытого программного обеспечения, высокое быстродействие, надежность и гибкость. Все основные дистрибутивные пакеты системы Linux включают Postfix, а начиная с версии 10.3 система Mac OS X по умолчанию оснащается почтовой системой Postfix, а не **sendmail**.

🕮 Более подробно регулярные выражения описаны в разделе 2.3.

О системе Postfix следует знать два самых важных факта: во-первых, она поставляется в практически работоспособном виде (простейшие файлы конфигурации состоят из одной-двух строк) и, во-вторых, эффективно использует ассоциативные массивы регулярных выражений для фильтрации почты, особенно в сочетании с библиотекой PCRE (Perl Compatible Regular Expression). Система Postfix совместима с программой **sendmail** в том смысле, что файлы **aliases** и **.forward** системы Postfix имеют тот же формат и семантику, что и аналогичные файлы в системе **sendmail**.

Система Postfix поддерживает протокол ESMTP. Кроме того, поддерживаются виртуальные домены и фильтация спама.

Для перезаписи адресов система Postfix использует поиск в таблицах, которые могут быть записаны в обычных файлах или в форматах Berkeley DB, DBM, LDAP, NIS, NetInfo и SQL. Кроме того, система Postfix поддерживает протокол почтового фильтра программы **sendmail**, поэтому ее функционирование можно легко настроить с помощью множества открытых почтовых фильтров (внешних программ, решающих специальные задачи в ходе сеанса SMTP; см. раздел 20.6).

## Архитектура системы Postfix.

Система Postfix состоит из нескольких небольших взаимодействующих программ, посылающих сообщения по сети, получающих сообщения, выполняющих локальную доставку почты и т.д. Взаимодействие между ними осуществляется через сокеты локальных доменов или именованные каналы FIFO. Эта архитектура довольно сильно отличается от архитектуры программ **sendmail** и Exim, в которых основную работу выполняет большая монолитная программа.

Запуск системы Postfix и слежение за всеми ее процессами выполняет программа **master**. В ее конфигурационном файле, **master.cf**, перечислены все вспомогательные программы, а также информация о том, как они должны запускаться. Значения, по умолчанию установленные в этом файле, удовлетворяют большую часть потребностей пользователей, так что изощряться не обязательно. В основном, приходится отключать программы, например **smtpd**, которые клиент не должен прослушивать на порту SMTP.

Самые важные серверные программы, вовлеченные в доставку электронной почты, показаны на рис. 20.4.

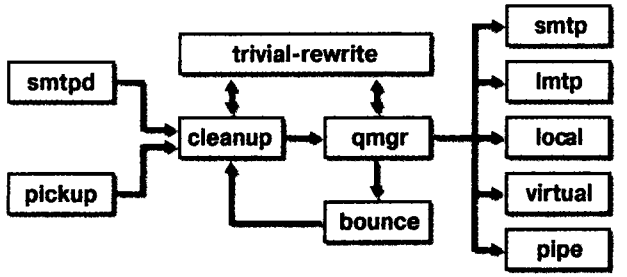



*Рис. 20.4. Серверные программы системы Postfix*

### Получение почты

Программа **smtpd** получает почту, поступающую в систему через сервер SMTP. Она также проверяет, имеют ли право клиенты, установившие соединение, отправлять почту, которую они пытаются доставить. Если электронная почта послана локально с помощью программы **/usr/lib/sendmail**, то файл записывается в каталог **/var/spool/postfix/maildrop**. Этот каталог периодически сканируется программой **pickup**, обрабатывающей любой найденный новый файл.

Вся входящая почта проходит через программу **cleanup**, которая добавляет пропущенные заголовки и перезаписывает адреса в соответствии с таблицами `canonical` и `virtual`. Прежде чем вставить сообщение в очередь входящих сообщений, программа **cleanup** передает его программе **trivial-rewrite**, которая выполняет небольшое редактирование адресов, например добавляет почтовый домен к не полностью заданным адресам.

### Управление почтовыми очередями ожидания

Программа **qmgr** управляет пятью очередями, содержащими почту, ожидающую доставки.

- `incoming` — поступающая почта
- `active` — доставленная почта
- `deferred` — почта, доставка которой завершилась неудачей
- `hold` — почта, заблокированная в очереди администратором
- `corrupt` — почта, которую невозможно прочитать или проанализировать

Менеджер очереди обычно выбирает следующее сообщение для обработки, руководствуясь простой стратегией FIFO, но помимо нее он также поддерживает сложный алгоритм приоритетного обслуживания, который среди всей массы сообщений отдает предпочтение сообщениям, направленным избранным получателям.

Для того чтобы не перегружать узел, предназначенный для получения почты, особенно после его выхода из строя, система Postfix использует алгоритм медленного пуска, управляющий скоростью доставки почты.

Отложенная почта получает временную метку о повторной доставке, которая имеет экспоненциальное затухание, чтобы не затрачивать ресурсы на недоставленные сообщения. Избежать излишних попыток доставить почту, которую невозможно доставить, позволяет кеш статуса.

### Отправка сообщений

Программа **qmgr** с помощью программы **trivial-rewrite** решает, куда следует послать сообщение. Вместо решений, принимаемых программой **trivial-rewrite**, можно использовать поиск в таблицах (`transport_maps`).

Доставка почты удаленным узлам через сервер SMTP выполняется программой **smtp**. Программа **lmtp** доставляет почту, используя протокол LMTP (Local Mail Transfer Protocol), определенный в документе RFC2033. Протокол LMTP основан на протоколе SMTP, но был модифицирован так, что почтовый сервер не обязан управлять почтовой очередью. Этот обработчик почты очень полезен для доставки сообщений на почтовые серверы, такие как Cyrus IMAP.

Программа **local** предназначена для локальной доставки электронной почты. Она находит адреса в таблице **aliases** и следует инструкциям, указанным в файлах получателя **.forward**. Сообщения пересылаются на другой адрес, передаются внешней программе для обработки или сохраняются в почтовых папках пользователя.

Программа **virtual** доставляет почту в "виртуальные почтовые ящики", т.е. в почтовые ящики, не связанные с локальной учетной записью в системе Linux, но имеющие корректные адреса.

В заключение программа **pipe** выполняет доставку посредством внешних программ.

## Безопасность

Система Postfix обеспечивает безопасность на нескольких уровнях. Большинство серверных программ системы Postfix могут выполняться в окружении, созданном с помощью команды **chroot**. Это отдельные программы, не имеющие отношений типа "предок-потомок". Ни одна из них не имеет механизма setuid. Каталог `maildrop` заполняется группой postdrop, для которой программа **postdrop** установила идентификатор setgid.

Впечатляет тот факт, что ни в одной версии системы Postfix еще не обнаружены дефекты, которые можно было бы использовать дистанционно.

## Команды и документация системы Postfix

Взаимодействие пользователя с системой обеспечивают несколько утилит, запускаемых из командной строки.

- **sendmail**, **mailq**, **newaliases** — выполняет замены, совместимые с системой **sendmail**.
- **postfix** — запускает и останавливает почтовую систему (должна запускаться из корня).
- **postalias** — создает, модифицирует и ставит в очередь таблицы псевдонимов..
- **postcat** — выводит на печать содержимое файлов очереди..
- **postconf** — выводит на печать и редактирует главный файл конфигурации **main.cf**.
- **postmap** — создает, модифицирует или ставит в очередь таблицы поиска..
- **postsuper** — управляет почтовыми очередями..

Дистрибутивный пакет Postfix содержит набор справочных страниц, описывающих все программы и их параметры. Документация, размещенная на сайте `postfix.org`,

объясняет, как конфигурировать разные аспекты программы Postfix и управлять ими. Эти документы также входят в дистрибутивный пакет Postfix и находятся в каталоге **README_FILES**.

## Конфигурация системы Postfix

Файл **main.cf** – главный конфигурационный файл системы Postfix. Файл **master.cf** конфигурирует серверные программы. Он также определяет разные таблицы поиска, на которые ссылается файл **main.cf**, и поддерживает разные типы служебных отображений.

Справочная страница **postconf**(5) описывает каждый параметр, заданный в файле **main.cf**. Существует также программа **postconf**, которая с помощью команды **man postconf** позволяет получить справочную страницу, заменяющую страницу **postconf**(5). Для того чтобы получить правильную версию справочной страницы, следует использовать команду **man -s 5 postconf** (соответственно, **man 5 postconf** в системах HP-UX и AIX).

Язык конфигурирования системы Postfix выглядит как ряд комментариев **sh** и инструкций присваивания. В определении одной переменной можно ссылаться на другие переменные с помощью префикса $. Определения переменных заносятся в память сразу, как только они обнаруживаются в файле конфигурации; они не раскрываются, пока не будут использованы, и в это время могут производиться любые подстановки.

Можно создавать новые переменные, присваивая им значения. При выборе имен следует быть осторожным, чтобы не создать конфликт с существующими переменными конфигурации.

Все файлы конфигурации системы Postfix, включая таблицы поиска, интерпретируют строки, начинающиеся с пробела, как продолжение предыдущей строки. Это позволяет создавать очень удобные для чтения файлы конфигурации, но при этом новую строку приходится начинать с первой позиции.

### *Что должно находиться в файле main.cf*

В файле **main.cf** можно задать более 500 параметров. Однако для среднестатистического сайта необходимы только некоторые из них. Автор системы Postfix настоятельно рекомендует включать в свои конфигурации только те параметры, которые имеют значения, не заданные по умолчанию. Таким образом, если в будущем значение параметра, заданное по умолчанию, изменится, ваша конфигурация учтет новое значение автоматически.

Образец файла **main.cf**, поставляемый в дистрибутивном пакете, содержит много закомментированных параметров, а также короткое описание. Исходную версию файла лучше всего оставить в качестве ориентира. Начните создание собственной конфигурации с пустого файла, чтобы ваши установки не затерялись среди многочисленных комментариев.

### *Основные установки*

Начнем с самой простой конфигурации: пустого файла. Как это ни удивительно, это вполне допустимая конфигурация системы Postfix. Результатом такой конфигурации является почтовый сервер, доставляющий электронную почту локально внутри того же самого домена, в котором находится локальная хост-система, и посылающий все сообщения на нелокальные адреса непосредственно на соответствующий удаленный сервер.

Еще одна простая конфигурация называется "нулевым клиентом"; иначе говоря, система, не выполняющая локальной доставки никаких электронных сообщений и перенаправляющая внешнюю почту на предназначенный для этого центральный сервер. Для реализации этой конфигурации определим несколько параметров: параметр `mydomain`, задающий доменную часть имени узла, и `myorigin`, представляющий собой имя почтового домена, добавляемое в не полностью заданные адреса. Если эти два параметра совпадают, то можно написать, например, следующие строки.

```
mydomain = cs.colorado.edu
myorigin = $mydomain
```

Еще один параметр, который необходимо определить, — `mydestination`, который задает локальные почтовые домены. Если в адресе получателя сообщения имя почтового домена совпадает со значением параметра `mydestination`, сообщение доставляется с помощью программы **`local`** соответствующему пользователю (в предположении, что не найдены ни релевантный псевдоним, ни файл **`.forward`**). Если в параметре `mydestination` указаны имена нескольких доменов, то все они считаются псевдонимами одного и того же домена.

Для нулевого клиента локальная доставка не предусмотрена, поэтому этот параметр должен оставаться пустым.

```
mydestination =
```

В заключение, параметр `relayhost` означает, что система Postfix должна посылать все нелокальные сообщения на указанный узел, а не непосредственно в их заданные явно пункты назначения.

```
relayhost = [mail.cs.colorado.edu]
```

Квадратные скобки означают, что система Postfix должна интерпретировать указанную строку как имя узла (запись A в DNS ), а не как имя домена (запись MX в DNS).

Поскольку нулевые клиенты не должны получать почту от других систем, в конфигурации нулевого клиента осталось только закомментировать строку **`smtpd`** в файле **`master.cf`**. Это изменение не позволяет системе Postfix вообще запускать программу **`smtpd`**. Итак, с помощью всего нескольких строк мы определили полнофункционального нулевого клиента!

Для "реального" почтового сервера необходимо задать немного больше параметров, а также некоторые таблицы отображений. Эти темы рассмотрим в следующих подразделах.

### *Использование утилиты postconf*

Утилита **`postconf`** — это удобный инструмент, помогающий конфигурировать систему Postfix. Если запустить ее без аргументов, она выводит на печать все параметры, которые она сконфигурировала к данному моменту. Если в качестве аргумента указать конкретный параметр, утилита **`postconf`** выведет на печать его значение.

Флаг **`-d`** заставляет утилиту **`postconf`** выводить значения параметров, заданные по умолчанию, а не текущие значения, указанные в конфигурации. Рассмотрим пример.

```
$ postconf mydestination
mydestination =
$ postconf -d mydestination
mydestination = $myhostname, localhost.$mydomain, localhost
```

Еще один полезный флаг **`-n`** заставляет утилиту **`postconf`** выводить только те параметры, которые отличаются от заданных по умолчанию. Если вам необходима справка

по списку рассылок в системе Postfix, именно эту информацию следует указать в вашем электронном письме.

### Таблицы поиска

Многие аспекты функционирования системы Postfix зависят от использования таблиц поиска, которые могут отображать ключи в значения или реализовать простые списки. Например, настройки по умолчанию для таблицы `alias_maps` имеют следующий вид.

```
alias_maps = dbm:/etc/mail/aliases, nis:mail.aliases
```

Источники данных задаются с помощью обозначения *тип:путь*. Обратите внимание на то, что конкретная таблица одновременно использует два разных источника информации: базу данных **dbm** и ассоциативный массив NIS. Множественные значения можно разделять запятыми, пробелами или и тем, и другим. Доступные источники данных перечислены в табл. 20.21; команда **`postconf -m`** также выводит эту информацию.

**Таблица 20.21. Источники информации для таблиц поиска в системе Postfix**

| Тип | Описание |
|---|---|
| `dbm/sdbm` | Обычные файлы баз данных **dbm** или **gdbm** |
| `cidr` | Сетевые адреса в формате CIDR |
| `hash/btree` | Хеш-таблицы Berkeley DB или файл B-дерева (эквивалент **dbm**) |
| `ldap` | Служба каталогов LDAP |
| `mysql` | База данных MySQL |
| `nis` | Служба каталогов NIS |
| `pcre` | Регулярные выражения, совместимые с языком Perl |
| `pgsql` | База данных PostgreSQL |
| `proxy` | Доступ через сервер **proxymap**, например, чтобы избежать использования команды **chroot** |
| `regexp` | Регулярные выражения POSIX |
| `static` | Возвращает значение, указанное как *путь*, независимо от ключа |
| `unix` | Файлы **/etc/passwd** и **/etc/group**; использует синтаксис NIS [a] |

[a] `unix:passwd.byname` — это файл passwd, а unix:group.byname — файл group.

Типы `dbm` и `sdbm` следует использовать только для обеспечения совместимости с обычной таблицей псевдонимов программы **`sendmail`**. Berkeley DB (`hash`) — более современная реализация; она безопаснее и быстрее. Ее совместимость — это не проблема, используйте следующие инструкции.

```
alias_database = hash:/etc/mail/aliases
alias_maps = hash:/etc/mail/aliases
```

Параметр `alias_database` задает таблицу, которая перестраивается с помощью команды **`newaliases`** и должна соответствовать таблице, заданной параметром `alias_maps`. Причина, по которой используются два параметра, заключается в том, что таблица `alias_maps` может содержать источники, не являющиеся базами данных, например `mysql` или `nis`, которые не требуют перестройки.

Все таблицы баз данных (`dbm`, `sdbm`, `hash` и `btree`) представляют собой текстовые файлы, которые компилируются в эффективный бинарный формат, предназначенный для поиска. Синтаксис этих текстовых файлов похож на синтаксис файлов конфигура-

ции тем, что в них также используются комментарии и продолжения строк. Записи задаются как простые пары "ключ/значение", разделенные пробелами, за исключением таблиц псевдонимов, в которых используются двоеточия после ключей, чтобы обеспечить совместимость с программой **sendmail**. Например, следующие строки являются характерными для таблицы псевдонимов.

```
postmaster: david, tobias
webmaster: evi
```

В следующем примере задается таблица доступа для ретрансляции почты от любого клиента, имя узла которого заканчивается строкой `cs.colorado.edu`.

```
.cs.colorado.edu OK
```

Текстовые файлы компилируются в бинарный формат с помощью команд **postmap** (для обычных таблиц) и **postalias** (для таблиц псевдонимов). Спецификация таблицы (включая ее тип) должна передаваться как первый аргумент. Рассмотрим пример.

```
$ sudo postmap hash:/etc/postfix/access
```

Команда **postmap** может также запрашивать значения в таблице поиска (нет совпадения — нет вывода на экран).

```
$ postmap -q blabla hash:/etc/postfix/access
$ postmap -q .cs.colorado.edu hash:/etc/postfix/access
OK
```

### *Локальная доставка*

Программа **local** доставляет почту локальным получателям. Кроме того, она обрабатывает локальные псевдонимы.

Например, если параметр `mydestination` задан равным `cs.colorado.edu`, а почта поступает на адрес `evi@cs.colorado.edu`, то программа **local** сначала проверяет таблицы `alias_maps`, а затем рекурсивно заменяет все соответствующие записи.

Если в таблице нет соответствия псевдонимов, программа **local** просматривает файл **.forward** в рабочем каталоге пользователя evi и следует инструкциям, заданным в этом файле, если он существует. (Синтаксис совпадает с правой частью таблицы псевдонимов.) В заключение, если файл **.forward** не найден, то электронное сообщение доставляется в локальный почтовый ящик пользователя evi.

По умолчанию программа **local** записывает данные в файлы, имеющие формат **mbox** и находящиеся в каталоге **/var/mail**. Эти настройки можно изменить с помощью параметров, указанных в табл. 20.22.

**Таблица 20.22. Параметры для доставки почты в локальный почтовый ящик (задаются в файле main.cf)**

| Параметр | Описание |
|---|---|
| `mail_spool_directory` | Доставляет почту в центральный каталог, обслуживающий всех пользователей |
| `home_mailbox` | Доставляет почту в каталог ~пользователь по заданному относительному пути |
| `mailbox_command` | Доставляет почту с помощью внешней программы, обычно **procmail** |
| `mailbox_transport` | Доставляет почту с помощью службы, определенной в файле **master.cf** [a] |
| `recipient_delimiter` | Разрешает расширенные имена (см. описание ниже) |

[a] Этот параметр взаимодействует с почтовыми серверами, такими как Cyrus imapd.

Параметры mail_spool_directory и home_mailbox обычно генерируют почтовые ящики в формате **mbox**, но они также могут создавать почтовые ящики **Maildir**. Для того чтобы сделать это, необходимо добавить косую черту в конце имени пути.

Если параметр recipient_delimiter равен +, то почта, адресованная evi+whatever@cs.colorado.edu, принимается для доставки на учетную запись evi. С помощью этой функциональной возможности пользователь может создавать специальные адреса и сортировать свою почту по адресу получателя. Система Postfix сначала пытается выполнить поиск по полному адресу, и только если поиск оказался безуспешным, она разбирает дополнительные компоненты и "возвращается" к базовому адресу. Система Postfix также ищет соответствующий файл пересылки **.forward+***что_угодно* для дальнейшей обработки псевдонимов.

## Виртуальные домены

Есть три возможности организовать почтовый домен на почтовом сервере Postfix.

- Указать домен в параметре mydestination. Доставка выполняется, как описано выше: псевдонимы раскрываются и почта доставляется на соответствующие учетные записи.
- Указать домен в параметре virtual_alias_domains. Это позволяет оснастить домен собственным пространством адресов, которое не зависит от системных учетных записей пользователей. Все адреса внутри домена должны отображаться (с помощью ассоциативных массивов) во внешние реальные адреса.
- Указать домен в параметре virtual_mailbox_domains. Как и в случае параметра virtual_alias_domains, домен имеет собственное пространство имен. Все почтовые ящики должны находиться в указанном каталоге.

Домен можно указать только в одном из трех указанных параметров. Решение следует хорошенько обдумать, потому что от него зависят многие элементы конфигурации. Мы уже описывали применение метода mydestination. Рассмотрим остальные возможности.

### *Псевдонимы виртуальных доменов*

Если домен указан как значение параметра virtual_alias_domains, то почта, направляемая в этот домен, будет приниматься системой Postfix и должна направляться реальному получателю либо на локальном компьютере, либо где-то еще.

Перенаправление адресов в виртуальный домен должно определяться в таблице поиска, указанной в параметре virtual_alias_maps. Записи этой таблицы в левой части содержат адрес виртуального домена, а в правой — реальный адрес получателя.

Не полностью заданное имя в правой части интерпретируется как локальное имя пользователя.

Рассмотрим следующий пример из файла **main.cf**.

```
myorigin = cs.colorado.edu
mydestination = cs.colorado.edu
virtual_alias_domains = admin.com
virtual_alias_maps = hash:/etc/mail/admin.com/virtual
```

В файле **/etc/mail/admin.com/virtual** могут содержаться следующие строки.

```
postmaster@admin.com evi, david@admin.com
david@admin.com david@schweikert.ch
evi@admin.com evi
```

Почта, направляемая на адрес evi@admin.com, будет перенаправлена на адрес evi@cs.colorado.edu (добавлен параметр myorigin) и в итоге будет доставлена в почтовый ящик пользователя evi, потому что имя домена cs.colorado.edu указано как значение параметра mydestination.

Определения могут быть рекурсивными: правая часть может содержать адреса, которые в дальнейшем определяются в левой части. Обратите внимание на то, что правая часть может быть только списком адресов. Если необходимо выполнить внешнюю программу или использовать подстановку файлов с помощью инструкции :include:, то нужно перенаправить почту на псевдоним, который впоследствии может быть раскрыт в соответствии с потребностями пользователя.

Для того чтобы хранить всю информацию в одном файле, можно присвоить параметру virtual_alias_domains имя той же таблицы поиска, которая задана параметром virtual_alias_maps, и внести в эту таблицу специальную запись, которая отмечала бы этот файл как псевдоним виртуального домена. В файле **main.cf** соответствующие строки выглядели бы следующим образом.

```
virtual_alias_domains = $virtual_alias_maps
virtual_alias_maps = hash:/etc/mail/admin.com/virtual
```

Файл **/etc/mail/admin.com/virtual** содержал бы следующие данные.

```
admin.com notused
postmaster@admin.com evi, david@admin.com
...
```

Правая часть записи для почтового домена (admin.com) на самом деле никогда не используется; наличия имени admin.com в этой таблице в качестве независимого поля уже достаточно для того, чтобы система Postfix рассматривала его как псевдоним виртуального домена.

### *Виртуальные почтовые домены*

Домены, перечисленные в списке значений параметра virtual_mailbox_domains, аналогичны локальным доменам, но список пользователей и их соответствующие почтовые ящики должны управляться независимо от системных учетных записей пользователей.

Параметр virtual_mailbox_maps указывает на таблицу, в которой перечислены все корректные пользователи в домене. Соответствующее отображение имеет следующий вид.

```
user@domain /path/to/mailbox
```

Если имя пути заканчивается косой чертой, почтовые ящики хранятся в формате **Maildir**. Значение параметра virtual_mailbox_base всегда является префиксом заданных путей.

Часто возникает необходимость заменить псевдонимом некоторые адреса в виртуальных почтовых доменах. Для этого следует использовать параметр virtual_alias_map. Рассмотрим полный пример. В файле **main.cf**

```
virtual_mailbox_domains = admin.com
virtual_mailbox_base = /var/mail/virtual
virtual_mailbox_maps = hash:/etc/mail/admin.com/vmailboxes
virtual_alias_maps = hash:/etc/mail/admin.com/valiases
```

имеется ссылка на файл **/etc/mail/admin.com/vmailboxes**, который может содержать следующую запись.

```
evi@admin.com nemeth/evi/
```

Файл **/etc/mail/admin.com/valiases** может содержать следующую строку.

```
postmaster@admin.com evi@admin.com
```

Отображения виртуальных псевдонимов можно применять даже к адресам, которые не являются псевдонимами виртуальных доменов. Отображения виртуальных псевдонимов позволяют перенаправлять любой адрес от любого домена независимо от типа домена (канонического, виртуального псевдонима или виртуального почтового ящика). Поскольку пути к почтовому ящику могут находиться только в правой части отображения виртуального почтового ящика, использование этого механизма является единственным способом задать псевдоним для такого домена.

## Управление доступом

Почтовые серверы должны ретранслировать почту для третьих лиц только со стороны доверенных клиентов. Если почтовый сервер перенаправляет почту от неизвестных клиентов на другие серверы, то возникает так называемая открытая ретрансляция (open relay), а это плохо. Более подробно открытая ретрансляция описана в разделе 20.10.

К счастью, система Postfix по умолчанию не допускает открытой ретрансляции. На самом деле она имеет довольно строгие настройки, установленные по умолчанию; их приходится скорее смягчать, чем ужесточать. Управление доступом для транзакций SMTP конфигурируется в системе Postfix с помощью "списков ограничения доступа". Параметры, указанные в табл. 20.23, управляют тем, что следует проверять на разных этапах сеанса SMTP.

**Таблица 20.23. Параметры системы Postfix для ограничения доступа в рамках протокола SMTP**

| Параметр | Когда применяется |
|---|---|
| `smtpd_client_restrictionsq` | По запросу на соединение |
| `smtpd_helo_restrictions` | К команде ELO/EHLO (начало сессии) |
| `smtpd_sender_restrictions` | К команде MAIL FROM (спецификация отправителя) |
| `smtpd_recipient_restrictions` | К команде RCPT TO (спецификация получателя) |
| `smtpd_data_restrictions` | К команде DATA (тело почты) |
| `smtpd_etrn_restrictions` | К команде ETRN [a] |

[a] Это специальная команда, используемая для пересылки сообщений в очереди.

Наиболее важным параметром является `smtpd_recipient_restrictions`, поскольку управление доступом легче всего реализовать, если адрес получателя известен и можно определить, локальный он или нет. Все другие параметры из табл. 20.23 в конфигурации, заданной по умолчанию, остаются пустыми. По умолчанию параметр `smtpd_recipient_restrictions` имеет следующее значение.

```
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

Каждое из указанных ограничений проверяется по очереди, пока не будет принято определенное решение о том, что делать с почтой. Общие ограничения перечислены в табл. 20.24.

С помощью этих ограничений можно проверить все, а не только специфическую информацию наподобие адреса отправителя в параметре `smtpd_sender_restrictions`. Следовательно, для простоты, можно наложить все ограничения, используя только один

параметр, которым должен быть параметр smtpd_recipient_restrictions, поскольку только он позволяет проверять все (за исключением части DATA).

**Таблица 20.24. Общие ограничения доступа в системе Postfix**

| Ограничение | Функция |
|---|---|
| check_client_access | Проверяет адрес узла клиента, просматривая таблицу поиска |
| check_recipient_access | Проверяет почтовый адрес получателя, просматривая таблицу поиска |
| permit_mynetworks | Предоставляет доступ адресам, перечисленным в списке mynetworks |
| reject_unauth_destination | Отклоняет почту для нелокальных получателей; отменяет ретрансляцию |

Параметр smtpd_recipient_restriction позволяет также проверять ретранслируемую почту. Ограничения, заданные параметром reject_unauth_destination, следует сохранить и с осторожностью выбирать "разрешающие" ограничения, предшествующие ему.

### *Таблицы доступа*

Каждое ограничение возвращает одно действие, указанное в табл. 20.25. Таблицы доступа используются в ограничениях check_client_access и check_recipient_access для выбора действия, зависящего от адреса клиентского узла или адреса получателя соответственно.

**Таблица 20.25. Действия для таблицы доступа**

| Действие | Описание |
|---|---|
| 4*nn текст* | Возвращает код временной ошибки 4*nn* и сообщение *text* |
| 5*nn текст* | Возвращает код постоянной ошибки 5*nn* и сообщение *text* |
| DEFER_IF_PERMIT | Если результатом оценки ограничения является действие PERMIT, изменяет его на временную ошибку |
| DEFER_IF_REJECT | Если результатом оценки ограничения является действие REJECT, изменяет его на временную ошибку |
| DISCARD | Принимает сообщение, но скрытно удаляет его |
| DUNNO | Делает вид, что ключ не был найден; проверяет дальнейшие ограничения |
| FILTER *транспорт:пункт_назначения* | Пропускает почту через фильтр *транспорт:пункт_назначения*[a] |
| HOLD | Блокирует почту в очереди |
| OK | Принимает почту |
| PREPEND *заголовок* | Добавляет заголовок в сообщение |
| REDIRECT *адрес* | Перенаправляет данное сообщение на конкретный адрес |
| REJECT | Отклоняет почту |
| WARN *сообщение* | Заносит указанное сообщение в регистрационный журнал |

[a] См. ниже раздел, посвященный борьбе со спамом и вирусами в системе Postfix.

В качестве примера предположим, что мы хотим разрешить ретрансляцию для всех компьютеров в домене cs.colorado.edu и отправку сообщения списку внутренней рассылки newsletter@cs.colorado.edu только доверенным клиентам. Эти правила можно реализовать с помощью следующих строк в файле **main.cf**.

```
smtpd_recipient_restrictions =
        permit_mynetworks
        check_client_access hash:/etc/postfix/relaying_access
        reject_unauth_destination
        check_recipient_access hash:/etc/postfix/restricted_recipients
```

Обратите внимание на то, что запятые являются необязательными, если указан список значений параметра.

В файл **/etc/postfix/relaying_access** следует внести следующую строку.

```
.cs.colorado.edu OK
```

В файл **/etc/postfix/restricted_recipients** следует внести такую строку.

```
newsletter@cs.colorado.edu REJECT Internal list
```

Текст, следующий за словом REJECT, является необязательной строкой, которая посылается клиенту вместе с кодом ошибки. Он сообщает отправителю, почему сообщение было отклонено.

### *Аутентификация клиентов и шифрование*

Для пользователей, отправляющих сообщения из дома, обычно проще всего направлять исходящую почту через домашний почтовый сервер интернет-провайдера, независимо от адреса отправителя сообщения. Большинство интернет-провайдеров доверяют своим непосредственным клиентам и разрешают ретрансляцию. Если эта конфигурация невозможна или используется система Sender ID или SPF, то следует убедиться, что мобильные пользователи, находящиеся вне сети, могут быть авторизованы для представления сообщений демону **smtpd**.

Решить эту проблему можно, используя механизм SMTP AUTH непосредственно на уровне SMTP. Для этого систему Postfix необходимо скомпилировать с библиотекой SASL. Затем эту функциональную возможность можно конфигурировать следующим образом.

```
smtpd_sasl_auth_enable = yes
smtpd_recipient_restrictions =
  permit_mynetworks
  permit_sasl_authenticated
...
```

Вам также понадобится поддержка зашифрованных соединений, чтобы избежать отправки паролей открытым текстом. Добавьте в файл **main.cf** строки, похожие на следующие.

```
smtpd_tls_security_level = may
smtpd_tls_auth_only = yes
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes
smtpd_tls_cert_file = /etc/certs/smtp.pem
smtpd_tls_key_file = $smtpd_tls_cert_file
smtpd_tls_protocols = !SSLv2
```

Необходимо также поместить правильно подписанный сертификат в файл **/etc/certs/smtp.pem**. Кроме того, рекомендуется подключить шифрование на исходящие соединения SMTP.

```
smtp_tls_security_level = may
smtp_tls_loglevel = 1
```

## Борьба со спамом и вирусами

Система Postfix имеет много функциональных возможностей, помогающих блокировать подозрительную электронную почту. Один класс защитных функциональных возможностей предназначен для строгой реализации протокола SMTP. Законопослушный почтовый сервер должен следовать протоколу, но отправители спама и вирусов часто спешат и пренебрегают этим требованием, тем самым выдавая себя. К сожалению, неисправные обработчики почты, обеспечивающие легитимность такой почте, все еще существуют, поэтому такой метод нельзя считать панацеей. Следует тщательно выбирать ограничения и отслеживать регистрационные файлы.

Перечислим некоторые основные функциональные возможности, относящиеся к данной категории.

- `reject_non_fqdn_*` — отклоняет сообщения, не имеющие полностью определенного домена отправителя (`sender`), домена получателя (`recipient`) или имени узла `HELO/EHLO` (`hostname`).
- `reject_unauth_pipelining` — прекращает сеанс, если клиент не ждет, чтобы увидеть статус команды до обработки.
- `reject_unknown_sender_domain` — отклоняет сообщения с нераспознанным доменом отправителя. Система Postfix возвращает сообщение о временной ошибке, потому что эта проблема может быть результатом временного сбоя системы DNS.
- `reject_unknown_reverse_client_hostname` — отклоняет сообщения от узлов, не имеющих обратных записей DNS.
- `smtpd_helo_required` — запрашивает команды HELO/EHLO в начале обмена информацией (параметр, принимающий значение `yes` или `no`).
- `strict_rfc821_envelopes` — требует правильного синтаксиса для адресов электронной почты в командах MAIL FROM и RCPT TO (параметр, принимающий значения `yes` или `no`).

Все перечисленные выше параметры не являются ограничениями. Мы вызываем их, включая их имена в параметры `smtpd_helo_restrictions` (`reject_non_fqdn_*`) или `smtpd_client_restrictions` (другие). Для проверки ограничения до включения его в процесс (настоятельно рекомендуем), вставьте ограничение `warn_if_reject` перед ним, чтобы преобразовать эффект от прямого отказа принимать почту в предупреждения.

### Черные списки

Вы можете заставить систему Postfix сравнивать входящие сообщения с черным списком, основанным на системе DNS; подробнее эта процедура описана в разделе 20.10. Для того чтобы воспользоваться этой функциональной возможностью, используется ограничение `reject_rbl_client` с указанным адресом сервера DNS, с которым проводится сверка. Аналогичная функциональная возможность `reject_rhsbl_sender` проверяет имя домена в адресе отправителя, а не имя узла клиента.

### Пример борьбы со спамом

В следующем примере продемонстрирована относительно полная конфигурация из файла **`main.cf`**, предназначенная для борьбы со спамом.

```
strict_rfc821_envelopes = yes
smtpd_helo_required = yes
```

```
smtpd_recipient_restrictions =
  reject_unknown_sender_domain
  reject_non_fqdn_sender
  reject_non_fqdn_recipient
  permit_mynetworks
  permit_sasl_authenticated
  check_client_access hash:/etc/postfix/relaying_access
  reject_unauth_destination
  reject_unauth_pipelining
  reject_unknown_reverse_client_hostname
  reject_rbl_client zen.spamhaus.org
  permit
```

Обратите внимание на то, что мы поместили некоторые ограничения перед параметром `permit_mynetworks`. Это позволяет проверить, посылают ли собственные клиенты правильно отформатированные сообщения. Это простой способ найти ошибки конфигурации. Заключительное действие `permit` задается по умолчанию и приведено явно лишь для наглядности.

### Программы SpamAssassin и procmail

Система Postfix поддерживает работу программы SpamAssassin и других аналогичных фильтров. Общая информация об этих инструментах приведена в разделе 20.6.

Программу **`procmail`** можно запустить из пользовательских файлов **`.forward`**, но это сложный и уязвимый для ошибок способ запуска. Лучше поместить в файл **`main.cf`** следующую строку.

```
mailbox_command = /usr/bin/procmail -a "$EXTENSION"
```

Система Postfix впоследствии использует программу **`procmail`** для доставки почты вместо записи сообщений непосредственно в почтовую очередь. Аргументы программы **`procmail`** содержат расширение адреса (фрагмент, следующий за знаком +); впоследствии к ним можно обращаться с помощью программы **`procmail`**, указывая ссылку `$1`.

### Демоны политики

В системе Postfix 2.1 реализован механизм для делегирования управления доступом внешним программам. Эти программы, именуемые демонами политики (policy daemons), получают всю информацию, которую система Postfix имеет о сообщении, и возвращают одно из действий, перечисленных в табл. 20.25.

Серые списки — одна из более интересных функциональных возможностей, которую можно реализовать с помощью демона политики. Более подробную информацию о серых списках и ситуациях, в которых они могут понадобиться, можно найти в разделе 20.6.

### Фильтрация содержимого

Система Postfix может использовать регулярные выражения для проверки заголовков и тел электронных сообщений, используемых как контрабанда. Она также передает сообщения другим программам, например специализированным программам для борьбы со спамом или антивирусным приложениям.

Проверки заголовка и тела выполняются в реальном времени в ходе приема сообщения по протоколу SMTP. В случае совпадения каждое проверяемое регулярное сообщение вызывает действие, указанное в табл. 20.25. Например, строка

```
header_checks = regexp:/etc/postfix/header_checks
```

в файле **main.cf** в сочетании со строкой из файла **/etc/postfix/header_checks**

```
/^Subject: reject-me/ REJECT You asked for it
```

приведет к отказу от приема любого сообщения, тема которого начинается со слов "reject-me". Несмотря на то что поддержка регулярных выражений всегда работает отлично, в контексте обработки электронных сообщений она имеет несколько недостатков. В частности, она неэффективна для борьбы со спамом и фильтрации вирусов.

## Фильтрация содержимого с помощью программы amavisd

Промышленная фильтрация вирусов, как правило, осуществляется с помощью программы **amavisd** (см. раздел 20.6), которая написана на языке Perl и связывает программное обеспечение почтового сервера с одним или несколькими антивирусными приложениями. Такие фильтры конфигурируются параметром `content_filter` программы, который заставляет программу Postfix передавать все входящие сообщения указанной службе. Кроме настройки параметра `content_filter`, системный администратор должен модифицировать несколько существующих записей в файле **master.cf** и добавить несколько новых.

Система Postfix и программа **amavisd** взаимодействуют друг с другом посредством стандартных протоколов SMTP и LMTP. Система Postfix посылает почту для анализа программе **amavisd** посредством протокола LMTP. Программа **amavisd** сканирует ее и посылает обратно в систему Postfix в рамках протокола SMTP на альтернативный порт, доступный только для локального компьютера, на котором отключено сканирование содержимого (чтобы избежать замкнутого круга).

Ожидая почту, поступающую из Интернета, программа **amavisd** обычно прослушивает порт 10024, а для системы Postfix предназначен резервный порт 10025. Для обработки исходящей почты и отделения ее от входящей почты необходимо использовать специальный порт для программы **amavisd**, например 10026. Входящая почта может вернуться в систему Postfix через тот же порт возврата, что и исходящая почта.

Конфигурация, описанная ниже, представляет собой настройку "очереди с последующей обработкой" (post queue), которая сканирует почту после того, как она окажется в почтовой очереди системы Postfix. Если вы хотите реализовать сканирование на лету (in-line scanning), при котором почта сканируется в процессе первоначального диалога с клиентом по протоколу SMTP, запустите вспомогательную программу **amavisd-milter**, позволяющую присоединить программу **amavisd** как почтовый фильтр.

На стороне программы **amavisd** необходимо убедиться, что ее конфигурация содержит следующие строки. (Эта конфигурация использует разные порты для исходящих и входящих сообщений.)

```
$inet_socket_port = [10024,10026];
$notify_method = 'smtp:[127.0.0.1]:10025';
$forward_method = 'smtp:[127.0.0.1]:10025';
$interface_policy{'10026'} = 'ORIGINATING';
$policy_bank{'ORIGINATING'} = {
    originating => 1, # indicates client is ours
};
```

Почтовый агент Postfix необходимо настроить на отправку почты программе **amavisd**. Файл **README.Postfix** в дистрибутивном пакете **amavisd-new** включает около 20 строк готовой конфигурации, которые можно поместить в файл **/etc/postfix/master.cf**, чтобы программа **amavisd** была доступной и могла посылать почту обрат-

но системе Postfix. Мы не повторяем их здесь; просто скопируйте их из файла **README** и вставьте в файл **/etc/postfix/master.cf**.

Для того чтобы заставить систему Postfix посылать почту программе **amavisd** на сканирование, добавьте в файл **main.cf** следующую строку.

```
content_filter = amavisfeed:[127.0.0.1]:10024
```

Для того чтобы отличать входящую почту от исходящей, конфигурацию придется немного усложнить. Вместо директивы, приведенной выше, необходимо использовать следующую модификацию ограничения `smtpd_recipient_restrictions` (изменения выделены полужирным шрифтом).

```
smtpd_recipient_restrictions =
    reject_unknown_sender_domain
    reject_non_fqdn_sender
    reject_non_fqdn_recipient
    check_sender_access regexp:/etc/postfix/tag_as_originating.re
    permit_mynetworks
    permit_sasl_authenticated
    check_sender_access regexp:/etc/postfix/tag_as_foreign.re
    check_client_access hash:/etc/postfix/relaying_access
    reject_unauth_destination
    reject_unauth_pipelining
    reject_unknown_reverse_client_hostname
    reject_rbl_client zen.spamhaus.org
    permit
```

Затем поместите в файл **tag_as_originating.re** строку

```
/^/ FILTER amavisfeed:[127.0.0.1]:10026
```

и добавьте в файл **tag_as_foreign.re** следующую строку.

```
/^/ FILTER amavisfeed:[127.0.0.1]:10024
```

Почта от внешних узлов соответствует ограничению `tag_as_foreign.re`, которое заставляет систему Postfix фильтровать почту, посылая ее на порт 10024. Вся почта соответствует ограничению `tag_as_originating.re`, а для почты, поступающей от внешних узлов, применяются ограничения, в именах которых содержится слово `foreign`.

## Отладка

Если в системе Postfix возникает проблема, в первую очередь следует проверить регистрационные файлы. Ответы на возможные вопросы, скорее всего, найдутся в этих файлах; просто надо их поискать. Каждая программа в системе Postfix обычно создает регистрационную запись для каждого обработанного сообщения. Например, ниже приведен пример регистрационных записей для исходящего сообщения.

```
Aug 18 22:41:33 nova postfix/pickup: 0E4A93688: uid=506
   from=<dws@ee.ethz.ch>
Aug 18 22:41:33 nova postfix/cleanup: 0E4A93688: message-id=
   <20040818204132.GA11444@ee.ethz.ch>
Aug 18 22:41:33 nova postfix/qmgr: 0E4A93688: from=<dws@ee.ethz.ch>,
   size=577,nrcpt=1 (queue active)
Aug 18 22:41:33 nova postfix/smtp: 0E4A93688:
   to=<evi@ee.ethz.ch>,relay=tardis.ee.ethz.ch[129.132.2.217],delay=0,
   status=sent (250 Ok: queued as 154D4D930B)
Aug 18 22:41:33 nova postfix/qmgr: 0E4A93688: removed
```

Легко видеть, что интересующая нас информация разбросана по нескольким строкам. Обратите внимание на то, что идентификатор 0E4A93688 встречается на каждой строке: система Postfix присваивает идентификатор очереди сразу, как только сообщение поступает в почтовую систему, и никогда не изменяет его. Следовательно, при поиске регистрационных записей об истории сообщения в первую очередь следует определить идентификационный номер сообщения в очереди. Узнав его, легко применить команду **grep** ко всем релевантным записям.

Система Postfix делает подробные записи о замеченных ею проблемах. Однако иногда трудно найти важные строки среди тысяч сообщений о нормальном состоянии. В этом случае целесообразно использовать некоторые инструменты, перечисленные в разделе 11.5.

### Просмотр очереди

Еще одно место, где следует искать решение проблем, — почтовая очередь. Как и в системе **sendmail**, команда **mailq** выводит на печать содержание очереди. Его можно использовать, для того чтобы выяснить причину задержки сообщения.

Полезным инструментом является также сценарий **qshape**, который поставляется с последними версиями системы Postfix. Он демонстрирует суммарные статистические показатели о содержании очереди. Его вывод выглядит следующим образом.

```
$ sudo qshape deferred
                  T 5 10 20 40 80 160 320 640 1280 1280+
              TOTAL 78 0  0  0  7  3   3   2  12    2   49
          expn.com 34 0  0  0  0  0   0   0   9    0   25
     chinabank.ph  5 0  0  0  1  1   1   2   0    0    0
 prob-helper.biz   3 0  0  0  0  0   0   0   0    0    3
```

Сценарий **qshape** создает отчет о заданной очереди (в данном случае об очереди отложенных сообщений), упорядоченной по доменам получателя. В столбцах отчета указывается, сколько минут релевантное сообщение провело в очереди. Например, в отчете видно, что 25 сообщений, направленных в домен `expn.com`, провели в очереди более 1280 минут. Все пункты назначения в данном примере вызывают подозрения, что сообщения были посланы автоответчиком в ответ на спам.

Сценарий **qshape** также создает отчет в соответствии с доменами отправителей, если он запущен с флагом **-s**.

### Мягкий рикошет

Если параметр `soft_bounce` установлен равным `yes`, система Postfix посылает сообщения о временных ошибках вместо сообщений о постоянных ошибках, таких как "пользователь неизвестен" или "ретрансляция запрещена". Этот параметр предоставляет отличную возможность для тестирования; он позволяет отслеживать местонахождение сообщения после изменения конфигурации без риска потери законной электронной почты. В этом случае любой отказ в конце концов приводит к повторной попытке доставки.

Не забудьте отключить эту функциональную возможность, когда закончите тестирование, иначе попытки доставки отвергнутой почты будут повторяться снова и снова.

### Тестирование управления доступом

Легче всего проверить ограничения управления доступом, попытавшись послать сообщение с внешнего узла и посмотрев, что произойдет. Это хороший тест, но он не охватывает специальные условия, например, когда почта поступает из домена, в котором у вас нет учетной записи.

Система Postfix 2.1 внедрила расширение протокола SMTP, который называется XCLIENT и имитирует представление из другого места. Эта функциональная возможность по умолчанию отключена, но с помощью следующей строки конфигурации из файла **main.cf** вы можете включить ее для соединений, исходящих из локального узла.

```
smtpd_authorized_xclient_hosts = localhost
```

Сеанс тестирования может выглядеть примерно так.

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 tardis.ee.ethz.ch ESMTP Postfix
XCLIENT NAME=mail.cs.colorado.edu ADDR=192.168.1.1
250 Ok
HELO mail.cs.colorado.edu
250 tardis.ee.ethz.ch
MAIL FROM: <evi@colorado.edu>
250 Ok
RCPT TO: <david@colorado.edu>
554 <david@colorado.edu>: Relay access denied
```

# 20.16. Конфигурация механизма DKIM

Наше описание технологии DKIM не сосредоточено в одной главе. Вступительный материал содержится в главе о системе DNS (раздел 17.8), а также в разделе 20.8. Здесь мы сосредоточимся на деталях, связанных с использованием технологии DKIM в электронной почте как с помощью внешнего инструмента **amavisd**, так и непосредственно в сочетании с системами **sendmail**, Exim и Postfix. Системы **sendmail** и Postfix для реализации технологии DKIM используют почтовые фильтры; система Exim делает это непосредственно. Реализация технологии DKIM в системе Exim является новой (конец 2009 года) и остается не до конца отработанной.

## Технология DKIM: DomainKeys Identified Mail

Технология DKIM — это новая надежда на правильную идентификацию организации отправителя. При широком развертывании она способна резко ограничить возможности злоумышленников для подделки домена отправителя. Тогда почта от вашего банка может быть действительно почтой, посланной вашим банком, или, по крайней мере, классифицироваться как почта, которая не является фишингом или спамом.

Технология DKIM вытесняет прежнюю систему DomainKeys; она использует криптографию с открытым ключом (с ключами, хранящимися в системе DNS), чтобы позволить получателям проверять как источник сообщения, так и его целостность. Это становится важным в глобальном мире, где так много сделок совершается в электронном виде. Система DKIM также не позволяет отправителю отрицать, что это он отправил сообщение; так называемый отказ от авторства. Реализация системы DKIM имеет две стороны: та, которая подписывает исходящую почту, когда она покидает ваш сайт, и та, которая проверяет подписи входящей почты в момент ее поступления.

Первая операция должна выполняться концентратором исходящей почты непосредственно перед тем, как почта покинет ваш сайт (после всех внутренних переписываний и сканирования содержания). Вторая операция, верификация, должна выполняться сразу

после получения сообщения, перед добавлением другого инструмента сканирования или изменения заголовков.

На сайте `sendmail.org` есть много удобных и универсальных инструментов, которые могут использоваться любым программным обеспечением, реализующим технологию DKIM. Первым является мастер ADSP, принимающий имя домена и генерирующий соответствующие текстовые ADSP TXT , которые вы должны добавить в систему DNS, чтобы реализовать технологию DKIM.[21] Второй инструмент — это верификатор, с помощью которого можно проверить свои настройки после завершения приготовлений к работе.

Детали технологии DKIM и ресурсных записей ADSP изложены в разделе 17.8. Они до сих пор остаются текстовыми, хотя в будущем будут преобразованы в тип собственной базы данных о ресурсах DNS. Программы OpenSSL и **amavisd** содержат код для генерации ключей DKIM. Кроме того, различные пакеты почтовых фильтров DKIM также содержат сценарии для генерации ключей.

Настроив конфигурацию вашего почтового транспортного агента так, чтобы он генерировал подписи DKIM, вы можете послать сообщение по адресу `sa-test@sendmail.net`, чтобы проверить, что все работает правильно. Сервер пришлет вам сообщение, в котором укажет, какие функции безопасности он распознал. Если у вас есть учетная запись в системе Gmail, то проверить эти возможности можно, послав сообщение самому себе. Щелкнув на ссылке "show details", вы раскроете подписанное поле. Вы можете также попросить систему Gmail выполнить команду "Show original" в выпадающем меню. Эта команда показывает исходное сообщение со всеми его заголовками, включая подпись DKIM.

## Почтовые фильтры DKIM

Программное обеспечение для реализации технологии DKIM изначально было разработано корпорацией Sendmail, Inc., для того, чтобы использовать его как почтовый фильтр при взаимодействии с транспортным агентом программы **sendmail**. Существует две версии этого кода: оригинальная, DKIM-milter; и альтернативная, OpenDKIM. Оба пакета доступны с исходным кодом на сайте `sourceforge.net` и в скомпилированном виде в разных репозиториях пакетов. Мы проиллюстрируем версию для программы **sendmail** под названием DKIM-milter v2.8.3.

Этот пакет содержит программу **dkim-filter**, почтовый фильтр, создающий и проверяющий подписи, и несколько утилит для отладки и отслеживания эксплуатации. Ниже приведен список этих утилит.

- **dkim-filter** — генерирует и верифицирует подписи DKIM.
- **dkim-genkey** — генерирует пары ключей и требуемые записи DNS.
- **dkim-stats** — создает отчеты на основе статистических данных, собранных утилитой **dkim-filter**.
- **dkim-testkey** — проверяет, имеют ли ключи правильный формат и являются ли они доступными.
- **dkim-testssp** — проверяет запись ADSP (которая ранее называлась SSP).

[21] Не публикуйте записи ADSP, пока система подписи исходящих сообщений на будет налажена и не станет правильно работать, иначе другие сайты не будут принимать вашу электронную почту.

На первом этапе настройки пакета DKIM-milter необходимо создать специальную учетную запись пользователя и группы для своих собственных файлов, связанных с технологией DKIM. Для обеих записей следует использовать имя "dkim". При этом необходимо убедиться, что учетная запись имеет ограниченную оболочку, например **/bin/false**.

Утилита **dkim-genkey** генерирует пару ключей DKIM. Домен, для которого предназначен ключ, задается с помощью флага **-d**, а селектор (т.е. имя ключа) — с помощью флага **-s**. По умолчанию они имеют значения `example.com` и "default". Флаг **-r** ограничивает использование ключа только подписанием электронной почты, а флаг **-t** означает, что вы тестируете систему DKIM. Ключи сохраняются в отдельных файлах: *селектор*`.private` (для закрытого ключа) и *селектор*`.txt` (для текстовой записи DNS, содержащей открытый ключ).

Например, инструкция

```
$ dkim-genkey -r -d example.com -s email
```

генерирует пару ключей в файлах **email.private** и **email.txt**, находящихся в текущем каталоге.

Инсталлируйте закрытый ключ где-нибудь, например, в каталоге **/etc/mail/dkim/keys**. Установите режим каталога **/etc/mail/dkim** равным 600 и примените команду **chown** (рекурсивно) к пользователю и группе dkim. Добавьте запись TXT системы DNS в соответствующий зонный файл, введите порядковый номер зоны и пошлите сигнал вашему серверу имен. После окончания проверки всей системы можно добавить запись ADSP, но раньше этого делать не следует.

Запустите утилиту **dkim-testkey**, чтобы проверить правильность ключей. Эта утилита не создает вывода, если все в порядке, поэтому молчание в данном случае — действительно золото.

На следующем этапе необходимо задать конфигурацию утилиты **dkim-filter** для ее использования в качестве почтового фильтра. Создайте файл **/etc/mail/dkim.conf**, владельцем которого является пользователь и группа dkim. Ниже приведен пример, позаимствованный из статьи T.J. Nelson *Setting up DKIM with Sendmail.*

```
Canonicalization simple
Domain mail.example.com
KeyFile /etc/mail/dkim/keys/email.key.pem
MTA MSA
PidFile /var/run/dkim-filter.pid
Selector email
Socket inet:8891@localhost
SignatureAlgorithm rsa-sha1
Syslog Yes
UserID dkim
X-Header Yes
Mode sv
InternalHosts /etc/mail/dkim-internal-hosts
```

Общий формат записей в файле **dkim.conf** имеет следующий вид.

*параметр значение*

Символ # означает начало комментария. Дистрибутивный пакет содержит хорошо прокомментированный пример конфигурации **dkim-filter.conf.sample**, включающий описание переменных и их значения, заданные по умолчанию. На справочной странице **dkim-filter.conf** можно найти еще более подробное описание.

При настройке конфигурации центрального почтового концентратора строка Domain должна содержать список полностью определенных имен доменов, предназначенных для подписи, разделенных запятыми, или имя файла, в котором содержится этот список.

Параметр KeyFile задает местоположение закрытого ключа, используемого для подписи сообщений. Предполагается, что первая часть имени этого файла представляет собой селектор (в данном случае "email"). Строка MTA содержит список имен транспортных агентов, чья почта всегда должна подписываться, а не верифицироваться. Она аналогична части Name параметра DAEMON_OPTIONS в конфигурации программы **sendmail**.

Спецификация Socket идентифицирует прослушивающий сокет TCP; в данном случае он прослушивает порт 8891 на локальном узле. Параметр SignatureAlgorithm может принимать значения rsa-sha1 или rsa-sha256; второе значение задается по умолчанию, но, поскольку этот алгоритм является относительно новым, его поддерживают не все транспортные агенты. Параметр X-Header указывает, что **dkim-filter** должен добавить строку заголовка в каждое сканируемое сообщение. Параметр Mode может иметь значение s (для подписи), v (для верификации) или sv (для обоих режимов).

Параметр InternalHosts должен ссылаться на файл, содержащий список узлов, исходящая почта которых должна быть подписана. Узлы должны быть перечислены с полностью определенными именами.

Несколько параметров конфигурации позволяют облегчить отладку. К ним относятся параметры MilterDebug, LogWhy и SyslogSuccess. Некоторые из них генерируют так называемую регистрационную информацию, которую после отладки следует отключить.

Другие параметры указывают, что делать с сообщениями, подписи которых не могут быть верифицированы: On-Default, On-BadSignature, On-DNSError, On-Security, On-InternalError и On-NoSignature. Каждый из них принимает значения reject, tempfail, accept и discard.

Система допускает гибкую конфигурацию и имеет много параметров (более 80), которые описаны на справочной странице **dkim-filter.conf**.

## Конфигурация DKIM в программе amavisd-new

Для использования технологии DKIM в программе **amavisd-new** необходимо иметь модуль Mail::DKIM версии 0.33 или выше, написанный на языке Perl. Если такого модуля нет, загрузите его с сайта CPAN. Вы должны сгенерировать пару ключей, включить режим верификации и подписи DKIM, указать файл, содержащий закрытый ключ, и задать некоторые параметры подписи.

Программа **amavisd** может генерировать ключи для системы DKIM самостоятельно.

```
$ amavisd genrsa /var/db/dkim/example.com-email.key.pem
```

Еще одним аргументом программы **amavisd** является имя файла, в котором хранятся ключи. Поскольку он содержит и закрытый, и открытый ключи, лучше всего проверить разрешения и убедиться, что этот файл невозможно прочитать извне и несанкционированный доступ к нему закрыт.

Следующий фрагмент конфигурации был немного изменен по сравнению с разделом документации **amavisd-new** под названием "Bits and pieces." В этом примере мы предполагаем, что домен вашего сайта называется example.com и что вы используете селектор "email" для своих ключей.

```
$enable_dkim_verification = 1;
$enable_dkim_signing = 1;
```

```
dkim_key('example.com', 'email', '/var/db/dkim/example.com-email.key.pem');
@dkim_signature_options_bysender_maps = (
   { '.' => { ttl => 21*24*3600, c => 'relaxed/simple' } } );
@mynetworks = qw(127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16);
```

Строка с параметрами подписи DKIM позволяет заменить дескрипторы, заданные по умолчанию и хранящиеся в ассоциативном массиве. Подписи могут иметь 13 разных значений дескрипторов. Многие дескрипторы, такие как дескриптор d, который задает домен отправителя, устанавливаются автоматически. Важным дескриптором является s, который задает селектор, предназначенный для использования при запросе у системы DNS открытого ключа для верификации подписи. В данном примере мы заменили значение параметра ttl, задающего срок действия ключа, с 30 дней на 21. Некоторые дескрипторы становятся важными при ретрансляции почты (например, через списки рассылки), если вы хотите сохранить возможность подписания таких сообщений.

Присвоив булевы значения ассоциативному массиву signed_header_fields, можно задать поля заголовков, которые должны быть включены в подпись. Например, инструкции

```
$signed_header_fields{'received'} = 0;
$signed_header_fields{'sender'} = 1;
$signed_header_fields{'to'} = 1;
$signed_header_fields{'cc'} = 1;
```

исключают заголовок Received (Получено), но включают Sender (Отправитель), To (Кому) и Cc (Копии). Значения, заданные по умолчанию, вероятно, подойдут для большинства сайтов.

Для тестирования конфигурации можно использовать инструкции **amavisd showkeys** и **amavisd testkeys**. Команда **showkeys** выводит на печать открытый ключ, который необходимо добавить в зонный файл для домена example.com. (Не забудьте изменить порядковый номер зоны и отправить сигнал вашему серверу имен!) Команда **testkeys** проверяет как процесс подписания, так и тот факт, что ваш ключ был опубликован в системе DNS.

## Технология DKIM в системе sendmail

Используя почтовые фильтры (для оперативной фильтрации, описанной в разделе 20.6) или дублирующий сервер в сочетании с программой **amavisd**, можно реализовать технологию DKIM для программы **sendmail**. Здесь мы опишем использование почтовых фильтров DKIM для подписания и верификации сообщений.

Примитивы masquerade_as и genericstable конфигурации программы **sendmail** переписывают заголовки, поэтому эти примитивы должны быть реализованы до добавления любых подписей DKIM, в противном случае подпись станет недействительной.

Для поддержки технологии DKIM системный администратор должен скомпилировать программу **sendmail** с поддержкой почтовых фильтров, иметь доступ к библиотекам OpenSSL и Berkeley DB и инсталлировать пакет DKIM-milter (который можно найти на сайте sourceforge.net). Библиотека OpenSSL используется для генерирования ключей. Следует добавить ее в свою зону DNS, как указано в разделе 17.8, или использовать утилиту **dkim-genkey**, как показано выше.

В данном примере наш домен называется example.com. Мы выбрали селектор "email" и сохранили наш закрытый ключ в файле **email.private**, а открытый ключ — в файле **email.key.pem**. Проверить ключи можно с помощью утилиты **dkim-testkey**.

```
$ dkim-testkey -d example.com -k /var/db/dkim/email.key.pem -s ma
```

Если все в порядке, утилита **dkim-testkey** не выведет никаких результатов.

Добавьте в конфигурационный файл **sendmail.mc** программы **sendmail** строку

```
INPUT_MAIL_FILTER(`dkim-filter', `S=inet:8891@localhost')
```

и повторите сборку файла **sendmail.cf** (с помощью инструкций **./Build sendmail.cf**; **sudo make install-cf**).

Затем заново запустите программу **sendmail** и утилиту **dkim-filter**.

## Технология DKIM в системе Exim

Выпуск Exim 4.70 (конец 2009 года) добавил естественную поддержку технологии DKIM и перестал поддерживать технологию DomainKeys компании Yahoo!. Ожидается, что набор его функциональных возможностей будет расширяться по мере исправления ошибок и приобретения опыта работы с технологией DKIM.

Поддержка технологии DKIM в системе Exim включена по умолчанию. Для того чтобы ее отключить, установите значение `DISABLE_DKIM=yes` в файле **Local/Makefile**, а затем соберите и заново инсталлируйте пакет.

Реализация системы Exim подписывает исходящие сообщения в конфигурации транспортного протокола SMTP и верифицирует входящие сообщения с помощью нового списка управления доступом `acl_smtp_dkim`. Концентраторы могут отключать верификацию подписей на сообщениях, которые они ретранслируют на локальные узлы, установив для этих сообщений параметр `dkim_disable_verify`.

### Подписание исходящих сообщений

Первое действие при реализации технологии DKIM — генерация криптографических ключей с помощью библиотеки OpenSSL (см. раздел 17.8). Для транспортного протокола SMTP необходимо определить несколько новых параметров системы Exim. Эти параметры могут включать переменные, которые должны раскрываться при вызове транспорта. Список параметров подписей, связанных с технологией DKIM, приведен в табл. 20.26.

**Таблица 20.26. Параметры подписей DKIM в системе Exim**

| Параметр | Тип | Обязательность | Описание |
|---|---|---|---|
| `dkim_domain` | строка | да | Подписываемый домен |
| `dkim_selector` | строка | да | Селектор ключа (имя) |
| `dkim_private_key` | строка | да | Закрытый ключ или файл, который его содержит |
| `dkim_canon` | строка | нет | Метод канонизации: `simple` или `relaxed` |
| `dkim_strict` | строка | нет | Если true, то из-за ошибок в подписи сообщение возвращается в очередь |
| `dkim_sign_headers` | строка | нет | Заголовок для включения в подпись |

Первые три параметра являются обязательными и должны быть заданы в конфигурации; остальные параметры носят факультативный характер. По умолчанию для канонизации (`dkim_canon`) используется метод `relaxed`, сообщения, вызывающие ошибки в подписях, посылаются без подписей (`dkim_strict`), а система Exim для подписи использует список заголовков RFC4871 (`dkim_sign_headers`). Обязательные параметры не требуют дополнительных объяснений, но в сложном сайте, содержащем несколько реальных и виртуальных доменов, их настройка требует определенного опыта.

### Верификация входящих подписанных сообщений

Входящие почтовые сообщения, подписанные по технологии DKIM, проверяются с помощью списка управления доступом `acl_smtp_dkim`. Этот список вызывает каждую подпись и возвращает один из нескольких кодов.

- `none` — сообщение не подписано.
- `invalid` — подпись невозможно верифицировать (ключ недоступен или некорректен).
- `fail` — подпись не прошла верификацию заголовка, тела или и того, и другого.
- `pass` — подпись является корректной.

Статус сообщения записывается в переменную `$dkim_verify_status`, а информация об ошибках — в переменную `$dkim_verify_reason`. Существует много других дескрипторов вида `$dkim_`*переменная*, предоставляющих доступ к разным полям подписи и позволяющих реализовать специальную политику (например, маркировку сообщений от сайта `gmail.com`, которые не имеют подписей, или отказ от приема сообщений от сайта `paypal.com`, которые не поддаются верификации).

### Законченный пример

Следующий пример позаимствован из настроек системы DKIM, предложенных Филом Пенноком.[22] Она содержит определения параметра `acl_process_dkim` для верификации подписей, а также маршрутизатора и транспорта (`dnslookup_signed` и `remote_dksign` соответственно) для создания реальных подписей.

Эта конфигурация позволяет подписывать несколько доменов и использовать несколько ключей, чтобы осуществлять их смену. Этот файл хранится в формате CDB и отображает ключи вида "example.org" в значения вида "d200912."

Селектор ключа называется *dyyyymm*, где *d* — это просто первая буква слова "date" (дата), *yyyy* — это год, в котором был сгенерирован ключ, а *mm* — месяц, в котором он был сгенерирован. Файлы с ключами называются **`rsa.private.`***`селектор.домен`* и хранятся в каталоге ключей, определенном макросом `DKKEY_DIR`. Убедитесь, что этот каталог доступен для системы **`exim`** и не доступен для всего остального мира.

```
######## macros to define the directories for databases and keys
CONFIG_DIR = path_to_config_dir
DKKEY_DIR = path_to_key_dir

######## main section: define the domains to sign and required DKIM acl
domainlist dksign_domains = cdb;CONFIG_DIR/dk.selector.cdb
acl_smtp_dkim = acl_process_dkim

######## ACL section: verify signature on incoming mail, add a header
acl_process_dkim:
   warn !dkim_status = none
      add_header = :at_start:X-DKIM-Report: $dkim_verify_status \
        ${if !eq{$dkim_verify_status}{pass}{$dkim_verify_reason }{}} \
        (Signer=$dkim_cur_signer) (Testing=$dkim_key_testing)

######## Router section: put just before "dnslookup" router, sign nonlocal
dnslookup_signed:
    driver = dnslookup
```

[22] Фил является активным участником группы рассылки для пользователей системы Exim.

```
 domains = !+local_domains
 transport = remote_dksign
 condition = ${if match_domain{$sender_address_domain} \
     {+dksign_domains}}
no_verify

######## Transport section: does the actual signing
remote_dksign:
   driver = smtp
   dkim_domain = $sender_address_domain
   dkim_selector = ${lookup {$sender_address_domain} \
   cdb{CONFIG_DIR/dk.selector.cdb} {$value}fail}
   dkim_private_key = DKKEY_DIR/rsa.private.$dkim_selector.$dkim_domain
   dkim_strict = 1
```

В результате применения этих фрагментов исходящие сообщения будут подписываться, входящие сообщения, имеющие подпись, будут верифицироваться, а затем к ним будет добавляться заголовок отчета системы DKIM. Ниже приведен пример такого заголовка.

```
X-DKIM-Report: pass (Signer=gmail.com) (Testing=0)
```

Если вы собираетесь отказываться от приема или каким-то образом обрабатывать сообщения, подписи которых не были верифицированы, эти правила следует дополнить.

Строка `no_verify` в строке маршрутизатора относится не к верификации подписей DKIM, а к верификации адреса получателя; в данном маршрутизаторе этот параметр отключен, зато в следующем маршрутизаторе `dnslookup` он включен. Нет никакого смысла делать это дважды.

## Технология DKIM в системе Postfix

Технология DKIM реализована в системе Postfix с помощью пакета DKIM-milter, описанного выше. Сгенерируйте свою пару ключей и проверьте их с помощью утилиты **dkim-testkey**; скомпилируйте файл **dkim-filter.conf** на основании примера из дистрибутивного пакета, а затем укажите системе Postfix, чтобы она использовала утилиту **dkim-filter**. В файле **main.cf**, после всех параметров почтовых фильтров, добавьте следующие строки.

```
smtpd_milters = inet:localhost:8891
non_smtpd_milters = inet:localhost:8891
milter_protocol = 2
milter_default_action = accept
```

Теперь необходимо лишь запустить утилиту **dkim-filter** и повторно запустить систему **postfix**.

# 20.17. Интегрированные почтовые системы

В настоящее время существует много интегрированных почтовых систем: от свободно распространяемых систем с открытым кодом до дорогих коммерческих продуктов. Все делают намного больше, чем просто принимают и отправляют электронную почту. Обычно они предусматривают следующие функции для организации групп и конференций.

- Управление адресной книгой и списком контактов
- Управление календарем и расписанием

- Списки рассылки и доски объявлений
- Мгновенная передача сообщений
- Шифрование SSL/TLS
- Архивирование и автоматическое резервирование
- Поддержка мобильных устройств (BlackBerry, iPhone, etc.)

Большинство этих мегапакетов содержит описание конфигурации графического пользовательского интерфейса, которое в некоторой степени может заменить собой чтение этой главы. (Возможно, этот раздел следовало бы поместить не в конец, а в начало главы.) Многие системы предназначены для вытеснения программы Microsoft Exchange.

Некоторые программы достойны отдельного упоминания.

- Citadel (`citadel.org`) — это открытый пакет для электронной почты и программного обеспечения совместной работы (groupware), предусматривающий контракт на поддержку.
- Zimbra (`zimbra.com`) занимает промежуточное положение между программами с открытым кодом и коммерческими системами. Существует ее полнофункциональная коммерческая версия и упрощенная версия, открытая и бесплатная.
- Kerio MailServer (`kerio.com`) — это коммерческая система, которая, подобно системе Zimbra, лицензируется для каждого пользователя. Для крупных организаций эта система может стоить дорого.
- Communigate Pro (`communigate.com`) внедряет поддержку аудио- и видеофайлов в обычный пакет для электронной почты и программного обеспечения совместной работы и предусматривает либо традиционное безлимитное лицензирование, либо лицензирование для каждого пользователя.

Существуют также аппаратные устройства, предназначенные для реализации функций электронной почты, которые обычно являются частью систем FreeBSD UNIX или Linux. Заслуживают также внимания устройства IronPort Series компании Cisco, а также модели от компаний Sophos и Clearswift.

Эти устройства обычно выполняют фильтрацию вирусов и спама, а затем передают сообщения программе Microsoft Exchange для доставки.

## 20.18. Рекомендуемая литература

Для того чтобы не смешивать все ссылки в один список, мы упорядочили их по транспортным агентам и темам.

### Общие вопросы по борьбе со спамом


- Clayton R. Good Practice for Combating Unsolicited Bulk Email. — RIPE/Demon Internet, 2000, `ripe.net/ripe/docs/ripe-206.html`.

  Этот документ предназначен для интернет-провайдеров. Он содержит много информации о правилах и полезные описания технических вопросов.
- Field J. MailScanner: A User Guide and Training Manual. — University of Southampton Department of Electronics, 2007.
- McDonald A. SpamAssassin: A Practical Guide to Configuration, Customization, and Integration. — Packt Publishing, 2004.

- Schwartz A. SpamAssassin. — Sebastopol, CA: O'Reilly Media, 2004.
- Wolfe P., Scott C., Erwin M. The Anti-Spam Tool Kit. — Emeryville, CA: Osborne, 2004.

## Литература по программе sendmail

Costales B., Assmann C., Jansen G., Shapiro G. *Sendmail, 4th Edition.* —Sebastopol, CA: O'Reilly Media, 2007.

Эта книга представляет собой внушительный том о конфигурации программы **sendmail** размером более 1300 страниц. Она содержит руководство системного администратора, а также полный список библиографических ссылок. Кроме того, существует электронный вариант этой книги. В коллектив авторов входят два ключевых разработчика программы **sendmail** (Claus и Greg), что является гарантией технической корректности и авторитетности книги. В разделе *Sendmail Installation and Operation Guide* изложены инструкции по инсталляции и хорошее описание файла конфигурации. Файл с этим разделом можно найти в подкаталоге **doc/op** дистрибутивного пакета **sendmail**. Этот документ является достаточно полным, и в сочетании с файлом **README** из каталога **cf** он дает конкретное представление о системе **sendmail**.

Сайты `sendmail.org`, `sendmail.org/~ca` и `sendmail.org/~gshapiro` содержат документы, ответы на вопросы и справочные материалы по программе **sendmail**.

## Литература о системе Exim

- Hazel P. *The Exim SMTP Mail Server: Official Guide for Release 4, 2nd Edition.* — Cambridge, UK: User Interface Technologies, Ltd., 2007.
- Hazel P. *Exim: The Mail Transfer Agent.* — Sebastapol, CA: O'Reilly Media, 2001.
- Meers J. *Getting started with EXIM.* — `exim-new-users.co.uk`, 2007.

  Спецификация системы Exim представляет собой определяющий документ о ее конфигурации. Он является исчерпывающим и обновляется с появлением каждой новой версии. Текстовая версия включена в файл **doc/spec.txt** из дистрибутивного пакета, а соответствующий файл PDF доступен на сайте `exim.org`. Кроме того, на этом веб-сайте можно найти несколько справочных документов.

## Литература о системе

- Blum R. *Postfix.* — Sams Publishing, 2001.
- Dent K.D. *Postfix: The Definitive Guide.* — Sebastopol, CA: O'Reilly Media, 2003.
- Hildebrandt R., Koetter P. *The Book of Postfix: State of the Art Message Transport.* — San Francisco, CA: No Starch Press, 2005.

  Эта книга посвящает читателя во все тонкости конфигурации системы Postfix, даже в сложных средах. Авторы являются активными членами сообщества пользователей системы Postfix и регулярно участвуют в почтовых рассылках группы пользователей Postfix. К сожалению, эта книга не переиздается, но ее можно приобрести в букинистических магазинах.
- Страница `postfix.org/SOHO_README.html` представляет собой справочное руководство по использованию системы Postfix дома или в офисах небольших компаний.

## Документы RFC

Текущими версиями документов RFC821 и RFC822 являются документы RFC5321 и RFC5322. В них определен протокол SMTP и форматы сообщений и адресов для электронной почты в Интернете.

Документы RFC5335 и RFC5336 посвящены вопросам интернационализации адресов электронной почты.

Существует около 90 документов RFC, связанных с электронной почтой. Их слишком много, чтобы перечислять в этой книге. Все эти документы можно найти на сайте `rfc-editor.org` с помощью поисковой машины.

# 20.19. Упражнения

20.1. Кратко объясните разницу между пользовательским агентом (MUA), агентом доставки (DA) и агентом доступа (AA). Затем объясните разницу между почтовым транспортным агентом (MTA) и почтовым агентом передачи (MSA).

20.2. Проверьте почтовую очередь на вашем локальном почтовом сервере? Есть ли там сообщения без управляющих файлов или управляющие файлы без сообщений? Какое самое старое сообщение вы нашли в очереди? (Это действие требует прав суперпользователя.)

20.3. Объясните, что такое запись MX. Почему записи MX важны для доставки почты? Приведите пример, в котором неверно сконфигурированная запись MX может привести к отказу от доставки почты.

20.4. Определите схему почтовой службы на вашем сайте и нарисуйте диаграмму в стиле рис. 20.2. Где находится входящая почта, прошедшая сканирование на спам и вирусы? Что скажете об исходящей почте?

20.5. Сравните использование файла **`/etc/mail/aliases`** с использованием сервера LDAP или базы данных MySQL для хранения почтовых псевдонимов. Какие преимущества и недостатки имеет каждый из этих механизмов?

20.6. ★ Дайте краткое описание следующих заголовков сообщений. Какой путь пройдет электронная почта? Кому она адресована и кому была доставлена? Как долго она находилась в пути?

```
Delivered-To: sailingevi@gmail.com
Received: by 10.231.143.81 with SMTP id t17cs175323ibu;
  Mon, 28 Dec 2009 20:15:20 -0800 (PST)
Received: by 10.231.157.131 with SMTP id
  b3mr2134004ibx.19.1262060119841;
  Mon, 28 Dec 2009 20:15:19 -0800 (PST)
Return-Path: <garth@grsweb.us>
Received: from mail-relay.atrust.com (mail-relay.atrust.com
  [63.173.189.2]) by mx.google.com with ESMTP id
  12si19092249iwn.27.2009.12.28.20.15.19;
  Mon, 28 Dec 2009 20:15:19 -0800 (PST)
Received-SPF: neutral (google.com: 63.173.189.2 is neither permitted nor
  denied by best guess record for domain of garth@grsweb.us) clientip=
  63.173.189.2;
Authentication-Results: mx.google.com; spf=neutral (google.com:
  63.173.189.2 is neither permitted nor denied by best guess record for
  domain of garth@grsweb.us) smtp.mail=garth@grsweb.us
```

```
Received: from mout.perfora.net (mout.perfora.net [74.208.4.194]) by
  mail-relay.atrust.com (8.12.11/8.12.11) with ESMTP id nBT4FI9r017821
  for <evi@atrust.com>; Mon, 28 Dec 2009 21:15:19 -0700
Received: from grsweb.us (wolverine.dreamhost.com [75.119.201.185]) by
  mrelay.perfora.net (node=mrus1) with ESMTP (Nemesis) id 0MaORD-
  1NgKS52KT9-00LeuN; Mon, 28 Dec 2009 23:15:17 -0500
Date: Mon, 28 Dec 2009 20:15:13 -0800
From: UNIX and Linux System Administration Handbook
  <garth@grsweb.us>
Reply-To: garth@grsweb.us
To: evi@atrust.com
Cc: garth@grsweb.us
Message-Id: <4b398251b11ab_e92383578b2d9b036f@wolverine.tmail>
Subject: New comments on Printing
Mime-Version: 1.0
Content-Type: text/html; charset=utf-8
X-Provags-ID:
  V01U2FsdGVkX18pouiYXif/bVfh+D9wFXMr24TahAzDNZqM+jA04iLR7S4
  olDXRpXlrbQMblNoZf5jO6edc+WIGC8Fi4hd5Ak15vBARASOFQYxNJWea9
  8SyQg==
X-Spam-Status: No, hits=-99.3 required=4.0 tests=BAYES_30,HTML_20_30,
  MIME_HTML_ONLY,USER_IN_WHITELIST version=2.55
X-Spam-Level:
X-Spam-Checker-Version: SpamAssassin 2.55 (1.174.2.19-2003-05-19-exp)
```

20.7. ★ Какие последствия возникают из-за включения в черный список сайта `spamhaus.org` или аналогичной службы? Что следует делать, если выяснилось, что ваш сайт попал в черный список? Сначала опишите действия, которые позволяют не попасть в черные списки.

20.8. ★ Если ваш сайт позволяет использовать программу **`procmail`** и вы имеете разрешение от локального системного администратора группы, настройте ваш файл конфигурации программы **`procmail`** так, чтобы проиллюстрировать, как программа **`procmail`** может разрушить систему безопасности.

20.9. ★★ Исследуйте текущую конфигурацию транспортного агента на вашем сайте. Какие специальные функциональные возможности транспортного агента в нем используются? Есть ли проблемы в вашей конфигурации? Как улучшить эту конфигурацию?

20.10. ★★ Найдите спам в вашем почтовом ящике и проверьте заголовки. Укажите признаки подделанной почты. Затем запустите какую-нибудь программу, описанную в главе, например SpamAssassin, и проанализируйте результаты ее работы. Как вы распознали подделанные заголовки? Опишите спам и представьте ваши выводы об отправителе, корректности перечисленных узлов и другие факты, свидетельствующие о нарушениях.

## Упражнения, связанные с программой sendmail

20.11. Опишите утилиту **`smrsh`** и способ ее использования, вместо утилиты **`/bin/sh`**? Если утилита **`smrsh`** применяется на вашем сайте, укажите, какие программы разрешается запускать в качестве обработчика очереди? Являются ли какие-нибудь из этих программ небезопасными?

20.12. Напишите небольшой файл **/etc/mail/aliases**, демонстрирующий три разных вида псевдонимов. Кратко опишите, что делает каждая строка в этом файле и чем она может быть полезной.

20.13. ★ Перечислите префиксы для файлов в каталоге почтовой очереди и объясните, что означает каждый из них. Почему следует удалять одни файлы из очереди и почему большой ошибкой было бы удалять другие? Как могут некоторые префиксы использоваться для отладки ошибок в конфигурации программы **sendmail**?

20.14. ★ Объясните назначение каждого макроса препроцессора **m4**. Если макрос вставляет файл, кратко опишите, какое содержимое должен иметь такой файл.

а) `VERSIONID`

б) `OSTYPE`

в) `DOMAIN`

г) `MAILER`

д) `FEATURE`

20.15. ★ Объясните, как задать конфигурацию сервера **sendmail**, чтобы он принимал электронную почту как для вашего домена, так и для виртуального домена. Разрешите виртуальному домену ретранслировать почту на почтовые ящики, находящиеся за пределами сайта.

## Упражнения, связанные с системой Exim

20.16. Возьмите список управления доступом для команды SMTP RCPT, продемонстрированный в разделе 20.14, и измените значение параметра `deny`, заданное по умолчанию, на противоположное, разрешив тем самым использование одинаковых адресов.

20.17. ★Версии 4.70 и более поздние не поддерживают технологию DomainKeys, отдавая предпочтение технологии DKIM. Упростите пример настройки системы DKIM, продемонстрированный в разделе 20.16, так чтобы она поддерживала только один домен и один ключ подписи. Затем добавьте несколько правил, например регистрацию неподписанной почты, поступающей от сайтов Gmail или Yahoo!, или отказ от приема писем, не прошедших верификацию, если они поступили от системы PayPal или вашего банка.

20.18. ★ Объясните, как настроить конфигурацию сервера Exim, чтобы он принимал почту как для вашего собственного домена, так и для виртуального домена. Разрешите виртуальному домену ретранслировать почту на почтовые ящики, находящиеся за пределами сайта.

20.19. ★ Проанализируйте фрагменты конфигурации в документе **spec.txt** из дистрибутивного пакета Exim и попробуйте включить некоторые из них в вашу конфигурацию. Включите подробную регистрацию для каждого действия, которое вы предпринимаете, и проверьте регистрационные файлы, чтобы убедиться, что вы получили желаемый результат.

★★★★★

## Упражнения, связанные с системой Postfix

20.20. Попробуйте настроить "нулевого клиента", т.е. почтовую систему, которая только посылает почту и не может получать ее. Убедитесь, что порт 25 закрыт.

20.21. Задайте конфигурацию системы Postfix так, чтобы она аутентифицировала ваш сайт для вашего провайдера или сервера компании (даже Gmail!); используйте следующие параметры.

```
smtp_sender_dependent_authentication
sender_dependent_relayhost_maps
smtp_sasl_auth_enable
smtp_sasl_password_maps
```

20.22. Как вы думаете, почему система Postfix поддерживает так много типов отображений?

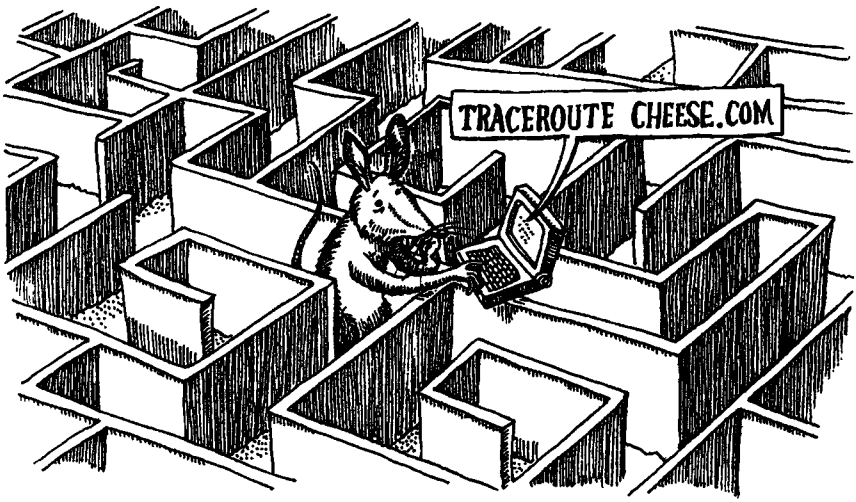
20.23. Для чего используется отображение `pcre`? Может ли быть полезным для почтовых систем подстановка значений? Необходимо ли использовать команду **`postmap`** для компиляции отображений `pcre`?

20.24. Найдите описание параметра `recipient_delimiter` в документации (справочная страница **`postconf`**). Для чего он может использоваться?

20.25. ★ Объясните, как настроить конфигурацию системы Postfix, чтобы она принимала почту как для вашего собственного домена, так и для виртуального домена. Разрешите виртуальному домену ретранслировать почту на почтовые ящики, находящиеся за пределами сайта.

# Глава 21

# Управление сетями




Поскольку при объединении компьютеров в сети число связей между ними увеличивается, в сетях, как правило, усугубляются все имеющиеся проблемы. Как гласит народная мудрость, "работа в сети — это когда вы не можете выполнить свое задание из-за отказа компьютера, о котором никогда не слышали".

Управление сетями — это, прежде всего, искусство и наука поддержания их в работоспособном состоянии. Сюда обычно входят следующие задачи:

- поиск неисправностей в сетях, шлюзах и важных серверах;
- разработка схем уведомления администратора о наличии проблем;
- общий мониторинг сети с целью распределения нагрузки в ней и планирования ее дальнейшего расширения;
- документирование и визуализация работы сети;
- управление сетевыми устройствами с центральной станции.

В отдельном сегменте сети вряд ли стоит внедрять формальные процедуры управления сетью. Достаточно провести тщательное тестирование сети после ее прокладки и время от времени проверять уровень нагрузки. Сломается — почините.

По мере роста сети процедуры управления должны становиться все более автоматизированными. В сети, где несколько подсетей объединяются посредством коммутаторов или маршрутизаторов, решение административных задач можно автоматизировать с помощью сценариев и простейших программ. Если используются глобальные или сложные локальные сети, подумайте об инсталляции станций управления сетью со специальным программным обеспечением.

В некоторых случаях усложнение системы управления сетью диктуется необходимостью обеспечения надежности. Часто бывает так, что возникновение проблемы в сети приводит к полной остановке работы. Если подобные задержки недопустимы, лучше приобрести и установить высококлассную корпоративную систему управления сетью.

К сожалению, даже самая лучшая система не поможет предупредить все проблемы. Важно иметь хорошо документированную схему сети и высококвалифицированный обслуживающий персонал, способный справляться с неизбежными сбоями.

# 21.1. Поиск неисправностей в сетях

Существует несколько хороших инструментов, позволяющих искать неисправности в сети на уровне TCP/IP. Большинство из них выдает низкоуровневую информацию, поэтому для того, чтобы пользоваться ими, нужно хорошо понимать принципы работы протоколов TCP/IP и маршрутизации.

С другой стороны, источником проблем в сети могут служить и ошибки в работе таких высокоуровневых протоколов, как DNS, NFS или HTTP. Прежде чем приступать к чтению этой главы, прочтите главы 14 и 15.

В этом разделе мы рассмотрим общую стратегию поиска неисправностей, после чего перейдем к знакомству с основными инструментами этого поиска: утилитами **ping**, **traceroute**, **netstat**, **tcpdump** и Wireshark. Команда **arp**, которая также предназначена для поиска неисправностей в сети, здесь не описывается; о ней рассказывалось разделе 14.6.

Когда в сети возникает неисправность, появляется желание побыстрее все устранить. Не торопитесь! Важно обдумать подход к решению проблемы, избегая необдуманных действий. Наибольшая ошибка заключается во внесении плохо спланированных изменений в несправную сеть.

Прежде чем "набрасываться" на собственную сеть, примите во внимание следующие принципы.

- Вносите пошаговые изменения в конфигурацию сети, тщательно проверяя результаты, чтобы убедиться в совпадении полученного эффекта с ожидаемым. Изменения, которые не дали нужного результата, должны отменяться.
- Документируйте возникшую ситуацию и все вносимые в нее изменения.
- Проблемы могут носить временный характер, поэтому начните с поиска релевантной информации с помощью утилит, таких как **sar** и **nmon**.
- Начните с "края" системы или сети и продвигайтесь по ключевым ее компонентам, пока не доберетесь до источника неисправности. Например, начните с исследования сетевой конфигурации клиентского компьютера, затем проверьте физическое соединение клиента с сетью, сетевое оборудование и, наконец, сетевые аппаратные средства сервера и его программную конфигурацию.
- Будьте в курсе событий. Сетевые проблемы оказывают влияние на многих людей: пользователей, провайдеров, системных администраторов, инженеров по телекоммуникациям, сетевых администраторов и т.д. Постоянный контакт с другими специалистами позволит вам не мешать друг другу при решении проблемы.
- Работайте в команде. Многолетний опыт показывает, что люди совершают меньше глупых ошибок, если им оказывают поддержку.
- Помните о многоуровневой структуре сетевых средств. Начните с верхнего или нижнего уровня и последовательно продвигайтесь по стеку протоколов, проверяя их работу.

Последнее замечание заслуживает более подробного рассмотрения. Как указывалось в разделе 14.2, в семействе протоколов TCP/IP определяются уровни абстракции, на ко-

торых функционируют различные компоненты сети. Например, протокол HTTP зависит от протокола TCP, который, в свою очередь, основан на протоколе IP, а последний взаимодействует с протоколом Ethernet, работоспособность которого зависит от целостности сетевого кабеля. Можно существенно уменьшить время поиска неисправности, если предварительно понять, средства какого уровня ведут себя неправильно.

Проходя стек протоколов уровень за уровнем (вверх или вниз), проверяйте следующее.

- Есть ли физическое соединение и светится ли индикатор связи?
- Правильно ли сконфигурирован сетевой интерфейс?
- Отображаются ли в таблицах маршрутизации адреса других компьютеров?
- Есть ли брандмауэр на вашем локальном компьютере?
- Если брандмауэр есть, проходят ли через него ICMP-пакеты утилиты **ping** и ответы?
- Находит ли команда **ping** узел **localhost** (127.0.0.1)?
- Находит ли команда **ping** другие компьютеры локальной сети по IP-адресам?
- Правильно ли сконфигурирована служба DNS?[1]
- Находит ли команда **ping** другие компьютеры локальной сети по именам?
- Находит ли команда **ping** компьютеры в другой сети?
- Работают ли высокоуровневые сетевые утилиты, такие как **telnet** или **ssh**?
- Вы правда проверили брандмауэры?

Определив, на каком уровне возникает проблема, не спешите с выводами. Подумайте сначала, как отразятся последующие проверки и вносимые изменения на других сетевых службах и компьютерах.

## 21.2. Команда ping: проверка доступности компьютера

Команда **ping** удивительно проста, но во многих случаях ее оказывается вполне достаточно. Она посылает ICMP-пакет ECHO_REQUEST конкретному компьютеру и ждет ответа.

Команду **ping** можно применять для проверки работоспособности отдельных компьютеров и сегментов сети. В обработке ее запроса участвуют таблицы маршрутизации, физические компоненты сетей и сетевые шлюзы, поэтому для достижения успешного результата сеть должна быть в более или менее рабочем состоянии. Если команда не работает, можно быть совершенно уверенным в том, что более сложные средства тем более не функционируют. Однако это правило неприменимо в сетях, где брандмауэры блокируют эхо-запросы ICMP. Убедитесь в том, что брандмауэр не препятствует работе коман-

[1] Если компьютер загружается очень медленно, зависает при загрузке или при попытке установить связь посредством утилиты **telnet**, вероятнее всего, служба DNS функционирует неверно. В системах Solaris и Linux используется очень сложный подход к разрешению имен, конфигурация которого задана в файле **/etc/nsswitch.conf**. В других системах особый интерес представляет демон кеширования службы имен (**nscd**). Если он дает сбой или имеет неправильную конфигурацию, это сказывается на поиске имен. По мере развития протокола IPv6 выяснилось, что многие маршрутизаторы DSL выполняют функции службы по перенаправлению DNS, которые просто игнорируют запросы на записи DNS для IPv6 (AAAA). Эта "оптимизация" вызывает длинные простои при выполнении запросов на распознавание имен. Для проверки правильности работы вашего распознавателя и сервера имен используйте команду **getend** (например, **getend hosts google.com**).

ды **ping**, прежде чем подозревать, что зондируемый компьютер игнорирует эту команду. В конце концов, отключите на короткое время брандмауэр для проверки работоспособности сети.

Если сеть находится в плохом состоянии, вероятно, система DNS работает неправильно. Упростите ситуацию, используя числовые IP-адреса при выполнении утилиты **ping**, и примените параметр **-n**, чтобы предотвратить обратный поиск IP-адресов, поскольку при этом также генерируются DNS-запросы.

Если утилита **ping** используется для проверки выхода в Интернет, необходимо проверить брандмауэр. Некоторые хорошо известные сайты отвечают на пакеты **ping**, а другие — нет. Например, оказалось, что сайт `google.com` отвечает на такие запросы.

Большинство версий команды работает в бесконечном цикле, если не задан аргумент "число пакетов". В системе Solaris команда **ping -s** обеспечивает расширенный вывод, а в других системах это происходит по умолчанию. Для того чтобы прервать работу команды, нажмите <Ctrl+C>.

Приведем пример.

```
linux$ ping beast
PING beast (10.1.1.46): 56 data bytes
64 bytes from beast (10.1.1.46): icmp_seq=0 ttl=54 time=48.3 ms
64 bytes from beast (10.1.1.46): icmp_seq=1 ttl=54 time=46.4 ms
64 bytes from beast (10.1.1.46): icmp_seq=2 ttl=54 time=88.7 ms
^C
--- beast ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss, time 2026ms
rtt min/avg/max/stddev = 46.490/61.202/88.731/19.481 ms
```

Информация о компьютере **beast** включает его IP-адрес, порядковый номер ответного ICMP-пакета и полное время прохождения пакета. Полученные результаты свидетельствуют о том, что компьютер **beast** работает и подключен к сети.

Порядковый номер ICMP-пакета — особенно ценный элемент информации. Нарушение последовательности свидетельствует об удалении пакетов. Обычно они сопровождаются сообщениями о каждом пропущенном пакете.

Несмотря на то что протокол IP не гарантирует доставку пакетов, пакеты будут пропадать только в том случае, если сеть перегружена. Потерю пакетов следует обязательно выявлять, потому что этот факт иногда скрывается протоколами более высокого уровня. Может показаться, что сеть функционирует нормально, хотя на самом деле она работает гораздо медленнее, чем должна, причем не только из-за повторной передачи пакетов, но и из-за "накладных расходов", требуемых для выявления и обработки пропавших пакетов соответствующими протоколами.

В случае исчезновения пакетов сначала выполните команду **traceroute** (описана ниже) для выяснения маршрута, по которому пакеты следуют к компьютеру-адресату. Затем посредством команды **ping** проверьте все промежуточные шлюзы, через которые пролегает маршрут, и узнайте, на каком этапе теряются пакеты. Для того чтобы точно диагностировать проблему, следует отправить такое количество пакетов, которое позволит получить статистически достоверные результаты. Неисправность сети находится на участке между последним шлюзом, для которого количество потерянных пакетов не больше некоторой заданной величины, и шлюзом, при обращении к которому количество потерянных пакетов превышает эту величину.

Полное время прохождения пакетов, сообщаемое командой **ping**, дает представление об общей скорости передачи пакета по сети. Колебания этого значения, как правило, не являются признаком проблем. Иногда пакеты задерживаются на десятки и сотни

миллисекунд без видимой причины — просто так работают протокол IP и сама операционная система Linux. Но все-таки следует ожидать, что полное время прохождения пакетов будет более-менее постоянным, за некоторыми исключениями. Большинство современных маршрутизаторов ограничивает скорость выдачи ответов на ICMP-запросы, т.е. маршрутизатор может задержать ответ на пакет команды **ping** в связи с общим ограничением трафика протокола ICMP.

Команда **ping** позволяет задать размер посылаемого эхо-пакета. Если передается пакет, размер которого превышает значение MTU для данной сети (в частности, 1500 байт в сети Ethernet), включается режим принудительной фрагментации. Это позволяет выявлять ошибки передачи данных в самом кабеле и другие низкоуровневые ошибки, например проблемы, связанные с перегрузкой сети или виртуальной частной сетью.

В системах семейства Linux желательный размер пакета в байтах задается с помощью флага **-s**.

```
# ping -s 1500 cuinfo.cornell.edu
```

В системах Solaris, HP-UX и AIX достаточно просто указать желательный размер пакета в конце команды **ping**.

```
# ping cuinfo.cornell.edu 1500
```

Помните, что даже простая команда вроде **ping** может повлечь драматические последствия. В 1998 году так называемая атака Ping of Death (Смертельный Ping) вызвала сбой большого количества UNIX- и Windows-систем. Эта атака началась с отправки чрезмерно большого ping-пакета. Когда фрагментированный пакет был собран заново, он заполнил весь буфер получателя и привел к сбою компьютера. Опасность повторения атаки Ping of Death давно устранена, но некоторые изъяны, связанные с использованием утилиты **ping**, все еще существуют.

Во-первых, сложно отличить неисправность сети от неисправности сервера, пользуясь только этой командой. Потеря эхо-пакетов означает лишь, что *что-то* работает неправильно.

Во-вторых, успешно выполненная команда **ping** не позволяет получить информацию о состоянии исследуемого компьютера. Эхо-запросы обрабатываются в стеке протокола IP и не требуют, чтобы на зондируемом компьютере выполнялся серверный процесс. Получение ответа гарантирует только то, что компьютер включен и ядро системы функционирует. Для проверки работы отдельных служб, таких как HTTP и DNS, следует применять высокоуровневые средства.

## 21.3. Инструмент SmokePing: сбор статистики в работе команды ping во времени

Как указывалось выше, даже исправная сеть иногда теряет пакеты. С другой стороны, сети не должны постоянно терять пакеты, даже если уровень потерь невелик, поскольку последствия для пользователей могут быть непропорционально серьезными. Поскольку высокоуровневые пакеты часто продолжают функционировать даже при потере пакетов, вы можете не заметить потери пакетов, пока не станете активно их отслеживать.

Для этой цели можно использовать утилиту SmokePing — программу с открытым исходным кодом, написанную Тобиасом Ойтикером (Tobias Oetiker) для отслеживания простоев сети. Программа SmokePing посылает несколько ping-пакетов на целевой узел через регулярные интервалы времени. Кроме того, она демонстрирует результаты отсле-

живания каждого соединения через веб-узел и может посылать сигналы, когда возникает неблагоприятная ситуация. Копию программы можно загрузить с сайта `oss.oetiker.ch/smokeping`.

На рис. 21.1 представлен график, построенный программой SmokePing. На вертикальной оси отложены среднее время прохождения ping-пакеты (секунды), а на горизонтальной — даты, в которые проходили ping-пакетов (номера недель). Сплошная черная линия, от которой поднимаются серые пики, означает среднее время прохождения пакетов; сами пики представляют собой время прохождения отдельных пакетов. Поскольку серые пики всегда находятся сверху средней линии, подавляющее большинство пакетов должно проходить по сети за время, близкое к среднему, и с очень небольшой задержкой. Это типичная ситуация.

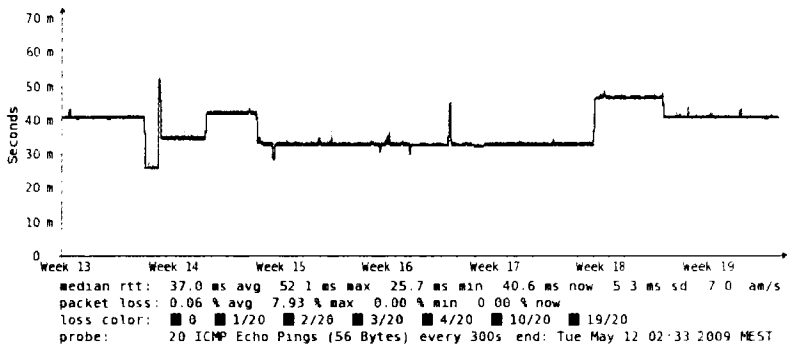



*Рис. 21.1. Пример графика, построенного программой SmokePing*

Ступенчатая форма средней линии означает, что базисная линия времени прохождения пакетов к данному узлу несколько раз изменялась на протяжении периода наблюдений. Наиболее вероятно, что к этому узлу ведут несколько маршрутов или что на самом деле он представляет собой совокупность нескольких узлов, имеющих одно и то же имя DNS, но разные IP-адреса.

## 21.4. Команда TRACEROUTE: трассировка IP-пакетов

Команда **traceroute**, написанная Ваном Джейкобсоном (Van Jacobson), позволяет выявлять последовательность шлюзов, через которые проходит IP-пакет на пути к пункту назначения. Эта команда есть во всех современных операционных системах.[2] Синтаксис ее вызова прост.

**traceroute** *имя_компьютера*

У этой команды очень много параметров, большинство которых в повседневной работе не применяется. Как обычно, *имя_компьютера* может быть задано как имя DNS или IP-адрес. Команда выдает список узлов, начиная с первого шлюза и заканчивая пунктом назначения. Например, на нашем компьютере **jaguar** проверка узла **nubark** дает следующие результаты.

[2] Она есть даже в системе Windows, но в ней она называется **tracert** (если догадаетесь, почему, получите дополнительные баллы по истории).

```
% traceroute nubark
traceroute to nubark (192.168.2.10), 30 hops max, 38 byte packets
 1  lab-gw (172.16.8.254) 0.840 ms  0.693 ms  0.671 ms
 2  dmz-gw (192.168.1.254)  4.642 ms  4.582 ms  4.674 ms
 3  nubark (192.225.2.10) 7.959 ms  5.949 ms  5.908 ms
```

Эти результаты говорят о том, что для попадания с компьютера **jaguar** на узел **nubark** пакеты должны сделать три перехода. Кроме того, они содержат информацию о шлюзах, через которые проходят пакеты. Показано также полное время прохождения пакетов для каждого шлюза — проведено по три замера для каждого перехода. Обычно количество переходов от одного узла Интернет к другому составляет более 15, даже если два сайта расположены через дорогу.

Команда **traceroute** работает путем записи искусственно заниженного значения в поле TTL (Time To Live — время жизни, или предельное число переходов) отправляемого пакета. Когда пакет приходит на очередной шлюз, его значение TTL уменьшается на единицу. Если шлюз обнаруживает, что значение TTL стало равным нулю, он удаляет пакет и возвращает узлу-отправителю специальное ICMP-сообщение: "Время жизни пакета истекло".

🕮 Более подробную информацию об обратном поиске имен в системе DNS можно найти в разделе 17.8.

У первых трех пакетов команды **traceroute** значение TTL равно 1. Первый шлюз, получивший пакет (в нашем примере это **lab-gw**), обнаруживает, что его время жизни истекло. Тогда он удаляет пакет и посылает компьютеру **jaguar** указанное ICMP-сообщение, в поле отправителя которого записан IP-адрес шлюза. Команда **traceroute** обращается к службе DNS и по имеющемуся адресу находит имя шлюза.

Для получения данных о следующем шлюзе посылается очередная группа пакетов, поле TTL которых равно 2. Первый шлюз маршрутизирует эти пакеты и уменьшает значение TTL на единицу. Второй шлюз удаляет пакеты и генерирует описанные выше ICMP-сообщения. Этот процесс продолжается до тех пор, пока пакеты не достигнут нужного компьютера; значение TTL при этом будет равно числу переходов.

Большинство маршрутизаторов посылает свои ICMP-сообщения через тот интерфейс, который является "ближайшим" к узлу-отправителю (т.е. имеет наименьшую метрику стоимости). Если, наоборот, запустить команду **traceroute** на узле-получателе, чтобы узнать маршрут к исходному компьютеру, то, скорее всего, будет получен другой список IP-адресов, соответствующий тому же набору маршрутизаторов. Иногда весь набор шлюзов оказывается совершенно другим; такая конфигурация называется *асимметричной*.

Для каждого значения TTL команда **traceroute** посылает три пакета, что иногда приводит к интересным результатам. Если промежуточный шлюз распределяет трафик по нескольким маршрутам, то пакеты могут возвращаться разными компьютерами. В таком случае команда **traceroute** сообщает обо всех полученных ответах.

Рассмотрим пример, в котором посредством команды **traceroute** определяется маршрут с компьютера в Швейцарии к домену `caida.org` в суперкомпьютерном центре Сан-Диего (San Diego Supercomputer Center). [3]

```
linux$ traceroute caida.org
traceroute to caida.org (192.172.226.78), 30 hops max, 46 byte packets
 1 gw-oetiker.init7.net (213.144.138.193) 1.122 ms 0.182 ms 0.170 ms
 2 r1zur1.core.init7.net (77.109.128.209) 0.527 ms 0.204 ms 0.202 ms
 3 r1fra1.core.init7.net (77.109.128.250) 18.279 ms 6.992 ms 16.597 ms
```

[3] Мы удалили несколько цифр в дробной части миллисекунд, чтобы сделать строки короче.

```
 4 rlams1.core.init7.net (77.109.128.154) 19.549 ms 21.855 ms 13.514 ms
 5 rllon1.core.init7.net (77.109.128.150) 19.165 ms 21.157 ms 24.866 ms
 6 rllax1.ce.init7.net (82.197.168.69) 158.232 ms 158.224 ms 158.271 ms
 7 cenic.laap.net (198.32.146.32) 158.349 ms 158.309 ms 158.248 ms
 8 dc-lax-core2--lax-peer1-ge.cenic.net (137.164.46.119) 158.60 ms * 158.71 ms
 9 dc-tus-agg1--lax-core2-10ge.cenic.net (137.164.46.7) 159 ms 159 ms 159 ms
10 dc-sdsc-sdsc2--tus-dc-ge.cenic.net (137.164.24.174) 161 ms 161 ms 161 ms
11 pinot.sdsc.edu (198.17.46.56) 161.559 ms 161.381 ms 161.439 ms
12 rommie.caida.org (192.172.226.78) 161.442 ms 161.445 ms 161.532 ms
```

Эти строки свидетельствуют о том, что пакеты долго путешествовали внутри сети Init Seven. Иногда по именам шлюзов можно догадаться об их географическом местоположении. Сеть Init Seven раскинулась от Цюриха (`zur`) до Франкфурта (`fra`), Амстердама (`ams`), Лондона (`lon`) и, наконец, до Лос-Анджелеса (`lax`). В этом месте пакеты достигли сайта `cenic.net`, который доставил пакеты на узел `caida.org` внутри сети суперкомпьютерного центра Сан-Диего (`sdsc`) в городе Ла Хойя (La Jolla)[4], шт. Калифорния.

На восьмом переходе вместо одного из значений времени отображается звездочка. Это означает, что на один из отправленных пакетов не был получен ответ. Такая ситуация может быть вызвана перегрузкой сети, но это не единственное возможное объяснение. Команда **`traceroute`** использует низкоприоритетные ICMP-пакеты, которые отбрасываются многими маршрутизаторами, отдающими предпочтение "реальному" трафику. Появление нескольких звездочек в выводе команды **`traceroute`** не является причиной для беспокойства.

Если для какого-то шлюза отображаются все три звездочки, значит, от него не было получено ни одного сообщения. Вероятнее всего, шлюз просто выключен. Правда, иногда шлюз конфигурируют таким образом, чтобы он игнорировал устаревшие пакеты. В этом случае пакеты благополучно перейдут на следующий шлюз. Другой возможной причиной появления звездочек может быть то, что ответные пакеты от шлюза возвращаются слишком долго и команда **`traceroute`** перестает ожидать их прибытия.

Некоторые брандмауэры полностью блокируют передачу ICMP-сообщений о пакетах с истекшим временем жизни. Если маршрут пролегает через один из таких брандмауэров, команда **`traceroute`** не сможет получить информацию о шлюзах, находящихся за ним. Однако она узнает общее число переходов до пункта назначения, так как отправляемые эхо-пакеты пройдут весь маршрут.

Некоторые брандмауэры могут также блокировать исходящие UDP-пакеты, которые команда **`traceroute`** посылает для инициирования ICMP-ответов. В этом случае команда не выдаст никакой полезной информации. Если ваш брандмауэр не дает выполнить команду **`traceroute`**, убедитесь, что его конфигурация открывает порт 33434-33534 и допускает пакеты ICMP ECHO (тип 8).

Медленная передача данных не всегда свидетельствует о неисправности сети. Некоторые сети действительно имеют большую задержку, особенно если в них используется технология UMTS/EDGE/GRPS. Ярким примером являются беспроводные сети. Медлительность может быть также следствием перегрузки сети. Определить это можно по нелогичному значению полного времени прохождения пакета.

Иногда можно увидеть символы `!N` вместо значения времени. Это говорит о том, что соответствующий шлюз вернул сообщение о "недостижимости" сети, т.е. он не знает, куда посылать пакет. Сообщения о "недостижимости" узла или протокола помечаются как `!H` или `!P` соответственно. Шлюз, от которого получено одно из указанных сообщений, является последним в цепочке и обычно имеет проблемы с маршрутизацией (воз-

[4] Пригород Сан-Диего. — *Примеч. ред.*

можно, они вызваны разрывом связи): либо его статические маршруты заданы неверно, либо протоколы динамической маршрутизации не могут определить правильный маршрут для передачи пакета получателю.

Если команда **traceroute** не работает (или работает слишком медленно), это может быть вызвано превышением тайм-аута при попытках узнать имя компьютера в DNS. Возможно, на компьютере не функционирует служба DNS. В таком случае воспользуйтесь командой **traceroute -n**. Флаг **-n** отменяет поиск имен, и только этот способ иногда позволяет все же выполнить команду **traceroute** в поврежденных сетях.

Для выполнения команды **traceroute** необходимо иметь права суперпользователя. Чтобы обычный пользователь могут выполнить эту команду, она должна быть инсталлирована с установленным битом setuid. В некоторые дистрибутивные пакеты системы Linux команда **traceroute** включается без установленного бита setuid. В зависимости от окружения и потребностей, бит setuid можно включить снова или предоставить заинтересованным пользователям доступ к этой команде с помощью команды **sudo**.

В последние годы появилось несколько утилит, аналогичных команде **traceroute**, которые могут проходить через брандмауэры, блокирующие ICMP-пакеты. Обзор этих инструментов можно найти на странице `tinyurl.com/y99qh6u` на сайте PETTKB Wiki. Нам особенно понравилась утилита **mtr**, которая имеет интерфейс, похожий на интерфейс утилиты **top**, и работает, как утилита **traceroute**. Очень хорошо сделано!

В зависимости от вида маршрутизации, иногда полезно посмотреть на свой сайт с точки зрения внешнего мира. Несколько веб-служб для отслеживания маршрутов позволяют выполнять функции, обратные утилите **traceroute**, не выходя из браузера. Томас Кернен (Tomas Kernen) поддерживает список этих служб на сайте `traceroute.org`.

# 21.5. Команда NETSTAT: получение информации о состоянии сети

Команда **netstat** выдает различную информацию о состоянии сетевого программного обеспечения, включая статистику сетевых интерфейсов, данные о маршрутизации и таблицы соединений. Никакого объединяющего звена во всех этих информационных блоках нет, просто они касаются функционирования сети. Команду **netstat** можно сравнить с "кухонным сливом" сетевых инструментов — она отражает много сетевой информации, которая не появляется больше нигде.

Мы рассмотрим пять наиболее распространенных вариантов использования команды **netstat**, а именно:

- проверка состояния сетевых соединений;
- анализ конфигурации интерфейсов;
- изучение таблицы маршрутизации;
- проверка таблицы маршрутизации;
- получение статистических данных о различных сетевых протоколах.

## Контроль состояния сетевых соединений

Команда **netstat -i** демонстрирует конфигурацию и состояние каждого сетевого интерфейса узла и данные соответствующих счетчиков трафика. Результаты обычно выводятся в виде таблицы, структура которой зависит от операционной системы.

```
solaris$ netstat -i
Name    Mtu  Net/Dest Address   Ipkts     Ierrs  Opkts     Oerrs  Collis  Queue
lo0     8232 loopback localhost 319589661 0      319589661 0      0       0
e1000g1 1500 host-if1 host-if1  369842112 0      348557584 0      0       0
e1000g2 1500 host-if2 host-if2  93141891  0      121107161 0      0       0
```

```
hp-ux$ netstat -i
Name   Mtu   Network       Address           Ipkts    Ierrs  Opkts    Oerrs  Coll
lan0   1500  192.168.10.0  hpux11            2611259  0      2609847  0      0
lo0    32808 loopback      hpux11.atrust.com
```

```
aix$ netstat -i
Name  Mtu    Network     Address  ZoneID   Ipkts   Ierrs   Opkts   Oerrs  Coll
en3   1500   link#2      0.11.25.39.e0.b6  41332     0     14173     3      0
en3   1500   192.168.10  IBM               41332     0     14173     3      0
lo0   16896  link#1                        1145121   0     1087387   0      0
lo0   16896  127         loopback          1145121   0     1087387   0      0
lo0   16896  ::1                    0      1145121   0     1087387   0      0
```

```
linux$ ifconfig -a
eth0     Link encap:EthernetHWaddr 00:15:17:4c:4d:00
         inet addr:192.168.0.203Bcast:192.168.0.255Mask:255.255.255.0
         inet6 addr: fe80::215:17ff:fe4c:4d00/64 Scope:Link
         UP BROADCAST RUNNING MULTICASTMTU:1500Metric:1
         RX packets:559543852 errors:0 dropped:62 overruns:0 frame:0
         TX packets:457050867 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:478438325085 (478.4 GB)TX bytes:228502292340 (228.5 GB)
         Memory:b8820000-b8840000
eth1     Link encap:EthernetHWaddr 00:15:17:4c:4d:01
         BROADCAST MULTICASTMTU:1500Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)TX bytes:0 (0.0 B)
         Memory:b8800000-b8820000
lo       Link encap:Local Loopback
         inet addr:127.0.0.1Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNINGMTU:16436Metric:1
         RX packets:1441988 errors:0 dropped:0 overruns:0 frame:0
         TX packets:1441988 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:327048609 (327.0 MB)TX bytes:327048609 (327.0 MB)
```

Этот узел имеет два сетевых интерфейса: один предназначен для регулярного трафика, а второй в настоящее время не используется (он не имеет IP-адреса и не отмечен меткой UP). Команды RX packets и TX packets сообщают количество пакетов, полученных интерфейсом и прошедших через него после загрузки компьютера. В разделах, предназначенных для подсчета ошибок, регистрируются многие виды ошибок, и обычно в них можно увидеть небольшие числа.

Количество ошибок не должно превышать 1% от связанных с ними пакетов. Если уровень ошибок выше, сравните его с уровнем ошибок на соседних компьютерах. Большое количество ошибок на отдельном компьютере свидетельствует о проблеме с интерфейсом или соединением. Высокий уровень ошибок почти наверняка означает проблему с окружением или сетью. Одна из наиболее частых причин большого количества ошибок заключается в том, что несоответствие скорости работы платы Ethernet или дуплекса вызывает сбой при автоопределении.

Несмотря на то что коллизия — это разновидность ошибки, программа **nestat** подсчитывает ее отдельно. Поле с заголовком Collisions содержит количество коллизий, зарегистрированных при отправке пакетов. На основе этого числа можно вычислить процент исходящих пакетов (TX packets), вызвавших коллизию.

В коммутированной сети с полнодуплексными связями, т.е. в современных версиях плат Ethernet, коллизий быть не должно, даже если сеть перегружена. Если обнаружены коллизии, значит, произошло нечто серьезное. Кроме того, следует проверить, что поток управления поступает на ваши коммутаторы и маршрутизаторы, особенно если ваша сеть содержит связи с разными скоростями.

Причинами сетевых проблем часто становятся дешевые компоненты настольного сетевого оборудования, например коммутатор с временной проводкой, который следует заменить.

## Отслеживание состояния сетевых соединений

Если программа **netstat** запущена без аргументов, она выводит на экран состояние активных TCP- и UDP-портов. Неактивные ("прослушивающие") серверы, ожидающие соединения, как правило, скрываются, но вы можете увидеть их, выполнив команду **netstat -a**.[5] Результаты ее работы выглядят следующим образом.

```
linux$ netstat -a
Active Internet connections (servers and established)
Proto  Recv-Q  Send-Q  Local Address   ForeignAddress    State
tcp    0       0       *:ldap          *:*               LISTEN
tcp    0       0       *:mysql         *:*               LISTEN
tcp    0       0       *:imaps         *:*               LISTEN
tcp    0       0       bull:ssh        dhcp-32hw:4208    ESTABLISHED
tcp    0       0       bull:imaps      nubark:54195      ESTABLISHED
tcp    0       0       bull:http       dhcp-30hw:2563    ESTABLISHED
tcp    0       0       bull:imaps      dhcp-18hw:2851    ESTABLISHED
tcp    0       0       *:http          *:*               LISTEN
tcp    0       0       bull:37203      baikal:mysql      ESTABLISHED
tcp    0       0       *:ssh           *:*               LISTEN
...
```

[5] В отчете показаны также соединения для "сокетов доменов UNIX", но поскольку они не относятся к работе сети, мы их не обсуждаем.

Этот отчет сильно сокращен; например, здесь не показаны соединения сокетов UDP и UNIX. В этом отчете упомянуты входящее SSH-соединение, два входящих IMAPS-соединения, одно входящее HTTP-соединение, исходящее MySQL-соединение и группа портов, прослушивающих остальные соединения.

Адреса представлены в формате *имя_компьютера.служба*, где *служба* — это номер порта. Для известных служб порты указаны в символическом виде (соответствия между номерами портов и их именами определены в файле **/etc/services**). При наличии опции **-n** все адреса отображаются в числовом виде. Помните: когда служба DNS не функционирует, команда **netstat** будет выполняться очень медленно, если не указать флаг **-n**.

В колонках `Send-Q` и `Recv-Q` показывается, сколько запросов находится во входящих и исходящих очередях на данном компьютере. На другом конце соединения размеры очередей могут быть иными. Желательно, чтобы эти значения были близки к нулю и не были ненулевыми постоянно. Конечно, если команда **netstat** запускается через сетевой терминал, для ее соединения размер исходящей очереди, скорее всего, никогда не будет равен нулю.

Состояние соединения определено только для протокола TCP. Протокол UDP не проверяет факт установления соединения. Наиболее распространенные состояния таковы: `ESTABLISHED` (установлено) — для активных соединений, `LISTEN` (ожидание) — для серверов, ожидающих поступления запросов (при отсутствии опции **-a** обычно не показываются), `TIME_WAIT` (ожидание закрытия) — для соединений, находящихся в процессе закрытия.

Эта информация полезна, главным образом, для устранения проблем на высоком уровне, если, конечно, базовые сетевые средства работают нормально. Команда **netstat** позволяет проверить правильность настройки серверов и диагностировать определенные виды нарушений связи, особенно при работе с протоколом TCP. Например, если соединение находится в состоянии `SYN_SENT`, то это означает наличие процесса, который пытается установить контакт с несуществующим или недоступным сервером.

Если команда **netstat** сообщает о большом количестве соединений, находящихся в состоянии `SYN_WAIT`, то компьютер, очевидно, не в состоянии обработать имеющееся число запросов на установление соединений. Такая ситуация может быть вызвана ограничениями системного ядра или даже злонамеренными попытками вызвать перегрузку.

🕮 Информация о настройке ядра приводилась в главе 13.

## Идентификация прослушивающих сетевых служб

Один из наиболее часто возникающих вопросов, связанных с безопасностью, звучит так: "Какие процессы на данном компьютере прослушивает сеть в ожидании входящих соединений?" Команда **netstat -a** показывает все порты, активно прослушивающие соединения (любой TCP-порт в состоянии `LISTEN` и потенциально любой UDP-порт), но на загруженной машине эти линии могут оказаться затерянными в шуме установленных TCP-соединений.

В системе Linux для выявления только прослушивающих портов следует использовать команду **netstat -l**. Кроме того, можно добавить флаг **-p**, чтобы идентифицировать специфичные процессы, связанные с каждым прослушивающим портом.[6] Пример, продемонстрированный ниже, описывает три обычные службы (**sshd**, **sendmail** и **named**) и одну необычную.

[6] В системах UNIX, не поддерживающих флаг **-p**, эту и другую информацию можно получить с помощью команды ***lsof***, описанной в разделе 6.2.

```
linux$ netstat -lp
...
tcp    0   0   0.0.0.0:22    0.0.0.0:*    LISTEN      23858/sshd
tcp    0   0   0.0.0.0:25    0.0.0.0:*    LISTEN      10342/sendmail
udp    0   0   0.0.0.0:53    0.0.0.0:*                30016/named
udp    0   0   0.0.0.0:962   0.0.0.0:*                38221/mudd
...
```

Служба **mudd** с идентификатором PID 38221 прослушивает UD-порт 962. Если вы не знаете, что такое служба **mudd**, уточните в справочной системе.

Для безопасности полезно смотреть на компьютеры с точки зрения внешнего пользователя, запускающего сканер порта. Для этого очень полезна команда **nmap** (см. раздел 22.8).

## Проверка таблицы маршрутизации

Команда **netstat -r** отображает таблицу маршрутизации ядра. Следующие результаты получены на компьютере, который имеет два сетевых интерфейса.

```
redhat$ netstat -r -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0   0      0    eth1
192.168.2.0     0.0.0.0         255.255.255.0   U     0   0      0    eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0   0      0    lo
0.0.0.0         192.168.1.254   0.0.0.0         UG    0   0      0    eth0
```

Пункты назначения и шлюзы могут быть представлены доменными именами либо IP-адресами. Флаг **-n** задает вывод IP-адресов.

В колонке `Flags` отображаются флаги, характеризующие маршрут: `U` (up) — активный, `G` (gateway) — шлюзовой, `H` (host) — узловой (связан с конкретным адресом, а не сетью). Флаг `D` (не показан) обозначает маршрут, полученный в результате переадресации по протоколу ICMP. Флаги `G` и `H`, стоящие вместе, обозначают маршрут к компьютеру, проходящий через промежуточный шлюз. Остальные поля содержат статистические данные о маршруте: текущее число TCP-соединений по этому маршруту, количество отправленных пакетов и имя используемого интерфейса.

О таблицах маршрутизации рассказывалось в разделе 14.5.

Рассмотренный вариант команды **netstat** полезен для проверки правильности таблицы маршрутизации. Особенно важно убедиться в наличии и корректности стандартного маршрута. Он обозначается в виде адреса со всеми нулями (0.0.0.0) или словом `default`. Стандартный маршрут может не существовать, но такая конфигурация крайне нетипична.

## Просмотр статистических данных функционирования различных сетевых протоколов

Команда **netstat -s** отображает содержимое всевозможных счетчиков, используемых в сетевых программах. Информация разбивается на разделы в соответствии с протоколами: IP, ICMP, TCP и UDP. Ниже показаны отдельные фрагменты листинга команды **netstat -s**, полученного на компьютере-шлюзе; они отредактированы, чтобы остались только наиболее интересные данные.

```
Ip:
    671349985 total packets received
    0 forwarded
    345 incoming packets discarded
    667912993 incoming packets delivered
    589623972 requests sent out
    60 dropped because of missing route
    203 fragments dropped after timeout
```

Проверьте, что ни один пакет не был потерян или забракован. Некоторые входящие пакеты могут быть забракованы, но быстрое увеличение этого показателя обычно означает уменьшение объема памяти или другую проблему, связанную с ресурсами.

```
Icmp:
  242023 ICMP messages received
  912 input ICMP message failed.
  ICMP input histogram:
     destination unreachable: 72120
     timeout in transit: 573
     echo requests: 17135
     echo replies: 152195
  66049 ICMP messages sent
  0 ICMP messages failed
  ICMP output histogram:
     destination unreachable: 48914
     echo replies: 17135
```

В этом примере количество эхо-запросов в разделе входа соответствует количеству эхо-ответов в разделе выхода. Обратите внимание на то, что сообщение "destination unreachable" может генерироваться даже в том случае, когда все пакеты явно допускают перенаправление.

Неисправные пакеты в конец концов достигнут шлюза, который их отвергнет, и сообщение об ошибках будет отправлено обратно по цепочке шлюзов.

```
Tcp:
    4442780 active connections openings
    1023086 passive connection openings
    50399 failed connection attempts
    0 connection resets received
    44 connections established
    666674854 segments received
    585111784 segments send out
    107368 segments retransmited
    86 bad segments received.
    3047240 resets sent
Udp:
    4395827 packets received
    31586 packets to unknown port received.
    0 packet receive errors
    4289260 packets sent
```

Рекомендуем определить приемлемые диапазоны этих показателей, чтобы можно было сразу же распознать ненормальное состояние сети.

## 21.6. Проверка функционирования интерфейса в реальном времени

Хороший способ обнаружить сетевые проблемы — посмотреть, что происходит прямо сейчас. Сколько пакетов было послано за последние пять минут через данный интерфейс? Сколько байтов? Происходили ли коллизии или другие ошибки? Ответить на все эти вопросы можно, наблюдая за функционированием сети в реальном времени. Для этого были разработаны разные инструменты.

В системе Solaris в командную строку **netstat -i** можно добавить интервал в секундах и значение счетчика.

```
solaris$ netstat -i 2 3
   input    e1000g          output    input     (Total)          output
packets    errs packets    errs      colls     packets errs packets errs colls
17861       0   26208      0         0         17951   0    26298   0    0
4           0   2          0         0         4       0    2       0    0
1           0   1          0         0         1       0    1       0    0
```

В системах HP-UX и AIX используется единственное число, которое задает интервал (в секундах), за который следует собрать статистические показатели.

```
$ netstat -i 2
(lan0)-> input      output     (Total)-> input    output
       packets     packets     packets            packets
       9053713     9052513    10115002           10113803
             8           8           8                  8
            22          22          22                 22
             9           9           9                  9
...
```

Команда **netstat** в системе Linux не имеет параметра, задающего интервал, поэтому в системе Linux мы рекомендуем использовать совершенно другой инструмент: утилиту **sar**. (Утилита **sar** рассматривается в разделе 29.4 с точки зрения общего мониторинга системы.) В большинстве дистрибутивных пакетов утилита **sar** по умолчанию не инсталлируется, но всегда доступна как дополнительная функциональная возможность. Пример, продемонстрированный ниже, содержит отчеты, которые формируются каждые две секунды в течение одной минуты (т.е. 30 отчетов). Аргумент **DEV** представляет собой литеральное ключевое слово, а не обозначение устройства или имя интерфейса.

```
redhat$ sar -n DEV 2 30
17:50:43 IFACE   rxpck/s txpck/s  rxbyt/s  txbyt/s rxcmp/s  txcmp/s rxmcst/s
17:50:45     lo      3.61     3.61   263.40   263.40    0.00     0.00     0.00
17:50:45   eth0     18.56    11.86  1364.43  1494.33    0.00     0.00     0.52
17:50:45   eth1      0.00     0.00     0.00     0.00    0.00     0.00     0.00
```

Этот пример получен на компьютере, работающем под управлением операционной системы Red Hat, с двумя сетевыми интерфейсами. Вывод содержит как моментальные, так и средние данные об использовании интерфейса в байтах и пакетах. Совершенно очевидно, что второй интерфейс (eth1) не используется вообще.

В первых двух столбцах приведены моменты времени, в которые были сделаны измерения, а также имена сетевых интерфейсов. Следующие два столбца содержат количество полученных и переданных пакетов соответственно.

Столбцы `rxbyt/s` и `txbyt/s`, вероятно, являются самыми полезными, поскольку содержат информацию о реально используемой пропускной способности. Последние три столбца состоят из статистических показателей о сжатых (`rxcmp/s`, `txcmp/s`) и многоадресных (`rxmcst/s`) пакетах.

Команда **`sar -n DEV`** особенно полезна для отслеживания источников ошибок. Команда **`ifconfig`** может предупредить о существовании проблем, но не способна сообщить, является ли ошибка следствием постоянной низкоуровневой проблемы или кратковременного, но катастрофического события. В таком случае для того, чтобы понять, что происходит, необходимо наблюдать за работой сети в течение некоторого времени и при разных условиях загруженности. Попробуйте выполнить команду **`ping`** с загрузкой большого по размеру пакета, а затем просмотрите отчет о результатах выполнения команды **`sar -n DEV`**.

## 21.7. Анализаторы пакетов

Программы **`tcpdump`** и Wireshark относятся к классу утилит, известных как *анализаторы пакетов*. Они следят за трафиком в сети и регистрируют либо выводят на экран пакеты, удовлетворяющие заданным критериям. Например, можно перехватывать все пакеты, посылаемые на какой-то компьютер или с него, либо TCP-пакеты, относящиеся к конкретному сетевому соединению.

Анализаторы пакетов полезны как для решения известных проблем, так и для выявления абсолютно новых. Важно время от времени запускать эти программы и проверять, нормально ли обрабатывается сетевой трафик.

Поскольку анализаторы должны уметь перехватывать пакеты, которые локальный компьютер обычно не получает (или на которые не обращает внимания), базовые сетевые аппаратные средства должны разрешать доступ к каждому пакету. Это характерно для широковещательных технологий, в частности Ethernet, а также для большинства современных локальных сетей.

Анализаторы пакетов должны иметь доступ к низкоуровневому трафику, поэтому их работе могут мешать коммутаторы, одной из функций которых является препятствие распространению "ненужных" пакетов. Однако с помощью анализаторов удается получить полезную информацию даже в коммутируемых сетях. Можно, к примеру, обнаружить проблемы, затрагивающие широковещательные и групповые пакеты. Объем выдаваемой информации зависит от модели коммутатора.

🕮 О коммутаторах рассказывалось в разделе 16.1.

Аппаратный интерфейс должен не только иметь возможность получать доступ ко всем сетевым пакетам, но и содержать механизм, обеспечивающий передачу этих пакетов вверх на программный уровень. Ведь обычно адреса пакетов проверяются на аппаратном уровне, а ядру предоставляются лишь широковещательные/групповые пакеты и те, которые адресованы данному компьютеру. В *беспорядочном режиме* (promiscuous mode) интерфейс позволяет ядру получать все сетевые пакеты, даже если они предназначены для других компьютеров.

Анализаторы понимают многие форматы пакетов, используемые стандартными демонами, и часто могут отображать содержимое пакетов в удобном для восприятия виде. Это облегчает пользователю анализ сеанса между двумя программами. Существуют анализаторы, которые, кроме заголовка пакета, выводят и его содержимое в текстовом виде, что полезно при исследовании протоколов верхних уровней.

Некоторые из этих протоколов пересылают информацию (в том числе и пароли) по сети в незашифрованном виде, поэтому следует заботиться о сохранении конфиденциальности пользовательских данных. С другой стороны, ничто так не подчеркивает необходимость шифрования, как вид паролей, пересылаемых в открытом виде в сетевом пакете.

В связи с тем что анализатору необходим доступ к пакетам на самом низком уровне, он должен запускаться от имени пользователя **root**. Подобное ограничение снижает шансы обычных пользователей получить доступ ко всему сетевому трафику, но на самом деле этот барьер можно преодолеть. Во многих организациях анализаторы пакетов удалены с большинства компьютеров с целью уменьшения риска злоупотребления этими программами. Если такая мера невозможна, следует проверять интерфейсы системы, чтобы они не работали в "беспорядочном" режиме без ведома или согласия администратора. В Linux интерфейс, работающий в "беспорядочном" режиме, помечается в выводе команды **ifconfig** флагом PROMISC. В системах семейства Linux факт, что интерфейс был переключен в "беспорядочный режим", также записывается в журнал ядра.

## Утилита tcpdump: стандартный анализатор

Замечательный анализатор пакетов **tcpdump**, написанный Ваном Якобсоном (Van Jacobson), входит в состав большинства дистрибутивов Linux. Он уже давно считается промышленным стандартом, и другие анализаторы читают и/или записывают файлы трассировки в формате **tcpdump**, который в настоящее время называется **libpcap**.

По умолчанию утилита **tcpdump** использует первый найденный ею сетевой интерфейс. Если она выбрала не тот интерфейс, то посредством опции **-i** следует задать нужный. В случае неисправности службы DNS или если необходимо видеть адреса компьютеров, воспользуйтесь опцией **-n**. Эта опция имеет большое значение, так как медленная работа DNS может привести к удалению пакетов еще до того, как они будут проанализированы утилитой **tcpdump**.

Опция **-v** позволяет получить более детальное описание каждого пакета, а самые подробные результаты выдаются при задании опции **-w**. Если указана опция **-w**, утилита сохраняет перехваченные пакеты в файле. Для чтения пакетов из файла предназначена опция **-r**.

Отметим, что команда **tcpdump -w** по умолчанию сохраняет только заголовки пакетов. Это сделано для уменьшения размеров распечаток, но при этом можно потерять наиболее полезную и релевантную информацию. Поэтому, если вы уверены, что, кроме заголовков, вам нужна дополнительная информация, используйте флаг **-s** со значением, которое равно 1056 (фактические значения зависят от максимального размера пакета (MTU)), чтобы иметь возможность проанализировать весь пакет.

В качестве примера рассмотрим укороченный вариант распечатки, полученной на машине **nubark**. Спецификация фильтра **host bull** ограничивает вывод пакета на экран только информацией, относящейся к машине **bull**, которая может быть как источником, так и получателем.

```
$ sudo tcpdump host bull
12:35:23.519339 bull.41537 > nubark.domain: A? atrust.com. (28) (DF)
12:35:23.519961 nubark.domain > bull.41537: A 66.77.122.161 (112) (DF)
```

Первый пакет свидетельствует о том, что компьютер **bull** послал компьютеру **nubark** запрос на поиск домена atrust.com в системе DNS. Ответом является IP-адрес машины, связанной с этим именем, т.е. 66.77.122.161. Обратите внимание на временную метку в левой части строки и на то, что команда **tcpdump** понимает протокол на уровне

приложения (в данном случае DNS). Номер порта на компьютере **bull** является произвольным и записывается в виде числа (41537), но поскольку номер порта на сервере хорошо известен (53), команда **tcpdump** выводит его символическое имя `domain`.

Анализаторы пакетов могут порождать огромные массивы информации, неподъемные не только для человека, но и для операционной системы. Для того чтобы избежать перегрузки сети, утилита **tcpdump** позволяет задавать сложные фильтры. Например, следующий фильтр собирает только входящий веб-трафик из одной подсети.

```
$ sudo tcpdump src net 192.168.1.0/24 and dst port 80
```

Справочная страница утилиты **tcpdump** содержит несколько хороших примеров сложных фильтров, а также список примитивов.[7]

Система Solaris содержит анализатор, работающий так же, как **tcpdump**. Он называется **snoop**. Дистрибутивные пакеты систем HP-UX, AIX и большинства систем семейства Linux не содержат таких утилит.

```
solaris$ snoop
Using device /dev/e1000g0 (promiscuous mode)
nubark -> solaris TCP D=22 S=58689 Ack=2141650294 Seq=3569652094 Len=0
  Win=15008 Options=<nop,nop,tstamp 292567745 289381342>
nubark -> solaris TCP D=22 S=58689 Ack=2141650358 Seq=3569652094 Len=0
  Win=15008 Options=<nop,nop,tstamp 292567745 289381342>
? -> (multicast) ETHER Type=023C (LLC/802.3), size = 53 bytes
...
```

Если вы используете зоны Solaris, обратите внимание на то, что утилита **snoop** работает правильно, только если вы устраняете проблему в локальной зоне.

## Утилиты Wireshark и TShark: усовершенствованный вариант tcpdump

Утилита **tcpdump** появилась довольно давно, но в последнее время все более популярным становится новый открытый пакет под названием Wireshark (который ранее назывался Ethereal). Пакет Wireshark активно совершенствуется и содержит больше функциональных возможностей, чем большинство коммерческих приложений. Это необычайно мощный инструмент анализа, который должен входить в инструментальный набор каждого эксперта. Кроме того, он незаменим при обучении.

Пакет Wireshark содержит как пользовательский графический интерфейс (**wireshark**), так и интерфейс командной строки (**tshark**). В системе Linux он инсталлируется автоматически. Администраторы системы UNIX должны посетить сайт `wireshark.org`, на котором хранится открытый код и множество скомпилированных модулей.

Программа Wireshark может читать и записывать файлы слежения в форматах, используемых многими другими анализаторами пакетов. Еще одна удобная функциональная возможность заключается в том, что одним щелчком мыши можно выбрать любой пакет в сеансе TCP и попросить программу Wireshark заново собрать (соединить вместе) данные, включенные во все пакеты, находящиеся в потоке. Эта возможность полезна, если вы хотите проверить данные, переданные в ходе полного TCP-обмена, например, какое соединение было использовано для передачи сообщения об ошибке по сети.

[7] Если ваши требования, касающиеся фильтрации, превосходят возможности утилиты ***tcpdump***, обратите внимание на утилиту ***ngrep*** (`ngrep.sourceforge.net`), которая может фильтровать пакеты в соответствии с их содержимым.

Фильтры перехвата программы Wireshark функционально идентичны фильтрам утилиты **tcpdump**, поскольку программа Wireshark использует ту же самую библиотеку **libpcap**. Тем не менее следует быть внимательным: важный нюанс заключается в том, что программа Wireshark имеет дополнительную функцию "фильтры дисплея", которая влияет на внешний вид данных, а не содержание, перехваченное анализатором. Фильтр дисплея имеет более мощный синтаксис, чем библиотека **libpcap**, поддерживаемая в момент перехвата. Фильтры дисплея похожи на фильтры библиотеки **libpcap**, но они не совпадают с ними.

Программа Wireshark имеет встроенные дешифраторы для многих сетевых протоколов, включая многочисленные протоколы, используемые для реализации сетей хранения данных. Она преобразует пакет в структурированное информационное дерево, в котором каждый бит информации описан на английском языке.

На рис. 21.2 показан перехват DNS-запроса и ответа на него в программе Wireshark.

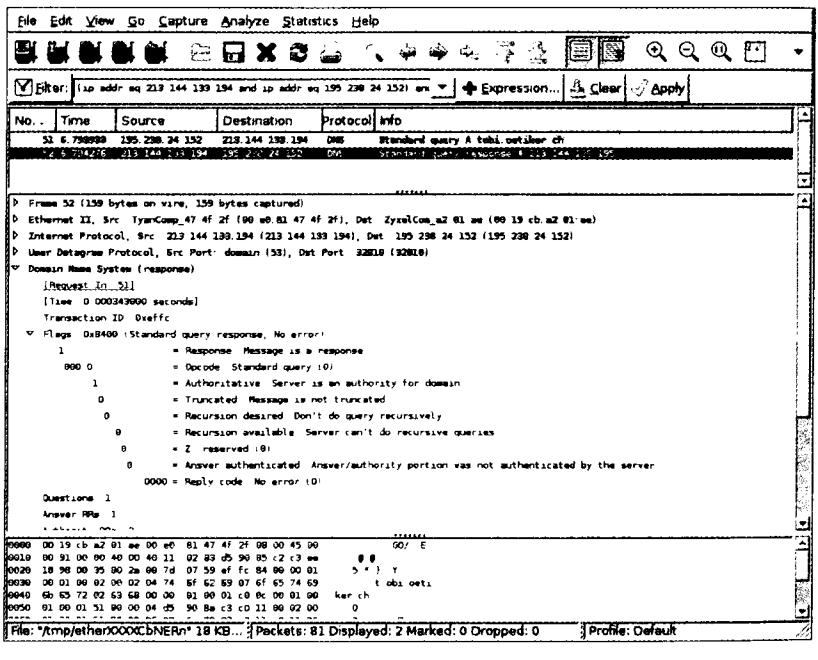



*Рис. 21.2. Пара DNS-пакетов в программе Wireshark*

В таблице, расположенной в верхней части рисунка, указаны два пакета. Первый пакет представляет собой запрос на пересылку на DNS-сервер, а второй пакет является полученным ответом. Выбран пакет запроса, поэтому в среднем окне показана его структура. В нижнем окне пакет представлен в виде набора байтов.

Раскрытый раздел дерева демонстрирует данные, содержащиеся в пакете. Содержание пакета в виде байтов также может представлять интерес, потому что иногда он содержит фрагменты, которые могут подсказать, что именно произошло. Сканировать текст особенно удобно, когда нет встроенного анализатора текущего протокола. Справочное меню программы Wireshark содержит массу примеров, с которыми можно начинать работать. Экспериментируйте!

Следует сделать одно предупреждение, касающееся программы Wireshark: несмотря на множество прекрасных функциональных возможностей, она еще требует многолетней доработки с точки зрения безопасности. Запустите текущую версию, но не оставляйте ее работать постоянно на важных компьютерах; программа Wireshark может стать потенциальным проводником для атаки.

## 21.8. Служба Netalizr Института ICSI

Мы рассмотрели несколько инструментов для отладки сети и анализа сетевой конфигурации. Но даже при самом тщательном мониторинге полезно иметь инструмент, который время от времени будет просматривать вашу сеть. Netalyzr — это служба, предоставляемая Международным институтом компьютерных наук из Беркли (International Computer Science Institute at Berkeley), которая представляет собой полезную альтернативу. Для ее использования просто зайдите на сайт `netalyzr.icsi.berkeley.edu` (обратите внимание на то, что буква 'e' пропущена специально), включив в вашем браузере поддержку языка Java.

Служба Netalyzr проверяет ваше интернет-соединение разными способами. Очень полезно иметь доступ к вашей сети как изнутри (через программу Java, выполняемую в вашем браузере), так и с серверов института ICSI.

На рис. 21.3 показан отчет службы Netalyzr для рабочей станции в частной сети, соединенной с внешним миром связью DSL. Служба Netalyzr отлично устанавливается, но имеет небольшие проблемы с веб-сервером Apache (тем не менее блокирование неверно составленных HTTP-запросов может оказаться полезным).

Полный отчет состоит из разделов, содержащих информацию о IP-соединениях, полосе пропускания, простоях, буферизации, обработке фрагментированных пакетов и других аспектах. Особенно сильны тесты для выявления аномалий в системе DNS и протоколе HTTP.

## 21.9. Протоколы управления сетями

Усложнение и разрастание сетей за последние 20 лет вызвало потребность в разработке средств управления сетями. Поставщики операционных систем и разработчики стандартов подходили к решению этой задачи самыми разными способами. Наиболее значительным результатом стало появление ряда стандартных протоколов управления сетевыми устройствами и множества высокоуровневых программных средств, в которых эти протоколы используются.

Протоколы управления сетями определяют стандартные методы зондирования какого-либо устройства с целью выявления его сетевых соединений, конфигурации и общего состояния. Кроме того, протоколы позволяют модифицировать часть этой информации, чтобы можно было стандартизировать управление различными устройствами на сетевом уровне и осуществлять это управление с центральной станции.

Самым распространенным протоколом управления, используемым в сетях TCP/IP, является SNMP (Simple Network Management Protocol — простой протокол управления сетями). Несмотря на название, этот протокол достаточно сложный. Он определяет иерархическое пространство имен для объектов управления и методы чтения/записи данных, относящихся к этим объектам, на каждом узле иерархии. Он также определяет способ, которым управляемые сущности ("агенты") посылают сообщения о происходящих событиях ("прерывания") станциям управления.

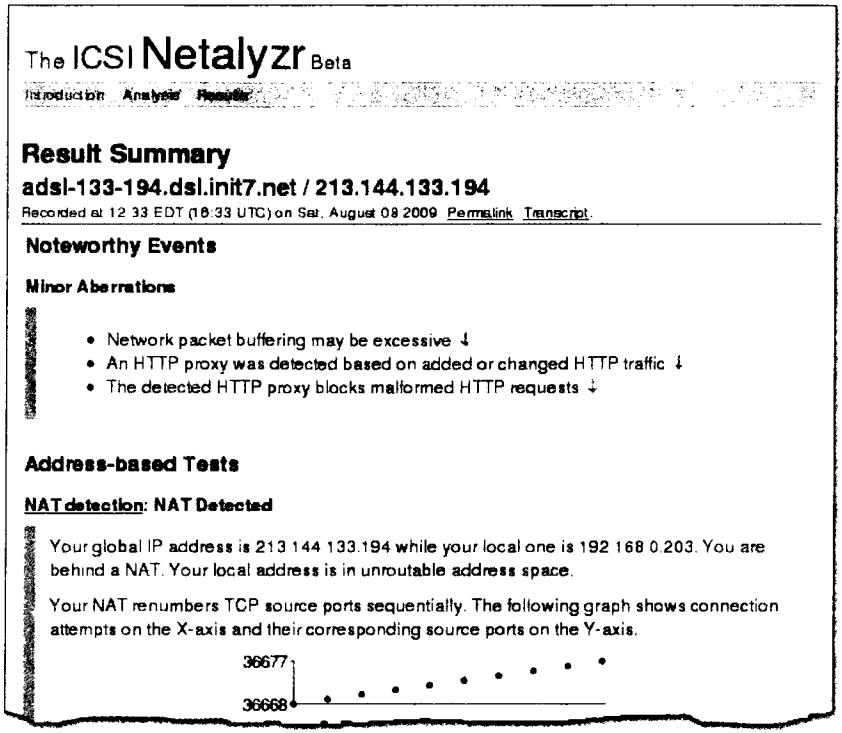



*Рис. 21.3. Отчет службы Netalyzr*

Протокол SNMP действительно является простым; самая сложная часть находится над протокольным уровнем и заключается в соглашениях по организации пространства имен и по форматированию элементов данных на узле иерархии. Протокол SNMP широко поддерживается разработчиками программного обеспечения.

Существует и ряд других стандартов управления сетями. Многие из них созданы организацией DMTF (Distributed Management Task Force — рабочая группа по разработке распределенных систем управления), которая реализовала такие концепции, как WBEM (Web-Based Enterprise Management — система управления предприятием на основе веб-технологий), DMI (Desktop Management Interface — интерфейс управления компьютером) и CIM (Conceptual Interface Model — концептуальная модель интерфейса). Некоторые из этих концепций, в частности DMI, приняты рядом известных фирм-производителей и могут служить полезным дополнением (а иногда и заменой) протоколу SNMP. В настоящее время, однако, основным средством управления сетями остается все же SNMP.

Поскольку SNMP — абстрактный протокол, для его использования нужны программа-сервер ("агент") и программа-клиент ("менеджер"). (Как ни странно, серверная сторона SNMP является управляемым элементом, а клиентская — управляющим.) Клиентом может быть как простая утилита, работающая в режиме командной строки, так и выделенная станция управления, на мониторе которой в графическом виде отображается сеть вместе со всеми неполадками.

Выделенные станции управления сетью являются главной причиной существования самих протоколов управления. Большинство программных продуктов позволяет строить не только логическую, но и топографическую модель сети. Обе модели выводятся на экран, при этом постоянно отображается текущее состояние каждого сетевого компонента.

Подобно тому, как график может показать скрытый смысл, заложенный в заполненной числами странице, так и станция управления сетью способна обобщить и представить состояние крупной сети в форме, приемлемой для человека. Другим способом такую информационную сводку получить практически невозможно.

Основное преимущество протокола SNMP в том, что абсолютно все типы сетевых аппаратных средств выводятся на один уровень. Linux-системы в основном похожи друг на друга, чего нельзя сказать о маршрутизаторах, шлюзах и остальных низкоуровневых компонентах. При использовании протокола SNMP все они начинают "общаться" на одном языке и могут зондироваться, сбрасываться в начальное состояние и конфигурироваться с центральной станции. Очень удобно, когда есть один согласованный интерфейс, применимый ко всем сетевым устройствам.

# 21.10. SNMP: простой протокол управления сетями

Когда в начале 90-х годов протокол SNMP впервые появился на рынке, возникла своеобразная "золотая лихорадка". Сотни компаний выпустили собственные пакеты управления по протоколу SNMP. Кроме того, многие поставщики аппаратных и программных средств стали включать в свои продукты SNMP-агенты. Большинство компонентов сетевого аппаратного обеспечения, предназначенных для производства (а не для домашнего использования), в настоящее время содержат агенты SNMP.

Прежде чем погрузиться в дебри SNMP, заметим, что терминология, употребляемая в этой области, одна из самых бестолковых и невразумительных. Стандартные названия объектов и концепций SNMP будут активно уводить читателей от понимания их назначения, так что запаситесь терпением. У людей, ответственных за такое положение вещей, следует отобрать клавиатуру.

## Структура протокола SNMP

В SNMP данные организованы иерархически, причем структура иерархии жестко определена. Это делает пространство имен универсальным и расширяемым, по крайней мере теоретически. Большие его области оставлены для перспективного использования; дополнения, создаваемые поставщиками операционных систем, локализуются в определенных диапазонах во избежание конфликтов. Для формирования пространства имен используются так называемые *базы управляющей информации* (Management Information Base, MIB). Это структурированные текстовые файлы, которые содержат описания данных, доступных по протоколу SNMP. Ссылки на конкретные переменные, описываемые в базе данных, называются *идентификаторами объектов* (Object Identifier, OID).

В переводе на человеческий язык это означает всего лишь, что протокол SNMP определяет иерархическое пространство имен переменных, хранящих "интересные" параметры системы.

Иерархия SNMP напоминает иерархию имен файловой системы. Только в качестве символа-разделителя используется точка, а каждому узлу иерархии присваивается не имя, а номер. Для облегчения ссылок узлам присваиваются также текстовые имена, но схема именования выходит за рамки самой иерархии и определяется на высоком уровне. (Это похоже на связь между именами компьютеров и их IP-адресами.)

К примеру, идентификатор OID, соответствующий показателю общего времени работы системы, выглядит так: 1.3.6.1.2.1.1.3. В текстовом виде его можно записать следующим образом.

iso.org.dod.internet.mgmt.mib-2.system.sysUpTime

Верхние уровни иерархии SNMP носят искусственный характер и обычно не содержат полезных данных. По сути, интересная информация начинает появляться только на уровне `iso.org.dod.internet.mgmt` (в числовом виде — 1.3.6.1.2).

Основная административная база данных SNMP для протоколов TCP/IP (MIB-I) обеспечивает доступ к наиболее важным управляющим данным: информации о системе, ее интерфейсах, преобразовании адресов и протоколах (IP, ICMP, TCP, UDP и др.). В документе RFC1213 описана новая, более полная версия этой базы: MIB-II. Большинство SNMP-серверов поддерживает базу MIB-II. В табл. 21.1 приведена выборка узлов иерархии из пространства имен MIB-II.

**Таблица 21.1. Некоторые идентификаторы из базы MIB-II**

| OID[a] | Тип | Содержание |
|---|---|---|
| system.sysDescr | строка | Информация о системе: производитель, модель, тип ОС и т.д. |
| system.sysLocation | строка | Физическое местонахождение компьютера |
| system.sysContact | строка | Информация о владельце компьютера |
| system.sysName | строка | Имя системы (обычно это полное DNS-имя) |
| interfaces.ifNumber | целое | Количество имеющихся сетевых интерфейсов |
| interfaces.ifTable | таблица | Таблица с информацией о каждом интерфейсе |
| ip.ipForwarding | целое | 1, если система является шлюзом, иначе 2 |
| ip.ipAddrTable | таблица | Таблица данных IP-адресации (маски и т.д.) |
| ip.ipRouteTable | таблица | Системная таблица маршрутизации |
| icmp.icmpInRedirects | целое | Число полученных директив переадресации протокола ICMP |
| icmp.icmpInEchos | целое | Число полученных эхо-пакетов |
| tcp.tcpConnTable | таблица | Таблица текущих TCP-соединений |
| udp.udpTable | таблица | Таблица с информацией о UDP-сокетах, через которые серверы ожидают прием запросов |

[a] Относительно узла iso.org.dod.internet.mgmt.mib-2.

Помимо основной административной базы данных, существуют базы для различных типов аппаратных интерфейсов и протоколов. Имеются также базы данных по отдельным поставщикам и конкретным изделиям.

База MIB — это лишь соглашение об именовании управляющих данных. Для того чтобы эта схема заработала, ее необходимо подкрепить программой-агентом, которая будет обеспечивать соответствие содержимого SNMP-переменных текущему состоянию устройства. Код для основной базы MIB (в настоящее время MIB-II) поставляется с большинством SNMP-агентов Linux. Некоторые агенты разрешают подключать дополнительные базы данных.

Агенты SNMP — сложные существа, имеющие общие проблемы с безопасностью. Не следует полагаться на агентов, которых поставщики предоставляют в готовом виде. Разумнее скомпилировать и инсталлировать текущую версию агента NET-SNMP (см. далее).

## Операции протокола SNMP

В пространстве имен SNMP определено всего четыре базовые операции: **get** (прочитать), **get-next** (прочитать следующий), **set** (записать) и **trap** (прерывание).

Операции **get** и **set** — это базовые операции чтения и записи данных на узле иерархии, который определяется конкретным значением OID. Операция **get-next** используется для последовательного прохода по базе MIB, а также для чтения таблиц.

*Прерывание* (операция **trap**) — это неожиданное уведомление клиента о том, что на сервере произошло нестандартное событие. Определен ряд стандартных прерываний, в том числе уведомления вида "я только что включился", сообщения об отказах и восстановлении сетевых каналов, а также сообщения, связанные с различными проблемами маршрутизации и аутентификации. Широко применяются и нестандартные прерывания, например для отслеживания значений требуемых SNMP-переменных. Если эти значения выходят за границы установленного диапазона, выдается соответствующее сообщение. Способ определения получателей таких сообщений зависит от реализации агента.

Поскольку сообщения SNMP потенциально могут изменять информацию о конфигурации компьютера, необходим какой-то механизм защиты. Простейшая защита основана на концепции *группового имени* (community name). Для тех, кто не страдает косноязычием, поясним: на языке разработчиков протокола это синоним слова "пароль". Доступу только для чтения соответствует один пароль, простите, групповое имя, а доступу для записи — другой.

Несмотря на то что многие организации продолжают использовать общепринятую строковую аутентификацию имени и пароля, версия 3 протокола SNMP включает методы управления доступом с высокой степенью безопасности. Их настройка требует дополнительной работы, но снижение риска стоит этого. Если по каким-то причинам вы не можете использовать версию 3 протокола SNMP, по крайней мере выберите строку имени и пароля, которую сложно угадать.

## RMON: база MIB для дистанционного мониторинга

База RMON (remote monitoring — дистанционный мониторинг) накапливает данные о общих характеристиках сети (т.е. тех, которые не относятся к конкретному устройству). Сетевые анализаторы, или "зонды", могут собирать информацию о загруженности и производительности сети. Полезные данные группируются, подвергаются статистической обработке и доставляются на центральную станцию управления для анализа и графического воспроизведения. Многие зонды имеют буферы для перехваченных пакетов и могут работать подобно утилите **tcpdump**.

База RMON формально описана в документе RFC1757, который был принят в качестве чернового стандарта в 1995 году. Она разделяется на девять "групп RMON". Каждая группа хранит собственный набор статистических данных. Если сеть достаточно большая и имеет много глобальных соединений, необходимо рассмотреть возможность приобретения зондов для снижения SNMP-трафика через глобальные соединения. При наличии итоговых статистических данных отпадет необходимость в дистанционном сборе первичных данных. Многие коммутаторы и маршрутизаторы поддерживают базы RMON и хранят в них собственные статистические данные.

# 21.11. Агент NET-SNMP

После публикации первой версии стандарта SNMP в университете Карнеги-Меллона (Carnegie Mellon University) и Массачусетсском технологическом институте (MIT) были разработаны реализации протокола. Университетская реализация протокола SNMP была более полной и быстро стала стандартом де-факто. После прекращения разработки SNMP-продуктов в университете Карнеги-Меллона работу продолжили специалисты Калифорнийского университета в Дэвисе. Когда код системы стабилизировался, она была передана для дальнейшего сопровождения компании SourceForge. В настоящее время пакет называется NET-SNMP. Этот пакет теперь считается официальной бесплатной реализацией протокола SNMP для систем UNIX и Linux. Последнюю версию пакета можно получить на веб-узле `net-snmp.sourceforge.net`.

В состав пакета входят SNMP-агент, несколько утилит командной строки и даже библиотека для разработки SNMP-приложений. Ниже рассмотрим работу самого агента, а об утилитах поговорим чуть позже.

Как и в других реализациях протокола, агент NET-SNMP собирает данные о локальном компьютере и предоставляет их SNMP-менеджерам по сети. По умолчанию инсталлируются базы MIB, содержащие статистические данные об использовании сетевых интерфейсов, памяти, дисков и центрального процессора. Возможности агента достаточно велики, так как он способен запустить любую Linux-команду и представить ее выходные данные в качестве SNMP-ответа. Это позволяет посредством протокола SNMP осуществлять мониторинг событий, происходящих в системе.

По умолчанию агент инсталлируется под именем **`/usr/sbin/snmpd`**. Обычно он запускается на этапе начальной загрузки системы и считывает параметры конфигурации из файлов, находящихся в каталоге **`/etc/snmp`**. Наиболее важный из этих файлов называется **`snmpd.conf`**; он содержит большинство конфигурационных параметров и описание множества доступных методов сбора данных. При разработке пакета его авторы полагали, что пользователи будут редактировать только файл **`snmpd.local.conf`**. Но на практике приходится хотя бы один раз изменять и файл **`snmpd.conf`**, чтобы отключить те методы сбора данных, использование которых не планируется.

Сценарий **`configure`** пакета NET-SNMP позволяет задать используемый по умолчанию журнальный файл и ряд других локальных параметров. С помощью опции **`-l`**, указываемой при вызове демона **`snmpd`**, можно выбрать другой журнальный файл, а благодаря опции **`-s`** журнальные сообщения направляются в систему Syslog. Перечень наиболее важных опций демона **`snmpd`** приведен в табл. 21.2. Мы рекомендуем всегда задавать опцию **`-a`**. При поиске неисправностей удобно применять опции **`-V`**, **`-d`** и **`-D`**, которые позволяют получать более детальную информацию о происходящих в системе событиях.

**Таблица 21.2. Опции демона snmpd пакета NET-SNMP**

| Опция | Назначение |
|---|---|
| **`-l`** *файл* | Направление журнальной информации в указанный файл |
| **`-a`** | Регистрация адресов всех SNMP-соединений |
| **`-d`** | Регистрация содержимого каждого SNMP-пакета |
| **`-V`** | Включение режима подробного описания событий |
| **`-D`** | Регистрация отладочной информации |
| **`-h`** | Отображение аргументов демона **`snmpd`** |

*Окончание табл. 21.2*

| Опция | Назначение |
|---|---|
| **-H** | Отображение директив конфигурационного файла |
| **-A** | Добавление данных в журнальный файл, а не его перезапись |
| **-s** | Направление журнальной информации в систему Syslog |

Следует заметить, что существует большое количество полезных модулей, написанных на языках Perl для SNMP, Ruby и Python. Их можно найти в соответствующих репозиториях.

# 21.12. Программы управления сетями

Мы начнем этот раздел с рассмотрения простейших SNMP-инструментов: команд пакета NET-SNMP. С их помощью удобно проверять значения конкретных идентификаторов OID. Затем познакомимся с программой Cacti, которая строит графики изменения SNMP-переменных, и системой мониторинга событий Nagios. В завершение приводятся рекомендации относительно того, на что следует обращать внимание при покупке коммерческих систем.

## Команды пакета NET-SNMP

Даже если система поставляется с собственным SNMP-сервером, для полноценного администрирования понадобится скомпилировать и инсталлировать семь клиентских команд пакета NET-SNMP (табл. 21.3).

**Таблица 21.3. Команды пакета NET-SNMP**

| Команда | Назначение |
|---|---|
| **snmpdelta** | Контролирует изменения SNMP-переменных во времени |
| **snmpdf** | Контролирует изменение дискового пространства на удаленном компьютере по протоколу SNMP |
| **snmpget** | Запрашивает у агента значение SNMP-переменной |
| **snmpgetnext** | Запрашивает значение следующей переменной последовательности |
| **snmpset** | Передает агенту значение SNMP-переменной |
| **snmptable** | Запрашивает таблицу значений SNMP-переменных |
| **snmptranslate** | Осуществляет поиск идентификаторов OID и их описаний в базе MIB |
| **snmptrap** | Генерирует сообщение о прерывании |
| **snmpwalk** | Просматривает базу MIB начиная с заданного идентификатора OID |

Перечисленные команды чрезвычайно удобно использовать в сценариях. Например, часто требуется сценарий, в котором данные, полученные командой **snmpget**, каждые несколько минут сохраняются в текстовом файле. (Составить такой график позволит демон **cron**; см. главу 9.)

Примечательна также команда **snmpwalk**. Начав с указанного идентификатора OID (или, по умолчанию, с начала базы MIB), она выполняет в цикле запрос **get-next**. В результате формируется полный список доступных идентификаторов OID и их значений. Команда **snmpwalk** особенно полезна, если необходимо обнаружить новые идентификаторы и организовать их мониторинг.

Ниже приведен сокращенный пример работы команды **snmpwalk** на узле **tuva**. Строка имени и пароля имеет вид "secret813community", а флаг **-v1** означает простую аутентификацию.

```
$ snmpwalk -c secret813community -v1 tuva
SNMPv2-MIB::sysDescr.0 = STRING: Linux tuva.atrust.com 2.6.9-11.ELsmp #1
SNMPv2-MIB::sysUpTime.0 = Timeticks: (1442) 0:00:14.42
SNMPv2-MIB::sysName.0 = STRING: tuva.atrust.com
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.3 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:11:43:d9:1e:f5
IF-MIB::ifPhysAddress.3 = STRING: 0:11:43:d9:1e:f6
IF-MIB::ifInOctets.1 = Counter32: 2605613514
IF-MIB::ifInOctets.2 = Counter32: 1543105654
IF-MIB::ifInOctets.3 = Counter32: 46312345
IF-MIB::ifInUcastPkts.1 = Counter32: 389536156
IF-MIB::ifInUcastPkts.2 = Counter32: 892959265
IF-MIB::ifInUcastPkts.3 = Counter32: 7712325
...
```

В этом примере команда отобразила основную информацию о системе и статистические данные о работе ее сетевых интерфейсов: `lo0`, `eth0` и `eth1`. В зависимости от поддерживаемых агентом баз MIB, вывод команды **snmpwalk** может содержать сотни строк.

## Сбор и накопление данных протокола SNMP

Сетевые данные лучше всего оценивать визуально и с учетом временной перспективы. Важно иметь возможность отслеживать и изображать графически показатели производительности, но выбор конкретного программного обеспечения не является критически важным.

Одним из наиболее популярных пакетов для сбора и графического отображения данных о протоколе SNMP был пакет MRTG, написанный Тоби Отикером. Большая часть пакета MRTG была написана на языке Perl. Он может регулярно запускаться демоном **cron** и собирать данные от любого источника SNMP. Каждый раз при запуске программы записываются новые данные и создаются новые графические изображения.

Еще одним полезным инструментом в этой области является пакет RRDtool, также написанный Тоби Ойтикером. Это набор прикладных программ для хранения и графического отображения показателей производительности. Все ведущие инструменты для мониторинга основаны на пакете RRDtool, включая нашего фаворита — программу Cacti.

Программа Cacti, доступная на сайте `cacti.net`, обладает несколькими привлекательными функциональными возможностями. Во-первых, используя пакет RRDtool как основу, она собирает данные мониторинга в статических базах данных, не требующих технического обслуживания. Программа Cacti собирает столько данных, сколько требуется для построения желаемых графиков. Например, программа Cacti может сохранять по одной выборке каждую минуту в течение дня, каждый час в течение недели и каждую неделю в течение года. Эта схема позволяет собирать важную информацию без хранения незначительных деталей и без затрат времени на управление базами данных.

Во-вторых, программа Cacti может записывать и изображать в графическом виде любую переменную SNMP, а также любые другие показатели производительности. Пользователь может свободно собирать любые данные, которые его интересуют. В сочетании с агентом NET-SNMP программа Cacti генерирует данные об истории функционирования практически любой системы или сетевого ресурса.

На рис. 21.4 приведено несколько примеров графиков, построенных с помощью программы Cacti. Эти графики демонстрируют среднюю загрузку сервера в течение нескольких недель, а также суточный трафик сетевого интерфейса.

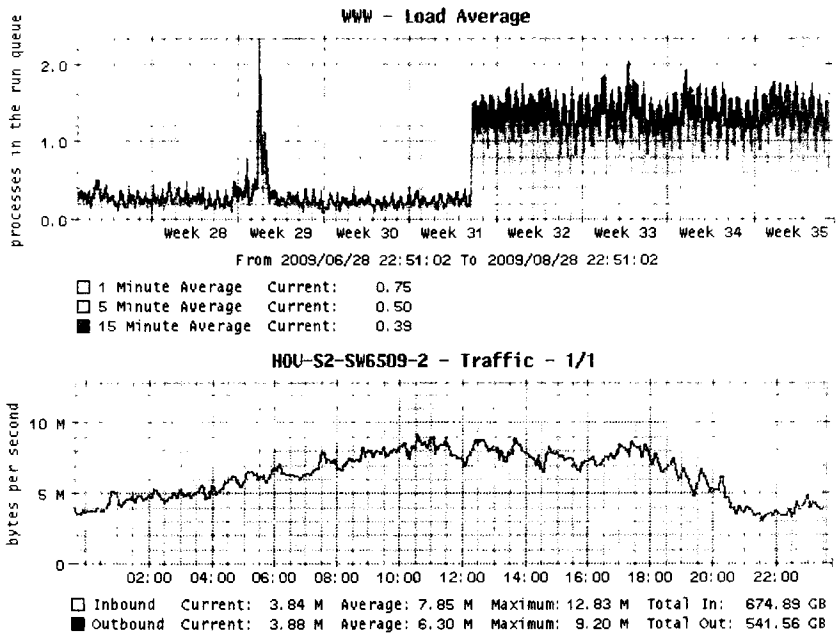



*Рис. 21.4. Пример графика, построенного программой Cacti*

Программа Cacti имеет простую конфигурацию, использующую средства веб, а также встроенные возможности пакета RRDtool, например небольшой объем технического обслуживания и прекрасные графические функции. Посетите домашнюю страницу пакета RRDtool `rrdtool.org` и обратите внимание на ссылки на текущие версии пакетов RRDtool и Cacti, а также десятки других инструментов для мониторинга.

## Nagios: событийная служба мониторинга

Служба Nagios специализируется на составлении отчетов об ошибках в реальном времени. Она содержит набор сценариев для обеспечения работы средств мониторинга всех форм и размеров, а также имеет широкие функциональные возможности для мониторинга протокола SNMP. Возможно, ее главным преимуществом является модульная, легко настраиваемая система конфигурации, позволяющая записывать пользовательские сценарии мониторинга любых мыслимых показателей. Несмотря на то что служба Nagios не поможет определить, насколько повысилась пропускная способность сети за последний месяц, она может оповестить вас, когда ваш веб-сервер выйдет из строя.

Дистрибутивный пакет Nagios содержит надстройки для слежения за типичными проблемными точками. Пользователь может создавать новые мониторы на языке Perl или даже на языке C, если чувствует в себе силы. Для уведомления об ошибках дистрибутивный пакет может посылать электронные сообщения, генерировать веб-отчеты и использовать модемы коммутируемых линий передачи. Как и надстройки мониторинга, все они легко настраиваются.

Кроме отсылки уведомлений об отключении систем в реальном времени, служба Nagios хранит архив ретроспективных данных. Она имеет несколько мощных интерфейсов для генерации графических отчетов о трендах доступности и производительности работы сети. Многие организации используют службу Nagios для измерения степени соответствия показателей заданным уровням обслуживания; на рис. 21.5 показана доступность DNS-сервера.

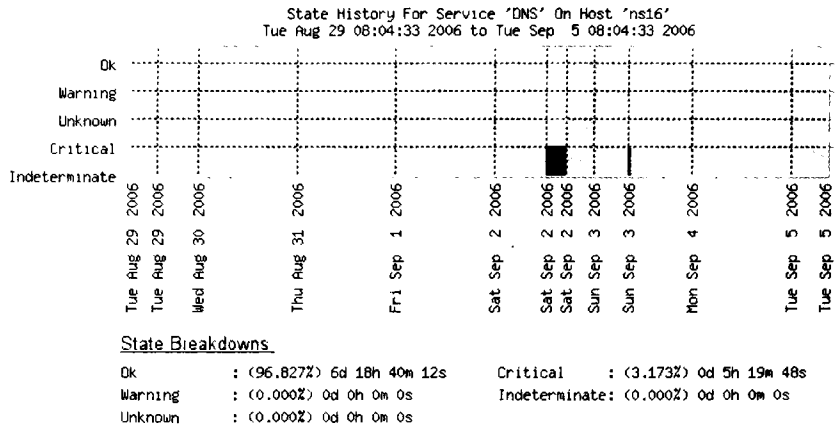



*Рис. 21.5. График показателя доступности сервера, построенный службой Nagios*

Система Nagios очень хорошо работает с сетями, состоящими менее чем из тысячи компьютеров и устройств. Ее легко настраивать и расширять. Кроме того, она имеет множество мощных функциональных возможностей, таких как резервирование, удаленный мониторинг и эскалация уведомлений. Если вы не можете себе позволить приобрести коммерческое программное обеспечение для мониторинга сети, мы настоятельно рекомендуем обратить внимание на систему Nagios. Более подробную информацию можно найти на сайте nagios.org.

## Совершенный пакет для мониторинга: поиски продолжаются

При исследовании пакетов для управления сетями в ходе редактирования нашей книги мы обнаружили, что в этой области на протяжении последнего десятилетия происходит активная деятельность. Однако большинство пакетов для регистрации и графического отображения информации по-прежнему используют пакет RRDtool. Высококачественный стандарт наподобие редакторов **vi** или **emacs** пока не появился.

Две хорошо финансируемые компании, использующие модель "открытый источник плюс" (Ground-Work Open Source и Zenoss), приступили к разработке пакетов для управ-

ления сетями, подкрепив ее серьезной рекламой и отшлифованными интерфейсами. В этой традиционной области программного обеспечения их конкурентами стали пакеты Munin (`munin.projects.linpro.no`) и **`collectd`** (`collectd.org`).

Пакет Munin особенно популярен в скандинавских странах. Он основан на остроумной архитектуре, в которой есть надстройки не только для сбора данных, но и для уведомления системы о способе их представления.

Пакет **`collectd`** написан на языке C, чтобы обеспечить высокое быстродействие и мобильность программ. Он работает даже на крошечных системах, не имея проблем с производительностью и не требуя дополнительной поддержки. Во время редактирования книги пакет **`collectd`** поставлялся с 70 надстройками для сбора данных.

## Коммерческие системы сетевого управления

Сотни компаний продают программное обеспечение для управления сетями, и каждую неделю на рынке появляются новые продукты. Мы не будем рекомендовать конкретные продукты (за время подготовки книги к печати все может измениться), а попробуем сформулировать требования к системе управления сетью.

**Возможность получения различных видов данных.** Для систем управления сетью важно уметь получать данные не только по протоколу SNMP. Многие программы позволяют принимать данные от большинства других сетевых служб, например выполнять SQL-запросы, обращаться к DNS и веб-серверам.

**Качество пользовательского интерфейса.** Дорогостоящие системы обычно позволяют выбирать между имеющимся графическим интерфейсом или веб-интерфейсом. Программы, разработчики которых учитывают конъюнктуру рынка, поддерживают XML-шаблоны для форматирования данных. Важно, чтобы интерфейс позволял представлять информацию в наглядном и понятном виде.

**Стоимость.** Некоторые программы имеют завышенную цену. Пакет OpenView компании Hewlett-Packard является одним из наиболее дорогостоящих, но в то же время одним из самых распространенных. Для многих компаний престижно, что их сети управляются высококачественной коммерческой системой. Если для вашей организации это не так важно, обратите внимание на бесплатное программное обеспечение, в частности программы Cacti и Nagios.

**Автоматическое обнаружение узлов.** Многие системы обладают способностью "обнаруживать" сеть. Посылая широковещательные эхо-пакеты, выполняя SNMP-запросы, просматривая таблицы маршрутизации и обращаясь к службе DNS, они могут идентифицировать все компьютеры и устройства в сети. Как правило, данная функция реализована достаточно хорошо, но не всегда удается получить точные результаты в сложных сетях или сетях с брандмауэрами.

**Средства отчета.** Многие программы способны посылать предупреждения по электронной почте, выдавать сообщения на пейджеры и автоматически генерировать уведомления для популярных систем обнаружения ошибок. Убедитесь в том, что выбранная платформа позволяет гибко настраивать систему отчетности. Никто не знает, какие устройства появятся в ближайшие годы.

**Возможности конфигурирования.** Некоторые производители в своих разработках пошли гораздо дальше простого мониторинга и выдачи сообщений. Их продукты позволяют управлять конфигурацией компьютеров и устройств. Например, система Cisco Works посредством специального интерфейса разрешает пользователю не только проверять состояние маршрутизатора по протоколу SNMP, но и менять его конфигурацию.

Поскольку данные о конфигурации устройства необходимы для глубокого анализа сетевых проблем, мы предполагаем, что в будущем большинство программ будет располагать средствами изменения конфигурации сетевых компонентов.

# 21.13. Протокол NetFlow: мониторинг соединений

Широко известна способность протокола SNMP регистрировать объем трафика, проходящего через интерфейс. Однако если вас интересует не только объем трафика, но и его тип и пункты назначения, протокол SNMP становится практически бесполезным. Для того чтобы выяснить дополнительные детали, можно запустить анализатор из набора инструментов системы UNIX, но эту возможность нельзя реализовать для конкретного маршрутизатора.

Для того чтобы решить эту проблему, поставщики маршрутизаторов предложили свои собственные инструменты. Самым популярным среди них стал протокол NetFlow компании Cisco.

Протокол NetFlow отслеживает каждое соединение по семи ключам: источник и IP-адрес получателя, источник и номер порта назначения, протокол (TCP, UDP и т.д.), тип службы (ToS) и логический интерфейс. Эти метаданные в сочетании с дополнительной информацией, такой как количество пакетов и байтов, можно послать любой подходящей программе для сбора данных.

Наиболее распространенными являются версии протокола NetFlow v5 и v7, которые часто объединяются, потому что они по существу одинаковы, за исключением того, что версия v7 имеет дополнительное поле (маршрутизатор источника). Версия v7 используется коммутаторами Cisco Catalyst. В настоящее время все более популярной становится версия 9. Ее шаблонная природа обеспечивает высокую гибкость.

Маршрутизатор NetFlow может посылать собранные метаданные соответствующему получателю, например пакету **cflowd**, разработанному организацией CAIDA. Если сетевое соединение сильно загружено, эта конфигурация порождает огромный объем данных, для хранения которого необходима большая дисковая память, и анализатор, способный решить поставленную задачу.

В качестве такого анализатора можно использовать пакет FlowScan, разработанный Дэйвом Плонка (Dave Plonka). К сожалению, он до сих пор не обновлен, но по-прежнему работает хорошо. Его можно найти на сайте `net.doit.wisc.edu/~plonka/FlowScan`.

## Мониторинг данных протокола NetFlow с помощью утилит nfdump и NfSen

Рассмотрим еще одну пару полезных инструментов для сбора и анализа данных протокола NetFlow — утилиты **nfdump** (`nfdump.sourceforge.net`; разработана Петером Хаагом (Peter Haag)) и **NfSen** (`nfsen.sourceforge.net`). Коллектор (**nfcapd**) записывает данные протокола NetFlow на диск для дальнейшей обработки утилитой **nfdump**.

Утилиты **nfcapd** и **nfdump** предназначены для версий v5/v7 и v9. Для поддержки протокола IPv6 следует использовать версию v9; версии 5 и 7 его не поддерживают.

Утилита **nfdump** работает почти так же, как и утилита **tcpdump** (см. раздел 21.7). Она имеет аналогичный синтаксис фильтра, адаптированный для данных протокола NetFlow. Гибкие форматы вывода позволяют настраивать внешний вид записей. Встроенные ге-

нераторы отчетов демонстрируют N активных пользователей[8] (**talkers**) вашей сети и другую полезную информацию.

Следующий (немного сжатый) отчет утилиты **nfdump** показывает, какие IP-адреса и сети составляют трафик, какие порты в настоящий момент являются наиболее активными и т.п. Параметры **-s ip/flows** запрашивают информацию о IP-адресах всех источников или пунктов назначения, упорядоченных по объему потока. Параметры **-n 10** ограничивают вывод десятью пунктами.

```
linux$ nfdump -M /data/nfsen/profiles-data/live/upstream
     -r 2009/07/28/12/nfcapd.200907281205 -n 10 -s ip/flows
Top 10 IP Addr ordered by flows:
Date first seen  Durat'n IP Addr         Flows  Pkts   Bytes   pps   bps    bpp
2009-07-28 12:02 467.596 192.168.96.92   27873  67420  3.8 M   144   67347  58
2009-07-28 12:02 462.700 192.168.96.107  18928  43878  4.7 M   94    85522  112
2009-07-28 12:02 464.443 192.168.96.198  17321  45454  3.5 M   97    63884  81
2009-07-28 12:02 454.299 172.16.152.40   11554  29093  1.3 M   64    23996  46
2009-07-28 12:02 362.586 192.168.97.203  6839   11104  1.2 M   30    28883  117
2009-07-28 12:02 393.600 172.16.220.139  4802   12883  618384  32    12568  48
2009-07-28 12:02 452.353 192.168.96.43   4477   5144   554709  11    9810   107
2009-07-28 12:02 456.306 192.168.96.88   3416   6642   697776  14    12233  105
2009-07-28 12:02 459.732 192.168.96.108  2544   25555   3.2 M  55    58478  131
2009-07-28 12:02 466.782 192.168.96.197  2143   24103   5.3 M  51    94988  229

Summary: total flows: 98290, total bytes: 311.6 M, total packets: 759205,    avg
    bps: 5.3 M, avg pps: 1623, avg bpp: 430
Time window: 2009-07-28 12:02:12 - 2009-07-28 12:09:59
Total flows processed: 98290, skipped: 0, Bytes read: 5111164
Sys: 0.310s flows/second: 317064.5 Wall: 0.327s flows/second: 300366.1
```

Поскольку данные протокола NetFlow хранятся на диске, их можно несколько раз анализировать с помощью разных наборов фильтров. Еще одна удобная функциональная возможность утилиты **nfdump** позволяет объединять входящие и исходящие потоки в один двунаправленный поток.

Утилита **NfSen** — это веб-интерфейс для данных протокола NetFlow, который дополняет утилиту **nfdump** и, следовательно, сочетает графические возможности со всеми функциями утилиты **nfdump**. Она отображает данные трех разных категорий: потоки, пакеты и байты. Утилита NfSen не только строит графики, но и позволяет просматривать их, указывать на интересующие пики графика и исследовать отдельные потоки. Для уточнения внешнего вида данных можно применять любые фильтры утилиты **nfdump**. Сочетание удобного графического пользовательского интерфейса для просмотра данных с функциями утилиты **nfdump** делает программу NfSen мощным инструментом.

Утилита NfSen позволяет сохранить пользовательский фильтр и настройки внешнего вида отчета в виде профиля, который легко использовать в будущем. Например, можно определить профиль, отслеживающий трафик для вашей зоны DMZ, веб-сервера или клиентской сети.

Профили также делают утилиту NfSen ценным инструментом для работы групп, обеспечивающих безопасность сети, поскольку они могут легко определять разновидность проблемы или сетевого трафика. Например, на рис. 21.6 показан отчет об исследовании атак на основе отказа в обслуживании типа "SYN flood".

[8] "Активный пользователь" — это термин протокола NetFlow, обозначающий устройство, создающее сетевой трафик.

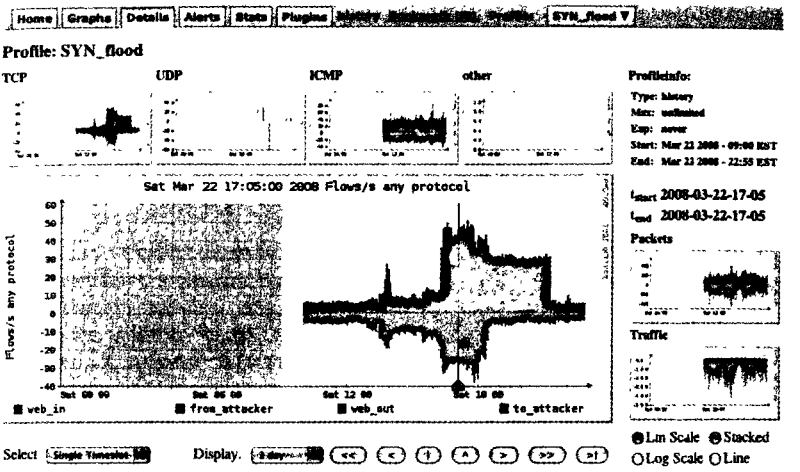



*Рис. 21.6. Профиль "SYN flood" для утилиты NfSen*

Исследование причин сбоя обычно проводится в течение ближайших часов и дней после инцидента, который его вызвал, но если данные протокола NetFlow были сохранены, то можно легко создать профиль для предшествующего периода. Этот ретроспективный анализ позволяет идентифицировать IP-адреса, вовлеченные в атаку, и отследить другие узлы, на которые она могла повлиять. Можно также настроить утилиту NfSen так, чтобы она следила за потоком в нерабочие часы и посылала сигналы тревоги в случае возникновения определенных ситуаций.

## Настройка протокола NetFlow для маршрутизатора Cisco router

Приступая к работе с протоколом NetFlow, необходимо сконфигурировать свое сетевое устройство так, чтобы оно посылало данные протокола NetFlow утилите **nfcapd**. В этом разделе мы рассмотрим конфигурацию протокола NetFlow для маршрутизатора Cisco.

Экспорт данных протокола NetFlow включен для каждого интерфейса.

```
ios# interface fastethernet 0/0
ios# ip route-cache flow
```

Для того чтобы сообщить маршрутизатору, куда посылать данные протокола NetFlow, введите следующую команду.

```
ios# ip flow-export nfcapd-hostname listen-port
```

Параметры, указанные ниже, разбивают долговременные потоки на пятиминутные сегменты. Пользователь может выбрать любой сегмент длиной от 1 до 60 минут, но их длина должна быть равной периоду ротации файлов утилиты **nfdump**, которая по умолчанию равна пяти минутам, или меньше его.

```
ios# ip flow-export version 5
ios# ip flow-cache timeout active 5
```

В маршрутизаторе Catalyst 6500/7600, кроме обычного экспорта данных протокола NetFlow, необходимо включить параметр NDE (NetFlow Data Export).

Для этого необходимо выполнить следующие команды.

```
ios# mls flow ip interface-full
ios# mls flow ipv6 interface-full
ios# mls nde sender version 5
```

На загруженном маршрутизаторе можно предусмотреть многочисленные простои маленьких потоков.

```
ios# mls aging fast time 4 threshold 2
ios# mls aging normal 32
ios# mls aging long 900
```

При этом обычная конфигурация протокола NetFlow все равно необходима, включая применение команд **ip flow ingress** или **ip route-cache flow** к каждому интерфейсу, так что потоки могут переключаться не только самим маршрутизатором, но и программным обеспечением.

Для версии NetFlow v9 конфигурация может быть еще длиннее. В зависимости от версии операционной системы , пользователь может задавать свой шаблон. С появлением технологии Flexible NetFlow (FNF), среда NetFlow стала еще более сложной.

## 21.14. Рекомендуемая литература

Википедия содержит прекрасный (хотя и неполный) обзор протокола SNMP со ссылками на документы RFC. Это хорошая отправная точка.

Mauro D., Schmidt K. *Essential SNMP (2nd Edition).*— Sebastopol, CA: O'Reilly Media, 2005.

Simple Web. *SNMP and Internet Management.* — Сайт `simpleweb.org`.

Могут оказаться полезными и перечисленные ниже документы RFC. Вместо названий мы приводим краткое описание их содержимого, так как названия представляют собой не вполне понятный набор модных словечек и жаргона SNMP.

- RFC1155 — Characteristics of the SNMP data space (data types, etc.) (Характеристики пространства имен SNMP (типы данных и т.д.)).
- RFC1156 — MIB-I definitions (description of the actual OIDs) (Определения базы MIB-I (описание идентификаторов OID)).
- RFC1157 — Simple Network Management Protocol (Протокол SNMP).
- RFC1213 — MIB-II definitions (OIDs) (Определения базы MIB-II (описание идентификаторов OID)).
- RFC3414 — User-based Security Model for SNMPv3 (Модель безопасности, определенная пользователем для протокола SNMPv3).
- RFC3415 — View-based Access Control Model for SNMPv3 (Модель управления доступом на основе представления для протокола SNMPv3).
- RFC3512 — Configuring devices with SNMP (Конфигурирование устройство в рамках протокола SNMP (лучшее обозрение)).
- RFC3584 — Practical coexistence between different SNMP versions (Практическое сосуществование разных версий протокола SNMP).
- RFC3954 — Cisco Systems NetFlow Services Export Version 9.

# 21.15. Упражнения

21.1. При поиске источника проблемы в сети, команда **netstat -rn** выдает показанные ниже результаты. В чем проблема и с помощью какой команды ее можно устранить?

```
Destination     Gateway  Genmask        Flags MSS Window irtt Iface
128.138.202.0   0.0.0.0  255.255.255.0  U      40 0          0 eth0
127.0.0.0       0.0.0.0  255.0.0.0      U      40 0          0 lo
```

21.2. ★ Напишите сценарий, который будет контролировать заданную группу компьютеров и сообщать администратору по электронной почте, если какой-нибудь компьютер не реагирует на эхо-запросы в течение заданного времени. Не задавайте список компьютеров, почтовый адрес и величину интервала времени в тексте сценария.

21.3. ★ Поэкспериментируйте с изменением сетевой маски одного из компьютеров локальной сети. Продолжит ли компьютер нормально работать? Все ли компьютеры доступны в сети? Можно ли с других компьютеров получить доступ к нему? Поддерживается ли широковещательный режим (т.е. запросы протокола ARP и пакеты DHCP)? Объясните полученные результаты. (Необходим доступ с правами суперпользователя.)

21.4. ★ Воспользуйтесь командой **traceroute** для обнаружения маршрутов в своей сети.

а) сколько переходов требуется сделать на пути к серверу?

б) присутствуют ли маршрутизаторы между компьютерами, на которых у вас есть учетные записи?

в) можно ли обнаружить узкие места в сети?

г) есть ли в сети компьютеры с несколькими IP-адресами?

21.5. ★★ Составьте базу MIB, содержащую все переменные, которые вам как администратору Linux может понадобиться запрашивать или устанавливать. Предусмотрите возможность расширения этой базы данных.

21.6. ★★ Воспользуйтесь утилитами **wireshark** или **tshark** для перехвата трафика перечисленных ниже протоколов. В случае протокола TCP укажите начальные и конечные пакеты. (Необходим доступ с правами суперпользователя.)

а) ARP;

б) эхо-запросы и эхо-ответы протокола ICMP;

в) SMTP;

г) FTP и FTP-DATA;

д) DNS;

е) Samba;

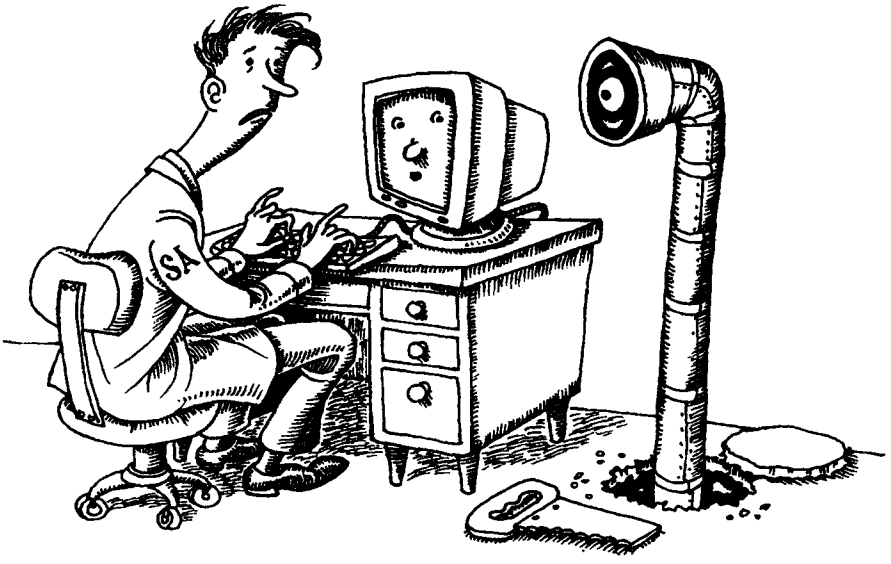
ж) SSH.

21.7. ★★ Настройте программу Cacti на отображение графика пакетов, посылаемых локальному маршрутизатору и получаемых от него. Необходимо, чтобы агент SNMP опрашивал маршрутизатор; для этого следует узнать SNMP-имя маршрутизатора.

21.8. ★★ Напишите сценарий, в котором используется пакет RRDtool для слежения за сетевым трафиком аналогично команде `netstat -i`, и создайте веб-страницу с помощью команды `rrdcgi`, демонстрирующую результаты. Если вы никогда не работали с пакетом RRDtool, это упражнение, вероятно, займет несколько часов. Дерзайте! Это стоит затраченных усилий, поскольку вы поймете, чем хороши такие сценарии, а освоение пакета RRDtool поможет вам правильно настроить его для эффективного управления сетью.

# Глава 22

# Безопасность




Несмотря на все попытки Голливуда, защита компьютерных сетей остается тяжелым и недооцененным занятием, далеким от гламура. Дисциплина системного администратора — следствие суровой необходимости; если системы UNIX и Linux обслуживают конфиденциальные данные и управляют важными процессами, мы *обязаны* их защитить.

Такая защита требует ресурсов: как времени работы системного администратора, так и специального оборудования. К сожалению, многие организации не вкладывают нужные инвестиции в эту область, пока не произойдет несчастный случай.

В ноябре 1988 года, когда Роберт Моррис-младший (Robert Morris, Jr.) выпустил в Интернет своего первого "червя", мы впервые столкнулись с реальной опасностью всемирного масштаба. До того момента Интернет переживал "век невинности". Вопросы безопасности, заботившие администраторов, звучали, в основном, так: "А что, если...?". Серьезным происшествием считалась поимка пользователя, узнавшего пароль администратора и прочитавшего чужую почту (часто не по злому умыслу, а просто для самоутверждения).

"Червь" Морриса привел к потере тысяч часов рабочего времени системных администраторов, зато бдительность пользователей Интернета возросла на порядок. Все мы просто еще раз вспомнили народную мудрость: чем выше заборы, тем добрее соседи. В результате появился ряд прекрасных инструментальных средств для системных администраторов (и была основана организация, предназначенная для борьбы с сетевым вредительством).

В настоящее время взлом систем безопасности стал распространенным явлением. Как показало совместное исследование Института компьютерных наук (CSI — Computer Science Intitute) и ФБР "Computer Crime and Security Survey"[1], проведенное в 2008 году,

[1] Это исследование проводится каждый год, а отчет о нем публикуется на сайте `gocsi.com`.

среднегодовые потери опрошенных организаций вследствие взломов систем безопасности составляли 234 тыс. долл. Большинство крупных организаций регистрируют по крайней мере одну попытку серьезного взлома системы безопасности в год.

Решить эту проблему не так просто, как кажется. Безопасность нельзя купить в магазине или заказать в сервис-центре. Соответствующие коммерческие продукты и услуги могут быть частью предлагаемого решения для конкретной организации, но это не панацея. Достижение приемлемого уровня безопасности требует огромного терпения, постоянной бдительности, немалых знаний и упорства. Это касается не только администраторов, но и всех пользователей и управляющего персонала.

Системный администратор несет личную ответственность за безопасность системы и степень образованности пользователей в данном вопросе. Он должен знать современные технологии защиты, быть подписчиком соответствующих групп новостей и нанимать экспертов в области компьютерной безопасности, если его знаний оказывается недостаточно для решения возникшей проблемы.

## 22.1. Безопасна ли система UNIX

Конечно, нет! Ни UNIX, ни Linux, ни какая-либо другая операционная система, управляющая сетью, не являются безопасными операционными системами. Для создания абсолютно непробиваемой защиты придется изолировать[2] компьютер от всех устройств доступа (и, возможно, поместить его в специальную комнату, стены которой не пропускают электромагнитное излучение). Кто может себе это позволить?

Кое-что для повышения надежности системы, разумеется, сделать можно. Но идеальная безопасность все же недостижима, ибо в модели безопасности Linux есть несколько фундаментальных изъянов, которые невозможно преодолеть.

- Операционная система UNIX ориентирована, прежде всего, на удобство применения, что отнюдь не предполагает естественность и простоту ее защиты. Концепция этой системы заключается в обеспечении удобного манипулирования данными в сетевой многопользовательской среде.
- Программное обеспечение для системы UNIX разрабатывается большим сообществом программистов. Все они имеют разную квалификацию, по-разному относятся к своей работе и обладают различными знаниями о строении операционной системы и ее особенностях. Поэтому даже самые современные средства защиты, выпущенные с самыми благими намерениями, могут приводить к появлению новых "дыр".
- Большинство административных функций реализовано за пределами ядра, где их можно проверять и модифицировать. Хакеры имеют широкий доступ в систему.

С другой стороны, исходный код этих систем (т.е. Linux, Solaris и других) доступен каждому, и тысячи людей могут проверить каждую строку этого кода на предмет наличия ошибок. Считается, что это приводит к существенному повышению безопасности по сравнению с закрытыми операционными системами, где доступ к коду имеет лишь небольшое число разработчиков.

Многие организации не торопятся устанавливать новые версии операционной системы: либо по причине сложности локализации, либо потому, что они не заключили с поставщиком договор о сопровождении системы. Даже если производитель "заткнул"

[2] Появление беспроводных сетей привело к возникновению новых проблем безопасности. В данном контексте "изолировать" означает "отключить сеть".

маленькую дырочку в системе защиты, на устранение "лазейки" потребуется какое-то время.

Может показаться, что по мере выявления и устранения брешей безопасность операционных систем будет непрерывно повышаться, но, к сожалению, это не так. Сложность системного программного обеспечения стремительно растет, хакерская деятельность все больше приобретает черты организованной преступности, компьютеры оказываются все теснее связанными посредством Интернета. Война переходит в новые измерения, и, похоже, победителей не будет.

Запомните такую формулу:

$$\text{Безопасность} = \frac{1}{1{,}072 \times \textit{Удобство}}$$

Чем безопаснее система, тем труднее пользователям работать в ней.

## 22.2. Слабые места в системе защиты

В последующих разделах будут рассматриваться некоторые наиболее распространенные проблемы, связанные с безопасностью, и стандартные контрмеры, позволяющие избежать этих проблем. Но прежде чем погрузиться в детали, необходимо взглянуть на ситуацию в целом и определить основные источники неприятностей.

### Человеческий фактор

Пользователи (и администраторы) системы часто являются ее слабейшим звеном. Даже сейчас, когда образованность людей в области безопасности повысилась, беспечные пользователи легко распространяют конфиденциальную информацию. Никакая технология не может защитить сеть от неосторожных пользователей, поэтому системный администратор обязан информировать пользователей о существующих угрозах, и это обстоятельство станет частью системы безопасности.

Эта проблема может проявляться в разных формах. Хакеры звонят своим жертвам и представляются легальными пользователями, попавшими в трудное положение, пытаясь проникнуть в сеть. Иногда сами системные администраторы неосторожно размещают конфиденциальную информацию на открытых форумах, описывая решения сложных проблем. Физические взломы возникают, например, когда внешне легальные сотрудники отдела технического обслуживания проникают в телефонный монтажный шкаф.

Термином "фишинг" (phishing) называют попытку сбора информации от пользователей с помощью обманных электронных сообщений, мгновенных сообщений и даже сообщений SMS. От фишинга особенно трудно защититься, поскольку сообщения часто содержат информацию, которая свидетельствует о знакомстве с жертвой и создает видимость аутентичности.

Использование человеческого фактора остается мощным хакерским приемом, которому трудно противостоять. Ваша стратегия обеспечения безопасности должна предусматривать обучение новых сотрудников. Эффективно также распространять среди сотрудников информацию о телефонных атаках, физических проблемах, фишинге и правильном выборе паролей.

Может показаться, что для того, чтобы противостоять попыткам использовать человеческий фактор, можно попытаться самому спровоцировать такую атаку. Однако сначала необходимо получить разрешение от менеджера. Если у вас не будет такого разре-

шения, такие действия будут выглядеть подозрительно. Кроме того, они могут вызвать "шпиономанию" и обиду.

Многие организации считают необходимым сообщить своим пользователям, что администраторы ни при каких обстоятельствах не имеют права спрашивать у них пароли ни через электронные сообщения, ни через мгновенные сообщения, ни по телефону. О любой попытке узнать пароль с помощью этих средств следует сообщать в отдел информационных технологий.

## Ошибки в программах

За много лет в программном обеспечении (включая сторонние программы, как коммерческие, так и бесплатные) было выявлено несметное число ошибок, связанных с безопасностью. Используя незаметные программистские просчеты или контекстные зависимости, хакерам удавалось манипулировать системой по своему усмотрению.

Основной программной ошибкой, одной из самых опасных по своим последствиям, является переполнение буфера. Для хранения информации разработчики часто выделяют заранее определенный объем временной памяти, который называется буфером. Если программа не следит за размерами данных и контейнера, в котором они должны храниться, то память, смежная с выделенной областью, рискует оказаться перезаписанной. Умелые хакеры могут ввести тщательно скомпонованные данные так, чтобы они привели к краху программы или, в худшем случае, выполнили некий заданный код.

К счастью, огромное количество случаев переполнения буферов, которое наблюдалось в последние годы, научило программистов бороться с этим. Несмотря на то что переполнение буфера остается сложной проблемой, она быстро распознается и исправляется, особенно в приложениях с открытым кодом. Новейшие программные системы, такие как Java и .Net, содержат механизмы, автоматически проверяющие размеры данных и предотвращающие переполнение буфера (правда, не всегда).

Переполнение буфера является частным случаем более широкого класса проблем, связанных с безопасностью программного обеспечения, который называется уязвимостью проверки вводимых значений (input validation vulnerability). Почти все программы принимают от пользователей вводимые данные (например, аргументы командной строки или формы HTML). Если программа обрабатывает эти данные без строгой проверки соответствующего формата и содержания, то могут возникнуть неприятности. Рассмотрим следующий простой пример.

```
#!/usr/bin/perl
# Example user input validation error

open(HTMLFILE,"/var/www/html/$ARGV[0]") or die "trying\n";
while(<HTMLFILE>) {print; }
close HTMLFILE;
```

Вероятно, этот код должен выводить на печать содержимое некоего HTML-файла, находящегося в каталоге **/var/www/html**, который по умолчанию является корневым каталогом документов для веб-сервера Apache среди серверов системы Red Hat. Этот код получает от пользователя имя файла и включает его как часть аргумента команды open. Однако злоумышленник ввел в качестве аргумента строку **../../../et/passwd** и получил в качестве ответа содержимое этого файла!

Как администратор может предотвратить такой тип атак? Почти никак, по крайней мере, пока не будет найдена ошибка и создана соответствующая заплатка. По этой причине выпуск заплаток и бюллетеней, посвященных вопросам безопасности, является

важной частью работы системного администратора. Большинство дистрибутивных пакетов системы Linux включает автоматизированные утилиты для создания заплаток, такие как **yum** в системе Red Hat и **apt-get** в системе Ubuntu. Система OpenSolaris также имеет автоматизированные (и надежные) модификации, реализуемые посредством команды **pkg image-update**. Воспользуйтесь преимуществами этих утилит, чтобы уберечь ваш сайт от опасностей.

### Ошибки конфигурации

Многие компоненты программного обеспечения можно сконфигурировать в режиме полной или частичной безопасности. К сожалению, поскольку программное обеспечение должно быть полезным и не должно раздражать пользователей, по умолчанию чаще всего принят второй вариант. Хакеры вламываются в системы, иезуитски эксплуатируя функциональные возможности, которые предоставлены разработчиками для удобства пользователей: учетные записи без паролей, глобальный совместный доступ к жестким дискам и т.д., и т.п.

Один из типичных примеров уязвимости узлов является стандартная практика, когда системы семейства Linux загружаются, не требуя ввода пароля загрузчика. Загрузчик операционной системы GRUB можно настроить так, чтобы он во время инсталляции запрашивал пароль, но администраторы часто отказываются от этой возможности. Это упущение открывает возможность для физической атаки системы. Однако этот же пример демонстрирует необходимость балансирования между безопасностью и удобством. Запрос пароля означает, что при непроизвольной перезагрузке системы (например, при сбое электропитания) администратор должен физически присутствовать при повторном включении компьютера.

Одна из наиболее важных задач, связанных с обеспечением безопасности системы, — убедиться в том, что, заботясь о благополучии пользователей, вы случайно не открыли двери для хакеров. Такие проблемы проще обнаружить и устранить, чем другие, хотя их может быть очень много и не всегда очевидно, что именно следует проверять. Сканирование портов и уязвимых мест, описанное в главе, может помочь заинтересованному администратору выявить проблему еще до ее возникновения.

## 22.3. Ключевые аспекты безопасности

В этой главе рассматривается множество аспектов компьютерной безопасности. В идеале их все нужно учитывать. Системные администраторы, вероятно, должны прочитать эту главу несколько раз.

Большинство операционных систем не поставляется с уже готовой защитой. Кроме того, настройка, которая осуществляется как во время инсталляции, так и после нее, изменяет профиль безопасности новых систем. Администраторы должны укреплять новые системы, интегрировать в локальную среду и планировать их долговременную безопасную эксплуатацию.

Когда к вам придет проверка, вы должны доказать, что следовали стандартной методологии, особенно если эта методология соответствует общепринятым рекомендациям и передовым достижениям в вашей области промышленности.

Для защиты новых систем мы используем технологическую карту. Системный администратор применяет стандартные меры по укреплению безопасности системы, а администратор по вопросам безопасности проверяет, чтобы эти меры были приняты правильно, и ведет системный журнал.

## Программные "заплаты"

Оснащение операционной системы вновь разработанными программными "заплатами" является первейшей обязанностью системного администратора. Большинство систем конфигурируется так, чтобы иметь связь с репозиторием поставщика. В этом случае установка программной "заплаты" сводится всего лишь к выполнению нескольких команд. В крупных сетях могут существовать локальные репозитории, являющиеся зеркалами репозитория поставщика.

При установке программных "заплат" следует учитывать следующее.

- Нужно составить расписание инсталляции программных "заплат" и строго его придерживаться. При составлении такого расписания необходимо учитывать его влияние на пользователей. Обычно достаточно выполнять ежемесячные обновления; регулярность важнее срочности. Нельзя сосредоточиваться только на защите от вредоносных программ и игнорировать другие обновления.
- Необходимо разработать план изменений, который учитывал бы влияние каждого набора "заплат", предусматривал соответствующие этапы тестирования после инсталляции системы и описывал возможности отказа от внесенных обновлений при возникновении проблем. Этот план следует обсудить со всеми заинтересованными сторонами.
- Необходимо понимать, какие заплаты подходят к данному окружению. Системные администраторы должны подписываться на специальные списки рассылки и блоки по вопросам безопасности, создаваемые поставщиками, а также принимать участие в работе форумов по вопросам безопасности, таких как Bugtraq.

## Ненужные службы

Большинство систем поставляется с несколькими службами, которые запускаются по умолчанию. Отключите (и по возможности удалите) все ненужные службы, особенно если они реализованы в виде сетевых демонов. Для того чтобы обнаружить выполняемые службы, можно использовать команду **netstat**. Рассмотрим частичные результаты работы этой команды в системе Solaris.

```
solaris$ netstat -an | grep LISTEN
*.111                *.*  0  0  49152  0  LISTEN
*.32771              *.*  0  0  49152  0  LISTEN
*.32772              *.*  0  0  49152  0  LISTEN
*.22                 *.*  0  0  49152  0  LISTEN
*.4045               *.*  0  0  49152  0  LISTEN
```

Для обнаружения служб, использующих неизвестный порт, существует несколько приемов. В большинстве операционных систем эту задачу можно решить с помощью команд **lsof** или **fuser**. В системе Linux эти команды могут распознать идентификатор PID процесса, использующего данный порт.

```
ubuntu$ sudo fuser 22/tcp
22/tcp:       2454  8387

ubuntu$ sudo lsof -i:22
COMMAND   PID   USER  FD   TYPE   DEVICE   SIZE  NODE   NAME
sshd     2454  root  3u   IPv4   5730          TCP    *:ssh(LISTEN)
sshd     2454  root  4u   IPv6   5732          TCP    *:ssh(LISTEN)
```

Выяснив идентификатор PID, с помощью команды **ps** можно идентифицировать конкретный процесс. Если данная служба не нужна, остановите ее и убедитесь, что она не будет заново стартовать во время загрузки системы.

К сожалению, доступность команд **lsof** и **fuser** зависит от системы, и их реализации сильно отличаются друг от друга. Многим версиям этих инструментов недостает поддержки сетевых сокетов.

Если утилиты **lsof** и **fuser** недоступны (или бесполезны), можно либо проверить "хорошо известные" порты для служб в файле **/etc/service**, либо запустить утилиту **netstat** без параметра **-n**, чтобы она выполнила эту проверку за вас.

Некоторым сетевым протоколам присущ риск, которому подвергается безопасность системы. В этом случае она остается почти все время уязвимой. Например, программы FTP, Telnet и BSD "r" (**rcp**, **rlogin** и **rsh**) используют ненадежную аутентификацию и ненадежные методы передачи данных. Они должны быть отключены во всех системах и заменены более безопасными программами, например программой SSH.

## Удаленная регистрация событий

🕮 Протокол syslog описан в главе 11.

Механизм syslog направляет регистрационную информацию в файлы, списки пользователей и другие узлы сети. Попробуйте настроить узел, отвечающий за безопасность, так, чтобы он действовал как центральная регистрационная машина, анализирующая полученные события и выполняющая соответствующие действия. Единый централизованный регистратор событий может получать сообщения от разных устройств и предупреждать администраторов о важных событиях. Удаленная регистрация также предотвращает перезапись или очистку регистрационных файлов во взломанных системах.

Большинство систем поставляется сконфигурированными так, что механизм syslog включен по умолчанию, но для настройки удаленной регистрации необходимо выполнить дополнительное конфигурирование.

## Резервные копии

Регулярное резервное копирование является важной частью системы безопасности любой сети. Оно входит в так называемую триаду ЦРУ и относится к категории "доступность" (см. раздел 22.15). Убедитесь, что все части системы регулярно копируются и что эта информация хранится за пределами сайта. Если произойдет серьезный инцидент, связанный с безопасностью сети, вы сможете воспользоваться "надежным" источником информации и восстановить работу.

Резервное копирование само по себе также может создавать угрозу безопасности. Украденная коллекция магнитных лент может уничтожить всю систему безопасности. Для хранения магнитных лент используйте огнеупорные сейфы, которые могут противостоять взлому, и применяйте шифрование. Рассматривая возможность заключения контракта на хранение информации, попросите разрешения совершить экскурсию и лично убедиться в надежности хранилища.

🕮 Резервное копирование подробно описывается в главе 10.

## Вирусы и черви

Системы UNIX и Linux имеют иммунитет от вирусов. Существует всего несколько вирусов (большинство из них носит чисто академический характер), которые могут за-

разить системы UNIX и Linux, но ни один из них не способен нанести серьезный урон, в отличие от среды Windows, где это стало частым явлением. Тем не менее этот факт не снижает озабоченности поставщиков антивирусных программ — разумеется, если вы покупаете их продукт по специальной цене.

Точная причина отсутствия вредоносных программ неясна. Некоторые утверждают, что система UNIX просто занимает слишком незначительную долю рынка настольных систем и поэтому не интересует авторов вирусов. Другие настаивают на том, что среда с контролем доступа, существующая в системе UNIX, ограничивает размер урона от самораспространяющихся червей или вирусов.

Последний аргумент заслуживает внимания. Поскольку система UNIX ограничивает доступ исполняемых модулей на файловом уровне, непривилегированные пользователи не могут инфицировать остальную среду. Если код вируса был запущен не из корня, его вред будет значительно ограничен. Следовательно, не нужно использовать корневую учетную запись для повседневной деятельности.

🕮 Более подробная информация о сканировании содержимого электронных сообщений приведена в главе 20.

Как ни странно, одна из причин внедрения антивирусного программного обеспечения на серверах UNIX — стремление защитить работающие под управлением Windows компьютеры, существующие в вашей сети, от Windows-ориентированных вирусов. Почтовый сервер может сканировать входящую электронную почту на вирусы, а файловый сервер в поисках инфекции может сканировать общие файлы. Однако это решение должно быть дополнительным по отношению к антивирусной защите настольных систем, а не основным.

Программа ClamAV, написанная Томашем Коймом (Tomasz Kojm), — это популярная свободно распространяемая антивирусная программа для систем UNIX и Linux. Она широко используется в качестве полноценного антивирусного обеспечения, которое хранит сигнатуры тысяч вирусов. Новейшую версию этой программы можно загрузить с сайта `clamav.net`.

## Троянские программы

Это программы, которые не выглядят программами. Примером является программа **`turkey`**, распространявшаяся в сети Usenet много лет тому назад. Эта программа утверждала, что способна нарисовать индейку на экране компьютера, но на самом деле удаляла файлы из рабочего каталога.

Фрагменты троянских программ встречаются в основных пакетах программного обеспечения до сих пор. Авторы программ **`sendmail`**, **`tcpdump`**, OpenSSH и nterBase даже опубликовали советы, касающиеся вредоносных программ, внедренных в их программное обеспечение. Эти троянские программы обычно внедряют вредоносный код, который позволяет хакерам проникать в систему. К счастью, большинство поставщиков регулярно исправляют свои ошибки и публикуют соответствующие рекомендации раз в одну-две недели. Следите за списками рассылок по вопросам безопасности, посвященными всем сетевым программным пакетам, которые выполняются на ваших узлах.

Несмотря на большое количество проблем, возникших в области безопасности систем UNIX за последние несколько лет, следует отметить малое количество инцидентов, связанных с троянскими программами. В основном, это объясняется скоростью коммуникаций через Интернет. Очевидные проблемы с безопасностью быстро выявляются и широко обсуждаются. Вредоносные пакеты не задерживаются надолго на хорошо известных интернет-серверах.

Вы можете быть уверены, что любое выявленное вредоносное программное обеспечение вызовет в Интернете широкое волнение. Поищите в системе Google имя программного пакета, прежде чем его инсталлировать, и посмотрите, нет ли на первой странице неприятных сообщений о нем.

## Руткиты

Умелые хакеры пытаются скрывать свои следы и избегать разоблачения. Часто они рассчитывают продолжить использование вашей системы для нелегального распространения программного обеспечения, зондирования других сетей или запуска атаки против других сетей. Для того чтобы оставаться незамеченными, они часто используют "руткиты" ("rootkits"). Троянские программы, внедренные в файлы, созданные фирмой Sony, используют возможности руткитов, чтобы скрыть себя от пользователей.

Руткиты — это программы и заплатки, скрывающие важную системную информацию, например процесс, диск или сетевую активность. Они имеют много обличий и разную степень сложности — от простых замен приложения (как взломанные версии утилит **ls** и **ps**) до модулей ядра, которые практически невозможно выявить.

Для эффективного мониторинга системы и выявления внедренных руткитов существует специальное программное обеспечение, например OSSEC. Кроме того, разработаны сценарии распознавания руткитов (такие, как **chkrootkit**, `chkrootkit.org`), сканирующие систему в поисках известных руткитов.

Несмотря на существование программ, позволяющих системным администраторам удалять руткиты из взломанных систем, время, которое они вынуждены затрачивать на очистку систем, можно было бы сэкономить, сохранив данные, переформатировав диск и начав работу с нуля. Большинство современных руткитов знают о существовании программ для их удаления и пытаются оказывать сопротивление.

## Фильтрация пакетов

Если система подключается к сети, где есть выход в Интернет, *необходимо*, чтобы между этой системой и внешним миром стоял брандмауэр либо маршрутизатор, фильтрующий пакеты. В качестве альтернативы можно включить фильтрацию пакетов в ядре (описывается в разделе 22.11). Какой бы ни была реализация, фильтр должен пропускать через себя трафик только важнейших служб, выполняющих "полезную" работу в сети.

## Пароли

Все мы любим простые правила. Вот одно из них: у *каждой* учетной записи должен быть пароль, который трудно угадать. Но даже самые лучшие пароли нельзя передавать через Интернет в текстовом виде. Поэтому, если в системе допускается удаленная регистрация, следует применять SSH или какой-нибудь другой механизм аутентификации (см. раздел 22.10).

## Бдительность

Для того чтобы быть уверенным в безопасности системы, следите за ее состоянием, сетевыми соединениями, таблицей процессов. Делать это нужно регулярно (желательно каждый день). Проблема всегда начинается с малого, а затем нарастает, как снежный ком, так что чем раньше будет обнаружена аномалия, тем меньшим окажется ущерб.

## Общие принципы защиты

Эффективная система безопасности должна базироваться на здравом смысле. Она во многом напоминает борьбу с мышами в доме. Вот ряд правил, которые при этом следует соблюдать.

Этими же правилами (в компьютерном варианте) можно с успехом руководствоваться при организации защиты Linux-систем. Вот как они звучат применительно к Linux.

- Не оставляйте без присмотра файлы, которые могут представлять интерес для хакеров и не в меру любопытных сослуживцев. Коммерческие тайны, персональные досье, бухгалтерские ведомости, результаты выборов и так далее — за всем этим нужен присмотр. Гораздо надежнее зашифровать данные, чем просто пытаться предотвратить к ним несанкционированный доступ.
- В организации должен существовать порядок работы с конфиденциальной информацией. Некоторые рекомендации по этому поводу даны в главе 32.
- Старайтесь, чтобы в системе не было мест, где хакеры могли бы закрепиться. Хакеры часто вламываются в одну систему, а затем используют ее как базу для взлома других систем. Иногда хакеры используют вашу сеть для сокрытия своих следов во время атаки реальной цели. Уязвимые службы с открытым доступом, каталоги, доступные для записи по протоколу FTP в анонимном режиме, групповые учетные записи, учетные записи с плохо подобранными паролями — вот основные уязвимые места.
- Устанавливайте ловушки для обнаружения вторжений или попыток вторжения. Такие инструменты, как OSSEC, Bro, Snort и John the Ripper (описаны в разделе 22.8), помогут предупредить возможные проблемы.
- Следите за отчетами, которые генерируются этими утилитами. Незначительная проблема, проигнорированная в отчете, к моменту получения следующего отчета может перерасти в катастрофу.
- Учитесь защищать системы своими силами. Применяйте традиционные технологии, обучайте пользователей, руководствуйтесь, в конце концов, здравым смыслом. В случае необходимости приглашайте специалистов со стороны, но при этом тщательно контролируйте их работу.
- Постоянно следите за тем, не появились ли отклонения от нормального хода работы системы. Обращайте внимание на все необычное, например на непонятные журнальные сообщения или изменение характера использования какой-либо учетной записи (резкий рост активности, работа в необычное время, использование учетной записи во время отпуска ее владельца).

# 22.4. Пароли и учетные записи пользователей

Очень часто источником неприятностей является плохое управление паролями. По умолчанию в файлах `/etc/passwd` и `/etc/shadow` содержатся данные о том, кто может входить в систему и что он при этом имеет право в ней делать. Эти файлы представляют собой передовую линию защиты системы от захватчиков. Их нужно вести с особой тщательностью, стараясь не допускать ошибок и не загромождать файлы устаревшими данными.

🕮 Подробнее о файле **passwd** см. раздел 7.1.

Система UNIX позволяет пользователям выбирать свои собственные пароли, но, несмотря на то, что это удобно, из-за этого возникает множество проблем, связанных с безопасностью. Когда вы даете пользователям их регистрационные имена, должны также инструктировать их о правилах выбора паролей. Рекомендуется выбирать пароли не менее чем из восьми символов, среди которых должны быть цифры, знаки препинания, а также прописные и строчные буквы. Бессмысленные сочетания символов, слогов, первые буквы слов легко запоминаемой фразы — вот самые лучшие пароли. При этом легко запоминаемая фраза не должна быть одной из широко распространенных. Лучше придумать свою собственную. Совет по выбору фраз приводился в разделе 4.3.

Важно постоянно проверять (желательно ежедневно), чтобы каждая регистрационная запись имела пароль. Записи в файле **/etc/shadow**, описывающие псевдопользователей, таких как "демон", которые владеют файлами, но никогда не регистрируются, должны быть отмечены звездочками или знаком восклицания в поле зашифрованного пароля. Эти символы не совпадают ни с одним паролем и тем самым предотвращают использование данной учетной записи.

В организациях, использующих централизованную схему аутентификации, такую как LDAP или Active Directory, используется та же логика. Они требуют сложных паролей и блокируют учетные записи после нескольких неудачных попыток регистрации.

## Устаревание паролей

В большинстве систем, использующих теневые пароли, можно реализовать механизм так называемого устаревания паролей, при котором пользователей заставляют периодически менять пароли. На первый взгляд, это хорошая идея, однако ее практическая реализация влечет за собой определенные проблемы. Не всякому пользователю по душе периодически менять пароль, поскольку это сопряжено с запоминанием нового пароля. Обычно для пароля выбирается простое слово, которое легко вводится и запоминается, и когда приходит время замены, многие пользователи, не желая себя утруждать, опять берут предыдущий пароль. Таким образом, дискредитируется сама идея. Модули PAM могут помочь выбрать сильные пароли, чтобы избежать описанной выше ситуации (раздел 22.5).

В системе Linux процессом устаревания паролей управляет программа **chage**. С ее помощью администраторы могут задавать минимальное и максимальное количество изменений пароля, дату истечения срока действия пароля, количество дней до наступления даты истечения срока действия пароля, когда следует заблаговременно предупредить пользователя, количество дней простоя, в течение которых учетные записи остаются заблокированными, и другие параметры. Следующая команда задает минимальное количество дней между изменениями пароля равным 2, максимальное количество изменений пароля равным 90, дату истечения срока действия пароля равной 31 июля 2010 года, а также то, что пользователя следует предупредить об истечении срока действия пароля за 14 дней.

```
$ sudo chage -m 2 -M 90 -E 2010-07-31 -W 14 ben
```

🕮 Более подробно процедура настройки параметров учетных записей описана в главе 7.

Другие системы иначе реализуют механизм устаревания паролей, обычно не так детально. В системе Solaris параметры механизма устаревания паролей задаются в файле **/etc/password**. Устареванием паролей в системах семейства HP-UX управляет утилита **smc**, а в системе AIX оно настраивается в файле **/etc/security/user**.

## Групповые и совместно используемые учетные записи

Опасно, если учетная запись используется несколькими людьми. Групповые регистрационные имена (например, **guest** или **demo**) представляют собой удобную лазейку для хакеров, поэтому они запрещены во многих сетях федеральными законами, такими как HIPAA. Не допускайте этого в своей сети. Однако технические средства не могут предотвратить совместное использование пользователями паролей, поэтому в этом вопросе лучше всего вести разъяснительную работу.

## Пользовательские оболочки

Теоретически можно установить в качестве оболочки для пользовательской учетной записи любую программу, включая пользовательский сценарий. На практике использование оболочек, отличающихся от стандартных, таких как **bash** и **tcsh**, весьма опасно. Еще опаснее беспарольные регистрационные имена, оболочкой которых является сценарий. Если у вас возникнет соблазн создать такое регистрационное имя, примените вместо него пару ключей SSH без пароля.

## Привилегированные учетные записи

Единственная отличительная черта пользователя **root** состоит в том, что его идентификатор равен 0. Поскольку в файле **/etc/passwd** может быть несколько записей с таким идентификатором, существует и несколько способов входа в систему в качестве суперпользователя.

Один из способов, который хакеры, получив доступ к интерпретатору команд суперпользователя, широко применяют для открытия "черного хода", — редактирование файла **/etc/passwd**. Поскольку такие команды, как **who** и **w**, работают с регистрационным именем, хранящимся в файле **/var/run/utmp**, а не с идентификатором владельца регистрационного интерпретатора, они не в состоянии разоблачить хакера, который выглядит как рядовой пользователь, хотя на самом деле зарегистрирован в системе в качестве суперпользователя.

Не допускайте удаленную регистрацию суперпользователя даже через стандартную корневую учетную запись. Для того чтобы это запретить, следует с помощью оболочки OpenSSH установить в файле **/etc/ssh/sshd_config** параметр конфигурации `PermitRootLogin` равным `No`.

Благодаря программе **sudo** (см. раздел 4.3) необходимость регистрироваться в качестве суперпользователя, даже с системной консоли, возникает редко.

# 22.5. Модули PAM: украшение или чудо аутентификации

Аббревиатура PAM означает "подключаемые модули аутентификации" (Pluggable Authentication Modules — PAM). Эти модули освобождают программистов от необходимости выполнять рутинные операции для реализации систем аутентификации. Концепция и термин были придуманы в компании Sun Microsystems (которая в настоящее время является частью компании Oracle) и описаны в 1996 году в статье, авторами которой были Самар (Samar) и Лаи (Lai) из компании SunSoft.

В далеком прошлом команды вроде **login** включали встроенный код аутентификации, который предлагал пользователю ввести пароль, сравнил его с зашифрованной записью из файла **/etc/shadow** (на самом деле в настоящее время этот файл называется **/etc/passwd**) и делал вывод об их совпадении или несовпадении. Разумеется, другие команды (например, **passwd**) содержали такой же код. Не имея доступа к открытому коду, было невозможно изменить метод аутентификации, поэтому администраторы имели мало возможностей для управления настройками паролей (например, задавать правила, описывающие правильные пароли) или не имели их вообще. Появление модулей PAM в корне изменило положение дел.

Модули PAM поместили системные процедуры аутентификации в совместно используемую библиотеку, которая может вызывать программу **login** и другие программы. Выделение функций аутентификации в отдельную подсистему облегчает интеграцию новых методов аутентификации и шифрования в компьютерную систему. Например, многофакторная аутентификация может поддерживаться без внесения изменений в программы **login** и **passwd**.

Для системного администратора выбор правильного уровня безопасности для аутентификации стал простой задачей конфигурирования. Программисты также получили выгоду: они больше не обязаны писать сложный код для аутентификации и, что еще более важно, их системы аутентификации правильно реализуются с первой попытки. Модули PAM могут аутентифицировать все виды деятельности: регистрацию пользователей, другие формы доступа к системе, использование защищенных веб-сайтов и даже конфигурирование приложений.

## Системная поддержка для моделей PAM

Все рассмотренные нами операционные системы содержат модули PAM. Конфигурационная информация хранится в каталоге **/etc/pam.d** (Linux) или в файле **/etc/pam.conf** (Solaris, HP-UX и AIX). Форматы файлов конфигурации, в основном, совпадают, но в системах семейства UNIX вся информация хранится в одном файле, а в системах семейства Linux каждая служба или команда, использующая модули PAM, имеет отдельный файл.

С этой точки зрения поддержка модулей PAM является практически универсальной, но если вы используете другой вариант системы UNIX и хотите проверить, использует ли ваша система модули PAM, можно выполнить команду **ldd /bin/login** и увидеть, ссылаются ли исполняемые модули на совместно используемую библиотеку моделей PAM **libpam**.

## Конфигурация модулей PAM

Файлы конфигурации модулей PAM содержат по одной строке, каждая из которых представляет собой имя модуля PAM, используемого в системе.

Общий формат этих файлов имеет следующий вид.

```
[служба] тип_модуля управляющий_флаг путь_к_модулю [аргументы]
```

Поля разделяются пробелами.

Системы Linux не используют поле *служба* или, точнее, они помещают каждую службу в отдельный файл конфигурации и имя этого файла играет роль параметра *служба*, принятого в системе UNIX. Параметр *служба* может называть контекст аутентификации, к которому относится строка конфигурации (например, `login` для обычных реги-

страционных имен пользователей), или содержать ключевое слово `other` для установки системных параметров по умолчанию.

Рассмотрим иллюстративный фрагмент конфигурации из системы Solaris; все поля *путь_к_модулю* являются относительными к каталогу **`/usr/lib/security`**.

```
# login service

login   auth requisite   pam_authtok_get.so.1
login   auth required    pam_dhkeys.so.1
login   auth required    pam_unix_cred.so.1
login   auth required    pam_unix_auth.so.1
login   auth required    pam_dial_auth.so.1
...
```

Отдельные модули PAM имеют более тонкие настройки, чем просто "аутентифицировать пользователя", поэтому в файле конфигурации модулей PAM для каждой службы и каждого типа модуля может содержаться несколько строк. Серия строк для заданной службы и типа модуля называется *стеком*.

Порядок, в котором модули перечисляются в файле конфигурации, имеет значение. Например, модуль, предлагающий пользователю ввести пароль, должен предшествовать модулю, проверяющему корректность пароля. Один модуль может передавать результаты своей работы другому модулю, изменяя параметры окружения или переменные PAM.

Поле *тип_модуля* может иметь значения `auth`, `account`, `session` или `password`. Модуль типа `auth` идентифицирует пользователя и может предоставлять ему право группового доступа. Модуль типа `account` выполняет действия, не связанные с аутентификацией, например предоставляет доступ в зависимости от времени суток, ограничивает количество одновременно работающих пользователей или количество портов, на которых может происходить регистрация. (Например, можно использовать модуль типа `account` для ограничения регистрации суперпользователя на консоли.) Действия, которые необходимо выполнить до или после того, как пользователь получит доступ к требуемой службе, например монтирование файловой системы, реализуются модулем типа `session`. Наконец, модуль типа `password` используется, когда пользователь должен ввести аутентификационную информацию (например, пароль или кодовое словосочетание).

Поле *управляющий_флаг* определяет, как должны взаимодействовать модули, образующие стек (табл. 22.1).

**Таблица 22.1. Управляющие флаги модулей PAM**

| Флаг | Остановка в случае ошибки | Остановка в случае успеха | Комментарии |
|---|---|---|---|
| `binding`[a] | Нет | Да | Похож на флаг `sufficient`, но только ошибка приводит к отказу всего стека |
| `include`[a] | – | – | Включает другой файл конфигурации в данную точку стека |
| `optional` | Нет | Нет | Имеет значение, только если модуль единственный |
| `required` | Нет | Нет | Ошибка приводит со временем к отказу всего стека |
| `requisite` | Да | Нет | Так же как и `required`, но отказ стека наступает мгновенно |
| `sufficient` | Нет | Да | Неудачное название; см. пояснения в тексте |

[a] Значение `include` используется только в системе Linux, значение binding — только в системе Solaris.

Если бы модуль PAM мог просто возвращать код ошибки как только первый модуль в стеке потерпит неудачу, то система этих флагов была бы проще. К сожалению, система разработана так, что большинство модулей имеет шанс на выполнение независимо от успеха или сбоя других модулей, находящихся с ними на одном и том же уровне, и этот факт имеет определенные тонкости, связанные с потоком управления. (Это было сделано для того, чтобы хакеры не могли понять, какие модули PAM в стеке вызывают отказ.)

Модули `required` должны следовать один за другим; отказ одного из них гарантирует, что весь стек со временем даст сбой. Однако отказ модуля, помеченного флагом `required`, не приводит к немедленному сбою стека. Если вы хотите, чтобы сбой наступал моментально, вместо флага `required` используйте флаг `requisite`.

Успех модуля типа `sufficient` немедленно прекращает работу стека. Однако окончательный результат работы стека не обязательно должен быть успешным, поскольку модуль типа `sufficient` не может компенсировать ошибку выполненного ранее модуля типа `required`. Если ранее выполненный модуль типа `required` выдал ошибку, успешно выполненный модуль типа `sufficient` прекращает работу стека *и* возвращает ошибку в качестве итогового результата. Флаг `binding` в системе Solaris действует точно так же, как флаг `sufficient`, но отказ модуля типа `binding` гарантирует общий сбой стека. В противоположность этому, отказ модуля типа `sufficient` интерпретируется как ошибка модуля типа `optional`: он не влияет на окончательный результат, если стек не состоит из одного модуля.

Дело ясное, что дело темное, не так ли? Для того чтобы еще больше усложнить ситуацию, в системе Linux предусмотрена параллельная система альтернативных управляющих флагов, которые теоретически можно использовать вместо перечисленных выше межсистемных стандартов. В общем, для того чтобы объяснить, что означают управляющие флаги, необходимо несколько страниц. Однако мы не будем углубляться в эту тему, потому что конфигурации модулей PAM довольно стереотипны. Вряд ли вы станете писать что-то совершенно оригинальное. Мы упомянули некоторые детали только для того, чтобы дать вам представление о том, что имена управляющих флагов не соответствуют их назначению. Если вы планируете изменить настройки безопасности вашей системы, сначала убедитесь, что вы правильно понимаете смысл параметров, и дважды перепроверьте это. (Конфигурации модулей PAM не изменяют ежедневно. Как долго вы можете помнить разницу между флагами `requisite` и `required`?)

Для простоты рассмотрим следующий фрагмент конфигурации из файла **`pam.conf`** в системе Solaris.

```
# login service

login   auth requisite    pam_authtok_get.so.1
login   auth required     pam_dhkeys.so.1
login   auth required     pam_unix_cred.so.1
login   auth required     pam_unix_auth.so.1
login   auth required     pam_dial_auth.so.1
...
```

Рассмотрим отдельные модули.

Библиотечный модуль `pam_authtok_get` предлагает пользователю ввести свое имя (если оно еще не было введено) и пароль, а также хранить эти значения в токене аутентификации с именем `PAM_AUTHTOK`. Модуль `pam_dhkeys` используется для аутентификации вызова RPC (remote procedure call — удаленный вызов процедуры) для систем NIS или NIS+ и ищет ключи Диффи-Хеллмана, т.е. имя.

Модуль pam_unix_cred устанавливает мандаты для аутентифицированного пользователя, а модуль pam_unix_auth выполняет фактическую аутентификацию, проверяя, является ли значение, хранящееся в переменной PAM_AUTHTOK, правильным паролем пользователя. В заключение, модуль pam_dial_auth аутентифицирует пользователя для доступа по коммутируемым телефонным линиям к содержанию файлов **/etc/dialups** и **/etc/d_passwd**.

Один и тот же модуль может встречаться в разных конфигурационных файлах с разными значениями *тип_модуля*. Это прекрасно; если реализации нескольких типов имеют значительный объем общего кода, они часто объединяются в одну библиотеку.

## Подробный пример конфигурации системы Linux

Система Linux хранит каждый набор строк конфигурации модулей PAM, ссылающихся на одну и ту же *службу*, в отдельном файле, имя которого совпадает с именем службы. В остальном формат прежний, за исключением того, что поле *служба* больше не нужно.

Например, ниже приведен файл **/etc/pam.d/login** из системы SUSE с включенными файлами, которые раскрыты, чтобы сделать пример более связным.

```
auth       requisite    pam_nologin.so
auth       [user_unknown=ignore success=ok ignore=ignore auth_err=die
    default=bad] pam_securetty.so
auth       required     pam_env.so
auth       required     pam_unix2.so
account    required     pam_unix2.so
password   requisite    pam_pwcheck.so nullok cracklib
password   required     pam_unix2.so use_authtok nullok
session    required     pam_loginuid.so
session    required     pam_limits.so
session    required     pam_unix2.so
session    optional     pam_umask.so
session    required     pam_lastlog.so nowtmp
session    optional     pam_mail.so standard
session    optional     pam_ck_connector.so
```

Стек auth содержит несколько модулей. В первой строке модуль pam_nologin проверяет существование файла **/etc/nologin**. Если он существует, модуль немедленно прекращает регистрацию, если пользователь не является корневым. Модуль pam_securetty гарантирует, что суперпользователь может регистрироваться только с терминалов, перечисленных в файле **/etc/securetty**. В этой строке используется альтернативный синтаксис системы Linux, описанный на справочной странице **pam.conf**. В этом случае функционирование модуля аналогично реакции на флаг required. Модуль pam_env устанавливает переменные окружения в файле **/etc/security/pam_env.conf**, и наконец, модуль pam_unix2 проверяет мандаты пользователя, выполняя стандартную аутентификацию системы UNIX. Если один из этих модулей дает сбой, стек auth возвращает ошибку.

Стек account содержит только модуль pam_unix2, который в данном контексте оценивает корректность самой учетной записи. Например, он возвращает ошибку, если учетная запись просрочена или пароль должен быть изменен. В последнем случае модуль принимает от пользователя новый пароль и передает его модулям password.

Строка pam_pwcheck проверяет устойчивость предложенных паролей, вызывая библиотеку **cracklib**. Она возвращает ошибку, если новый пароль не соответствует установленным требованиям. Однако она допускает пустые пароли, потому что установлен флаг nullok. Строка pam_unix2 обновляет действующий пароль.

В заключение модули session выполняют несколько рутинных процедур. Модуль pam_loginuid устанавливает атрибут процесса ядра loginuid равным идентификатору пользователя, модуль pam_limits считывает лимиты использования ресурсов из файла **/etc/security/limits.conf** и устанавливает соответствующие параметры процесса, чтобы ввести их в действие. Модуль pam_unix2 регистрирует доступ пользователя в систему, а модуль pam_unmask устанавливает режим создания начального файла. Модуль pam_lastlog выводит время последней регистрации пользователя, а модуль pam_mail — сообщение, если пользователь получил новое электронное сообщение. В заключение, модуль pam_ck_connector уведомляет демон **ConsoleKit** (системный демон, управляющий сеансами регистрации) о новой регистрации.

В результате пользователь был успешно аутентифицирован, и модули PAM вернули управление программе **login**.

## 22.6. Программы с установленным битом SETUID

Программы, выполняемые с установленным битом SETUID (Set User ID — смена идентификатора пользователя), запускаются пользователем, который считается их владельцем. Например, программа **passwd** должна запускаться с привилегиями суперпользователя, чтобы иметь возможность модифицировать файл **/etc/shadow**, когда пользователи изменят свои пароли. Основная информация об этой функциональной возможности изложена в разделе 4.1.

Программы, выполняемые с установленным битом SETUID, особенно имеющие привилегии суперпользователя, являются источником проблем для безопасности системы. Тем не менее программы этой категории, поставляемые вместе с операционными системами, являются безопасными, по крайней мере теоретически. Однако огрехи в защите обнаруживались в прошлом и, несомненно, будут обнаруживаться в будущем.

Самый надежный способ уменьшения количества проблем, вызванных сменой идентификатора, — сведение к минимуму числа программ с установленным битом SETUID. Подумайте дважды, прежде чем инсталлировать такую программу, и вообще избегайте устанавливать этот бит для своих программ.

Выполнение программ с установленными битами SETUID и SETGID (Set Group ID — смена идентификатора группы) можно отключать в отдельных файловых системах с помощью опции **nosuid** команды **mount**. Это целесообразно делать в файловых системах, содержащих начальные каталоги пользователей или смонтированные из ненадежных административных доменов.

Полезно периодически сканировать диски на предмет выявления новых программ с установленным битом SETUID. Хакер, взломавший систему, вполне может создать собственный командный SETUID-интерпретатор, который облегчит ему последующий вход в систему. Ряд программ, описанных в разделе 22.7, позволяет обнаруживать такие файлы, но вполне подойдет и простая команда **find**.

```
/usr/bin/find / -user root -perm -4000 -print |
   /bin/mail -s "Setuid root files" netadmin
```

В данном случае пользователю **netadmin** по электронной почте посылается список всех файлов, принадлежащих пользователю **root** и имеющих установленный бит

SETUID. (На практике может потребоваться более конкретная информация об искомых файловых системах.)

## 22.7. Эффективное использование команды chroot

Системный вызов **chroot** ограничивает процесс конкретной библиотекой. Он закрывает доступ к файлам, расположенным за пределами каталога, и тем самым смягчает последствия, которые могут возникнуть при его взломе.

Команда **chroot** — это простая оболочка описанного выше системного вызова. Кроме того, некоторые демоны, представляющие угрозу для безопасности системы, имеют встроенную поддержку команды **chroot** и должны запускаться только в этом режиме, который устанавливается в их конфигурационных файлах.

Эксперты по вопросам безопасности иногда косо смотрят на использование команды **chroot** в этом контексте, поскольку они считают, что неправильное использование или неверное понимание этой команды может создать у системных администраторов иллюзию безопасности. Они жалуются, что некоторые системные администраторы используют команду **chroot**, чтобы избавить себя от хлопот, связанных с использованием других инструментов для обеспечения безопасности, например регулярного обновления программного обеспечения и пристального мониторинга.

Эти утверждения не точны, и в любом случае их нельзя считать приговором команде **chroot**. Аналогичные высказывания звучали и в адрес брандмауэров, но лишь немногие эксперты осмелятся посоветовать удалить фильтры пакетов из вашей сети. При условии правильного использования в сочетании с другими средствами безопасности команда **chroot** достойна включения в арсенал инструментов для защиты сети (даже если это не входило в планы ее разработчиков).

Перечислим сценарии, в которых целесообразно использовать команду **chroot**.

- Допустим, что вы хотите запустить демона без привилегий суперпользователя, например Apache или BIND в ограниченном поддереве файловой системы. Если демон окажется взломанным, то хакер будет ограничен поддеревом, поскольку других уязвимых мест нет и эскалация привилегий не произойдет.
- Предположим, что вы хотите ограничить удаленных пользователей конкретной совокупностью файлов и команд.

Однако команда **chroot** может защитить вас в указанных сценариях только при выполнении следующих условий.

- Все процессы в окружении **chroot** запускаются без привилегий суперпользователя. Процессы, которые запускаются с привилегиями суперпользователя, всегда способны взломать окружение **chroot**.
- Вы не запускаете программы, выполняемые с установленным битом **setuid**, с правами суперпользователя.
- Окружение **chroot** соответствует современному уровню и является минимальным, т.е. содержит только исполняемые программы, библиотеки и конфигурационные файлы, необходимые для выполнения поставленной задачи.

В эпоху совместно используемых библиотек и зависимостей между процессами создание описанного выше окружения **chroot** становится непростой задачей. Некоторые сценарии, которые могут облегчить ее решение, содержатся в пакете JailKit (`olivier.sessink.nl/jailkit`).

# 22.8. Инструментальные средства защиты

Для решения задач, упоминавшихся в предыдущих разделах, можно использовать свободно распространяемые программы. Ниже будут рассмотрены наиболее интересные из них.

## Команда nmap: сканирование сетевых портов

Эта команда является сканером сетевых портов. Ее основное назначение — проверка указанных компьютеров на предмет того, какие TCP- и UDP-порты на них прослушиваются серверными программами[3]. За большинством сетевых служб закреплены "известные" порты, поэтому такая информация позволяет узнать, какие программы выполняются на компьютере.

Запуск команды **nmap** — отличный способ узнать, как выглядит система в глазах того, кто пытается ее взломать. В качестве примера приведем отчет, выданный этой командой на самом обычном, относительно незащищенном компьютере.

```
ubuntu$ nmap -sT ubuntu.booklab.atrust.com

Starting Nmap 4.20 ( http://insecure.org ) at 2009-11-01 12:31 MST
Interesting ports on ubuntu.booklab.atrust.com (192.168.20.25):
Not shown: 1691 closed ports
PORT      STATE   SERVICE
25/tcp    open    smtp
80/tcp    open    http
111/tcp   open    rpcbind
139/tcp   open    netbios-ssn
445/tcp   open    microsoft-ds
3306/tcp  open    mysql

Nmap finished: 1 IP address (1 host up) scanned in 0.186 seconds
```

Аргумент **-sT** сообщает команде **nmap** о необходимости подключиться к каждому TCP-порту на указанном узле[4]. Как только соединение устанавливается, команда **nmap** немедленно отключается, что не очень корректно, зато безболезненно для правильно написанного сетевого сервера.

Как следует из примера, на узле `ubuntu` выполняются две службы, являющихся традиционными источниками проблем для безопасности системы: **portmap** (`rpcbind`) и почтовый сервер (`smtp`). Это наиболее вероятные места для атаки хакеров.

Колонка `STATE` (состояние) содержит запись `open` (открыт) для портов, с которыми связаны серверы, **closed** (закрыт) для портов, с которыми не связан ни один сервер, запись `unfiltered` (нефильтруемый) для портов, пребывающих в неизвестном состоянии, и запись `filtered` (фильтруемый) для портов, доступ к которым невозможен из-за наличия брандмауэра. Чаще всего встречаются нефильтруемые порты. Они отображаются только при ACK-сканировании. Вот, например, результат запроса к более защищенному коммерческому веб-серверу `secure.booklab.atrust.com`.

---

[3] Как объяснялось в главе 14, порт — это нумерованный канал взаимодействия. IP-адрес идентифицирует весь компьютер, а комбинация IP-адреса и номера порта определяет конкретный сервер или сетевое соединение.

[4] На самом деле по умолчанию проверяются только привилегированные (их номера меньше 1024) и "известные" порты. С помощью опции **-p** можно явно задать диапазон сканирования.

```
ubuntu$ nmap -sT secure.booklab.atrust.com

Starting Nmap 4.20 ( http://insecure.org ) at 2009-11-01 12:42 MST
Interesting ports on secure.booklab.atrust.com (192.168.20.35):
Not shown: 1691 closed ports
PORT     STATE   SERVICE
25/tcp   open    smtp
80/tcp   open    http

Nmap finished: 1 IP address (1 host up) scanned in 0.143 seconds
```

В данном случае очевидно, что компьютер сконфигурирован на обработку электронной почты по протоколу SMTP и работу с сервером HTTP. Брандмауэр блокирует доступ к остальным портам.

Помимо стандартного опроса TCP- и UDP-портов, команда **nmap** способна проверять порты "по-хакерски", не устанавливая с ними реального соединения. В большинстве случаев посылаются пакеты, которые похожи на пакеты из уже имеющегося TCP-соединения (это не квитирующие пакеты), а затем проверяются полученные в ответ диагностические пакеты. Таким способом можно обойти брандмауэр или программу мониторинга сети, которая контролирует появление сканеров портов. Если в вашей организации имеется брандмауэр (см. раздел 22.11), не поленитесь проверить его в разных режимах сканирования.

Команда **nmap** обладает магической способностью угадывать, какая операционная система установлена на удаленном узле. Это делается путем анализа деталей реализации стека TCP/IP. Чтобы проверить работу команды **nmap** в этом режиме, воспользуйтесь опцией **-O**, например.

```
ubuntu$ sudo nmap -sV -O secure.booklab.atrust.com
Starting Nmap 4.20 ( http://insecure.org ) at 2009-11-
1 12:44 MST
Interesting ports on secure.booklab.atrust.com (192.168.20.35):
Not shown: 1691 closed ports
PORT     STATE   SERVICE    VERSION
25/tcp   open    smtp       Postfix smtpd
80/tcp   open    http       lighttpd 1.4.13
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.6.16 - 2.6.24
Nmap finished: 1 IP address (1 host up) scanned in 8.095 seconds
```

Эта возможность может оказаться весьма полезной при анализе состояния локальной сети. К сожалению, она же является рабочим инструментом хакеров, которые могут определить тип атаки на основании известных слабых мест конкретных операционных систем или серверов.

Не забывайте также, что большинство администраторов не приветствуют попыток сканирования сети и выяснения ее слабых мест, какими бы благородными ни были мотивы. *Никогда* не запускайте команду **nmap** в чужой сети без разрешения сетевого администратора.

## Nessus: сетевой сканер следующего поколения

В 1998 году Рено Дерезон (Renaud Deraison) выпустил интересный пакет Nessus, в котором реализуются многие функциональные возможности сканера. Он использует более 31000 надстроек для проверки локальных и удаленных дефектов. Несмотря на то что

эта программа имеет закрытый код и является коммерческим продуктом, она свободно распространяется и для нее регулярно появляются новые надстройки. Это наиболее широко распространенный и полный из доступных сканеров.

Программа Nessus — это сетевой сканер, который ничему не доверяет. Вместо того чтобы предполагать, к примеру, будто все веб-серверы работают через порт 80, он ищет веб-серверы на любом порту, а затем анализирует их слабые места. Программа не доверяет даже номеру версии, о котором сообщает сам сервер, проверяя все известные слабые места, чтобы понять, насколько уязвим сервер.

Для первого запуска программы Nessus требуется потратить немало усилий (необходимо установить много пакетов, которые не входят в стандартный дистрибутив), но конечный результат того стоит. Система Nessus состоит из клиента и сервера. Сервер играет роль базы данных, а клиент отвечает за графический пользовательский интерфейс. Серверы и клиенты могут работать на платформах UNIX или Windows.

Одним из главных преимуществ программы Nessus является ее модульная структура, позволяющая сторонним разработчикам добавлять собственные модули проверки. Благодаря активности сообщества ее пользователей, эта программа сможет быть полезной долгое время.

## John the Ripper: средство для выявления слабых паролей

Один из способов пресечь использование плохих паролей заключается в том, чтобы самостоятельно взломать пароль и заставить пользователя выбрать другой. John the Ripper — это сложная программа, разработанная компанией Solar Designer, которая реализует разные алгоритмы взлома паролей в виде единого инструмента. Она заменила программу **crack**, которая была описана в предыдущем издании книги.

Несмотря на то что большинство систем использует файл теневых паролей, чтобы скрыть зашифрованные пароли от публики, целесообразно проверять, что пароли пользователей являются устойчивыми ко взлому.[5] Знание пароля пользователя может оказаться полезным, потому что люди предпочитают все время использовать один и тот же пароль. Единственный пароль может обеспечить доступ к другой системе, зашифровать файлы, хранящиеся в рабочем каталоге, и открыть доступ к финансовым счетам в сети веб. (Не стоит и говорить, что использовать пароль таким способом очень глупо. Однако никто не хочет запоминать десять разных паролей.)

Несмотря на свою внутреннюю сложность, программа John the Ripper весьма проста в использовании. Достаточно просто нацелить программу **john** на файл, подлежащий взлому, чаще всего **/etc/shadow**, и произойдет чудо.

```
$ sudo ./john /etc/shadow
Loaded 25 password hashes with 25 different salts (FreeBSD MD5 [32/32])
password (jsmith)
badpass (tjones)
```

В этом примере из теневого файла были считаны 25 уникальных паролей. После взлома пароля программа John выводит его на экран и сохраняет в файл **john.pot**. Этот вывод содержит пароль в левом столбце и регистрационное имя в скобках в правом столбце. Для просмотра паролей после завершения работы программы **john**, следует выполнить ту же самую команду с аргументом **-show**.

[5] Особенно это относится к паролям системных администраторов, имеющих привилегии ***sudo***.

На момент написания книги самой новой версией программы John the Ripper была версия 1.7.3.4. Она доступна на сайте `openwall.com/john`. Поскольку вывод программы содержит взломанные файлы, его следует тщательно охранять и удалять сразу после проверки, убедившись, что пароли пользователей являются небезопасными.

Как и для большинства других методов мониторинга безопасности, перед взломом паролей с помощью программы John the Ripper необходимо получить одобрение руководства.

## Команда hosts_access: управление доступом к узлу

Брандмауэры сети являются первой линией защиты от доступа со стороны неавторизованных узлов, однако они не должны быть единственным средством защиты. Два файла, **`/etc/hosts.allow`** и **`/etc/hosts.deny`**, также являются оболочками протокола TCP, которые могут ограничивать доступ к службам для генерирования сетевых запросов. Файл **`hosts.allow`** содержит список узлов, которым разрешено соединяться с конкретной службой, а файл **`hosts.deny`** ограничивает доступ. Однако эти файлы управляют доступом только к службам, которые знают о существовании команды **`hosts_access`**, например к службам, управляемым демонами **`inetd`**, **`xinetd`**, **`sshd`** и некоторыми конфигурациями программы **`sendmail`**.

В большинстве ситуаций разумно установить ограничения и разрешить доступ только к службам, играющим важную роль для указанных узлов. Мы рекомендуем отключить доступ, установленный по умолчанию в файле **`hosts.deny`**, с помощью одной строки.

```
ALL:ALL
```

Затем вы можете разрешить доступ в конкретных ситуациях с помощью файла **`hosts.allow`**. Следующая конфигурация разрешает доступ к системе SSH с узлов сети 192.168/16, а также к агенту **`sendmail`** отовсюду.

```
sshd: 192.168.0.0/255.255.0.0
sendmail: ALL
```

Формат записи в файле имеет вид *служба: узел* или *служба: сеть*. Соединения, которым отказано в доступе, перечисляются в журнале syslog. Соединения, которым не был разрешен доступ к службе, немедленно закрываются.

Большинство дистрибутивных пакетов системы Linux по умолчанию содержит файлы **`hosts.allow`** и **`hosts.deny`**, но, как правило, они пустые. Другие рассмотренные нами операционные системы после инсталляции предлагают использовать оболочки TCP.

## Bro: программная система для распознавания вторжения в сеть

Открытая система Bro предназначена для распознавания вторжений в сеть (network intrusion detection system — NIDS). Она выполняет мониторинг сети и следит за подозрительной активностью. Эта система была написана Верном Паксоном (Vern Paxson) и доступна на сайте `bro-ids.org`.

Система Bro проверяет весь трафик, поступающий в сеть и исходящий из нее. Она может функционировать в пассивном режиме, генерируя предупреждения о подозрительной активности, или в активном режиме, в котором она пресекает враждебные действия. Оба режима, скорее всего, потребуют внесения изменений в конфигурацию вашей сети.

В отличие от других систем NIDS, система Bro отслеживает потоки трафика, а не просто сопоставляет образцы внутри отдельных пакетов. Это значит, что система Bro

может распознавать подозрительную активность, основываясь на информации о том, кто с кем говорит, даже не сравнивая никаких строк или шаблонов. Например, система Bro может решать следующие задачи.

- Распознавать системы, использующие "мостики" ("stepping stones"), определяя корреляцию между входящим и исходящим трафиками.
- Распознавать сервер, имеющий лазейку, инсталлированную для неожиданных исходящих соединений сразу после входящего соединения.
- Распознавать протоколы, выполняемые на нестандартных портах.
- Сообщать о правильно угаданных паролях (и игнорировать неправильные догадки).

Некоторые из этих функциональных возможностей требуют существенных системных ресурсов, но система Bro имеет поддержку кластеризации, облегчающую управление группами машин, распознающих вторжение.

Система Bro имеет сложный язык конфигурации, требующий значительного опыта кодирования. К сожалению, в системе нет конфигурации, заданной по умолчанию, которую было бы легко инсталлировать новичкам. Большинство сайтов требует умеренного уровня настройки.

Система до некоторой степени поддерживается Группой сетевых исследований Международного института компьютерных наук (Networking Research Group of the International Computer Science Institute (ICSI)), но в большей степени она поддерживается сообществом ее пользователей. Если вы ищете готовую коммерческую систему NIDS, то, вероятно, будете разочарованы системой Bro. Однако она может делать то, что не доступно коммерческим системам NIDS, и может служить дополнением или заменой коммерческого продукта.

## Snort: популярная программная система для распознавания проникновения в сеть

Открытая система Snort предназначена для предотвращения и распознавания несанкционированного проникновения в сеть; написана Марти Решем (Marty Roesch) и в настоящее время поддерживается коммерческой компанией Sourcefire (`snort.org`). Она де-факто стала стандартом для кустарных систем NIDS, а также основной для многих коммерческих реализаций систем NIDS с "управляемыми услугами".

Сама по себе система Snort распространяется как пакет с открытым кодом. Однако компания Sourcefire взимает плату за подписку на доступ к новейшим правилам распознавания.

Большое количество платформ, созданных сторонними производителями, включают в себя или экспортируют систему Snort, причем некоторые из этих проектов являются программами с открытым кодом. Ярким примером является проект Aanval (`aanval.com`), собирающий данные от многочисленных датчиков системы Snort на веб-консоль.

Система Snort перехватывает пакеты из кабеля и сравнивает их с наборами правил, которые называются сигнатурами. Когда система Snort распознает событие, которое представляет интерес, она может предупредить системного администратора или установить контакт с сетевым устройством и, помимо прочего, заблокировать нежелательный трафик.

Несмотря на то что система Bro намного мощнее, система Snort намного проще и легче конфигурируется, благодаря чему является удачным выбором для стартовой платформы NIDS.

## OSSEC: система для распознавания вторжения в сеть на уровне узла

Мучались ли вы бессонницей из-за опасения, что ваша система безопасности сети будет взломана? Не предполагали ли вы, что ваш рассерженный сотрудник может инсталлировать вредоносную программу в ваших системах? Если на один из этих вопросов вы ответили положительно, то подумайте над инсталляцией системы для распознавания проникновения на уровне узла (HIDS), такой как OSSEC.

Система OSSEC — это свободное программное обеспечение, доступное в виде открытого кода по лицензии GNU (General Public License). Ее коммерческая поддержка обеспечивается компанией Third Brigade (недавно приобретенной компанией Trend Micro). Система OSSEC имеет версии для систем Linux, Solaris, HP-UX, AIX и Windows. Она обеспечивает следующие функциональные возможности.

- Распознавание руткитов.
- Проверка целостности файловой системы.
- Анализ регистрационных файлов.
- Выдача сигналов по расписанию.
- Активные реакции.

Система OSSEC работает в заданной операционной системе и отслеживает ее активность. Она может выдавать сигналы или предпринимать действия, предусмотренные набором правил, которые устанавливает пользователь. Например, система OSSEC может выполнять мониторинг операционных систем с целью выявления добавления неавторизованных файлов и посылать по электронной почте уведомления, похожие на приведенное ниже.

```
Subject: OSSEC Notification - courtesy - Alert level 7
Date: Fri, 15 Jan 2010 14:53:04 -0700
From: OSSEC HIDS <ossecm@courtesy.atrust.com>
To: <courtesy-admin@atrust.com>

OSSEC HIDS Notification.
2010 Jan 15 14:52:52
Received From: courtesy->syscheck
Rule: 554 fired (level 7) -> "File added to the system."
Portion of the log(s):

New file
'/courtesy/httpd/barkingseal.com/html/wp-content/uploads/2010/01/hbird.jpg'
added to the file system.

--END OF NOTIFICATION
```

Таким образом, система OSSEC работает круглосуточно, следя за операционной системой. Мы рекомендуем запускать эту систему на каждой операционной системе, одновременно изменяя политику управления (см. главу 32).

### *Основные принципы системы OSSEC*

Система OSSEC состоит из двух основных компонентов: менеджера (сервера) и агентов (клиентов). В сети нужен один менеджер, который необходимо инсталлировать первым. Менеджер хранит базу данных для проверки целостности файловой системы, реги-

страционные журналы, события, декодеры, основные варианты конфигурации, а также записи об аудите системы для всей сети. Менеджер может соединяться с любым агентом OSSEC, независимо от его операционной системы. Менеджер также может отслеживать работу определенных устройств, которые не имеют специального агента OSSEC.

Агенты работают в системах, которые подвергаются мониторингу, и сообщают информацию менеджеру. Они имеют маленькую зону обслуживания и оперируют небольшим объемом привилегий. Большую часть конфигурации агент получает от менеджера. Соединение между сервером и агентом зашифровано и аутентифицировано. Для каждого агента менеджер должен создать отдельный ключ аутентификации.

Система OSSEC классифицирует серьезность сигналов по уровням от 0 до 15; число 15 соответствует высшему уровню опасности.

### *Инсталляция системы OSSEC*

Система OSSEC еще не стала частью основных дистрибутивных пакетов операционных систем UNIX и Linux, даже в качестве вспомогательного пакета. По этой причине вы должны загрузить пакет исходного кода в веб-браузер или инструмент, такой как **wget**, а затем собрать программу.

```
$ wget http://ossec.net/files/ossec-hids-latest.tar.gz
$ tar -zxvf ossec-hids-latest.tar.gz
$ cd ossec-hids-*
$ sudo ./install.sh
```

Сценарий инсталляции спрашивает, какой язык вы предпочитаете (например, "en" для английского) и какой тип инсталляции хотите выполнить — серверный, агентский или локальный. Если вы инсталлируете систему OSSEC в отдельной, персонально управляемой операционной системе, следует выбрать локальный тип инсталляции. В противном случае сначала следует выполнить инсталляцию сервера OSSEC в предназначенной для этого операционной системе, а затем инсталляцию агента во всех остальных системах, которые будут предметом мониторинга. Сценарий инсталляции тоже задает несколько дополнительных вопросов, например, по какому адресу посылать уведомления и какие модули мониторинга следует подключить.

После завершения инсталляции запустите систему OSSEC с помощью следующей команды.

```
server$ sudo /var/ossec/bin/ossec-control start
```

Затем зарегистрируйте каждого агента у менеджера. Находясь на сервере, выполните следующую команду.

```
server$ sudo /var/ossec/bin/manage_agents
```

Вы увидите меню, которое будет выглядеть примерно так.

```
****************************************
* OSSEC HIDS v2.3 Agent manager.
* The following options are available:
****************************************
(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.
Choose your action: A,E,L,R or Q:
```

Выберите пункт **A**, чтобы добавить агента, а затем наберите имя и IP-адрес этого агента. Затем выберите пункт **E** и извлеките ключ агента. Вот как это выглядит.

```
Available agents:
  ID: 001, Name: linuxclient1, IP: 192.168.74.3
Provide the ID of the agent to extract the key (or '\q' to quit): 001
Agent key information for '001' is:
MDAyIGxpbnV4Y2xpZW50MSAxOTIuMTY4Ljc0LjMgZjk4YjMyYzlkMjg5MWJlMT
...
```

В заключение зарегистрируйте в системе агента и выполните команду **manage_agents**.

```
agent$ sudo /var/ossec/bin/manage_agents
```

Находясь на компьютере клиента, вы увидите меню с другими пунктами.

```
****************************************
* OSSEC HIDS v2.3 Agent manager.
* The following options are available:
****************************************
(I)mport key from the server (I).
(Q)uit.
Choose your action: I or Q:
```

Выберите пункт **I**, а затем вырежьте и вставьте ключ, который был извлечен ранее. После того как добавите агента, вы должны заново запустить сервер OSSEC. Повторите процесс генерирования ключа, его извлечение и инсталляцию каждого агента, с которым хотите установить связь.

### *Конфигурация системы OSSEC*

После инсталляции и запуска системы OSSEC вы захотите настроить ее так, чтобы она выдавала достаточный объем информации, но не слишком большой. Большая часть конфигурации хранится на сервере в файле **/var/ossec/etc/ossec.conf**. Это XML-подобный файл, который подробно прокомментирован и вполне понятен, хотя и содержит десятки параметров.

Чаще всего необходимо конфигурировать список файлов, которые следует игнорировать при проверке целостности (наличия изменений) файла. Например, если у вас есть приложение, записывающее свой регистрационный файл в каталог **/var/log/customapp.log**, то вы можете добавить следующую строку в раздел <syscheck> этого файла.

```
<syscheck>
<ignore>/var/log/customapp.log</ignore>
</syscheck>
```

После того как внесете эти изменения и заново запустите сервер OSSEC, система OSSEC прекратит выдавать сообщения при каждом изменении регистрационного файла. Многие параметры конфигурации системы OSSEC описаны в файле ossec.net/main/manual/configuration-options.

Для того чтобы запустить и настроить систему HIDS, требуется затратить много времени и усилий. Но через несколько недель вы отфильтруете шум и система станет выдавать ценную информацию об изменяющихся условиях в вашем окружении.

# 22.9. Мандатное управление доступом

Мандатное управление доступом (Mandatory Access Control — MAC) представляет собой альтернативу традиционному механизму управления доступом в системе UNIX, который передает управление всеми разрешениями в руки администратора по безопасности. В противоположность стандартной модели, описанной в главах 4 и 6, система MAC не позволяет пользователям модифицировать никакие разрешения, даже для своих собственных объектов.

Стратегия безопасности в системе MAC управляет доступом в соответствии со степенью конфиденциальности управляемого ресурса. Пользователям присваиваются определенные уровни доступа, образующие структурную иерархию. Пользователи могут читать и записывать информацию, относящуюся к их уровню доступа или более низкому, но не имеют доступа к объектам, имеющим более высокую степень конфиденциальности. Например, пользователь с уровнем доступа "секретный" не может читать документы, классифицированные как совершенно секретные.

Хорошо реализованная стратегия системы MAC основывается на принципе минимальных привилегий (т.е. разрешает доступ только при необходимости), точно так как правильно спроектированные брандмауэры пропускают только знакомые службы и клиентов. Система MAC может предотвратить взлом системы через уязвимое программное обеспечение (например, из-за переполнения буфера), ограничивая область их влияния лишь несколькими конкретными ресурсами, необходимыми для данной программы.

Очевидно, что для реализации механизма MAC в операционных системах UNIX и Linux необходимо внести изменения в ядро. В рассмотренных нами системах семейства UNIX (Solaris, HP-UX и AIX) версии с внедренным механизмом MAC предоставляются за дополнительную плату. Эти версии называются Solaris Trusted Extensions (бывшая Trusted Solaris), HP-UX Security Containment и Trusted AIX соответственно.

Если вы не работаете с секретными правительственными документами, то вряд ли вам понадобятся эти версии.

## Система Linux с усиленной системой безопасности (SELinux)

Система SELinux (Security-enhanced Linux) — это вариант системы Linux, в котором реализован механизм MAC. Несмотря на то что в некоторых дистрибутивных пакетах эта система стала основной, она стала печально известной из-за трудностей в управлении и ошибок. Приведем безымянную цитату из страницы Википедии, посвященной ранней версии системы SELinux, в которой отражено отчаяние многих системных администраторов.

*"Как ни странно, несмотря на то, что система SELinux разрабатывалась для того, чтобы облегчать создание стратегий индивидуального управления доступом, специально приспособленных к способам и правилам хранения данных в организациях, вспомогательные программные средства настолько скудны и неудобны, что поставщики преимущественно ограничиваются консультациями, которые обычно принимают вид нарастающих модификаций стандартных стратегий безопасности."*

Несмотря на сложность в управлении системой SELinux, ее адаптация постепенно увеличивается, особенно в правительственных учреждениях, в которых приняты строгие правила безопасности. В рассмотренных нами примерах дистрибутивных пакетов

системы Linux самую совершенную модель SELinux имел пакет Red Hat Enterprise Linux. Механизм SELinux доступен как вспомогательный пакет для систем Ubuntu и SUSE.

Разработка стратегии — сложная тема. Для защиты нового демона, например, необходимо тщательно пронумеровать все файлы, каталоги и другие объекты, к которым другие объекты имеют доступ. Для сложного программного обеспечения, такого как **sendmail** или Apache **httpd**, эта задача может оказаться довольно сложной. По крайней мере одна компания предлагает трехдневный тренинг по разработке стратегий.

К счастью, многие общие стратегии доступны в Интернете, и большинство дистрибутивных пакетов имеет разумные решения, принятые по умолчанию. Их легко инсталлировать и конфигурировать для конкретной среды. Полноценный редактор стратегий, предназначенный для их легкой реализации, можно найти на сайте `seedit.sourceforge.net`.

Механизм SELinux встроен в систему Red Hat Enterprise Linux начиная с версии 4. По умолчанию инсталляция системы RHEL включает готовую защиту SELinux.

Конфигурацией системы SELinux управляет файл **/etc/selinux/config**. Интерес представляют следующие строки.

```
SELINUX=enforcing
SELINUXTYPE=targeted
```

Первая строка имеет три возможных значения: `enforcing`, `permissive` или `disabled`. Значение `enforcing` гарантирует применение загружаемых правил и запрещает их нарушение. Значение `permissive` разрешает нарушать правила, но регистрирует нарушения в журнале с помощью системы syslog, что оказывается полезным для отладки. Значение `disabled` вообще отключает механизм SELinux. Параметр `SELINUXTYPE` означает тип применяемой стратегии. Система Red Hat реализует две стратегии: `targeted`, которая определяет дополнительный уровень безопасности для демонов, которых защищает система Red Hat[6], и `strict`, которая защищает всю систему. Несмотря на доступность стратегии `strict`, в системе Red Hat она не поддерживается; ее ограничения настолько сильны, что системе их трудно применять. Стратегия `targeted` предоставляет защиту для важных сетевых демонов без влияния на систему в целом, по крайней мере, теоретически. Однако даже стратегия `targeted` не является идеальной. Если у вас есть проблемы со вновь установленным программным обеспечением, поищите в файле **/var/log/messages** сообщения об ошибках системы SELinux.

Система SUSE использует реализацию механизма MAC, выполненную компанией Novell и получившую имя AppArmor. Однако в версии 11.1 система SUSE также включает основные функциональные возможности механизма SELinux.

Система Ubuntu по умолчанию поставляется с механизмом AppArmor. Пакеты SELinux для систем Ubuntu поддерживаются Расселом Кокером (Russell Coker), бывшим сотрудником компании Red Hat, разработавшим стратегии `targeted` и `strict`.

## 22.10. Системы криптографической защиты

Большинство широко используемых в системах UNIX сетевых протоколов было создано до широкого распространения Интернета и до изобретения современных криптографических систем. При проектировании многих протоколов о безопасности просто не

[6] Защищаемыми демонами являются **httpd**, **dhcpd**, **mailman**, **named**, **portmap**, **nscd**, **ntpd**, **mysqld**, **postgres**, **squid**, **winbindd** и **ypbind**.

думали. Там же, где она якобы учитывалась, реализованные механизмы дискредитируются передачей паролей в незашифрованном виде или отсутствием проверки на надежность компьютера или порта, от которого поступил пакет.

Сегодня эти протоколы эксплуатируются в кишащих хакерами крупных корпоративных локальных сетях и Интернете, где весь трафик открыт для прослушивания. Более того, любой может вмешаться в сетевой диалог. Как удостовериться в личности собеседника?

Криптография — это решение многих проблем. Давным-давно существовала возможность шифровать сообщения так, чтобы даже в случае перехвата их нельзя было прочитать, но это лишь одно из чудес криптографии. Такие разработки, как шифрование с открытым ключом и защитное хеширование, сделали возможным построение криптосистем, удовлетворяющих самым строгим требованиям.

Тем, кого интересуют вопросы криптографии, рекомендуем обратиться к документу "RSA Labs' Frequently Asked Questions about Today's Cryptography" (`rsa.com/rsalabs`). Несмотря на название, это объемистый трактат, который можно загрузить в формате PDF. Этот документ не обновлялся с 2000 года, но большая часть его содержания все еще не потеряла актуальности. Кроме того, историю криптографии можно подробно изучить по книге Стивена Леви (Stephen Levy) *Crypto*.

## Kerberos: унифицированный подход к сетевой безопасности

Система Kerberos, разработанная в Массачусетсском технологическом институте (MIT), ориентирована на решение задач, связанных с обеспечением безопасности компьютерных сетей. Kerberos — это система аутентификации, предоставляющая "гарантию" того, что пользователи и служебные программы действительно являются теми, за кого себя выдают. Никаких дополнительных проверок и шифрования передаваемых данных не предусмотрено.

Используя алгоритм DES, Kerberos создает вложенные наборы идентификаторов, называемых *билетами* или *мандатами*. Мандаты передаются по сети с целью подтверждения личности пользователя и предоставления ему доступа к сетевым службам. В каждой организации, где используется Kerberos, должен выделяться хотя бы один физически защищенный компьютер (называемый сервером аутентификации) для выполнения демона Kerberos. Этот демон выдает мандаты пользователям и служебным программам, требующим аутентификации, на основании предъявляемых ими "удостоверений", в частности паролей.

По сути, Kerberos улучшает традиционную схему парольной защиты всего двумя способами: пароли никогда не передаются по сети в незашифрованном виде и пользователи избавляются от необходимости многократно вводить пароли.

Сообщество пользователей Kerberos может похвастаться одним из самых толковых и оригинальных документов, посвященных криптосистемам, — "Designing an Authentication System: a Dialogue in Four Scene" (Проектирование системы аутентификации: диалог в четырех актах) Билла Брайанта (Bill Bryant). Его будет интересно почитать любому, кто интересуется темой криптографии. Документ можно найти по адресу:

```
web.mit.edu/kerberos/www/dialogue.html
```

Лучше использовать Kerberos, чем вообще отказаться от средств защиты. К сожалению, Kerberos нельзя назвать полностью безопасной системой, а ее инсталляция и запуск — не самое приятное занятие. В то же время она не заменяет собой ни одну из программ, описанных в данной главе.

К сожалению, (хотя, возможно, вполне предсказуемо) система Kerberos, распространяемая как часть службы Active Directory системы Windows, использует запатентованные, недокументированные расширения. Как следствие, она плохо взаимодействует с традиционной версией системы. К счастью, модуль **winbind** позволяет системам UNIX и Linux взаимодействовать с версией системы Kerberos для службы Active Directory. Более подробную информацию о настройке системы Kerberos можно найти в разделе 30.9.

## PGP: высокая конфиденциальность

Пакет PGP (Pretty Good Privacy), написанный Филом Циммерманом (Phil Zimmermann), содержит криптографические утилиты, ориентированные, в основном, на безопасность систем электронной почты. Он используется для шифрования данных, генерирования сигнатур и проверки источников происхождения файлов и сообщений.

🕮 Более подробная информация о защите электронной почты приведена в разделе 20.6.

Пакет PGP имеет интересную историю, которая включает в себя судебные иски, криминальные расследования и приватизацию фрагментов исходного пакета PGP. В настоящее время формат файла и протоколы PGP стандартизованы организацией IETF под именем OpenPGP, и уже существует несколько реализаций предложенного стандарта. Проект GNU обеспечивает превосходную, свободную и широко применяемую реализацию под названием GnuPG, которую можно найти по адресу `gnupg.org`. Для ясности, мы будем называть эту систему просто PGP, даже если ее отдельные реализации имеют другие названия.

Вероятно, PGP — наиболее популярная криптографическая программа. К сожалению, для того чтобы работать с ней в системе UNIX/Linux, нужно быть специалистом в области криптографии. Пакет PGP может оказаться полезным, но мы не рекомендуем заставлять пользователей работать с ним из-за нетривиальности процесса обучения. Гораздо удобнее использовать Windows-версию PGP, чем команду **pgp** с ее 52 различными режимами работы.

Программные пакеты, распространяемые через Интернет, часто содержат файл сигнатуры PGP, подтверждающий их подлинность и целостность. Но проверить эти сигнатуры не так-то легко людям, не являющимся фанатиками PGP. Не потому, что процесс проверки сложен, а потому, что при использовании PGP истинная безопасность достигается только путем создания персональной коллекции открытых ключей тех людей, которые были проверены напрямую. Загрузка дистрибутива с открытым ключом и файлом сигнатуры безопасна примерно в той же степени, что и загрузка одного дистрибутива.

Некоторые почтовые клиенты, такие как Mozilla Thunderbird, имеют надстройки, обеспечивающие простой графический пользовательский интерфейс для зашифрованных входящих и исходящих сообщений. Программа Enigmail для клиента Mozilla Thunderbird может даже выполнять поиск баз открытых ключей в сети, если ключ для вашего получателя еще не был загеристрирован. Подробности можно найти на сайте `enigmail.mozdev.org`.

## SSH: безопасная оболочка

Система SSH (Secure Shell) является безопасной заменой утилит **rlogin**, **rcp** и **telnet**. Она использует криптографическую аутентификацию для подтверждения личности пользователя и шифрует любые соединения между двумя компьютерами. Протокол, используемый системой SSH, разрабатывался для противодействия ряду возможных атак. Он уже подробно описан в документах RFC4250–4256 и предложен организацией IETFв качестве стандарта.

Пакет SSH трансформировался из свободно распространяемой открытой системы (SSH1) в коммерческий продукт с немного иным (более надежным) протоколом (SSH2). К счастью, сообщество разработчиков открытых систем взяло ситуацию под контроль и выпустило пакет OpenSSH (поддерживаемый операционной системой OpenBSD), в котором реализованы оба протокола.

Основными компонентами пакета SSH являются серверный демон **sshd** и две утилиты пользовательского уровня: **ssh** предназначена для удаленной регистрации и **scp** — для копирования файлов. Среди остальных компонентов отметим команду **ssh-keygen**, генерирующую пары открытых ключей, и несколько утилит, необходимых для поддержки безопасных серверов X Windows.

Демон **sshd** способен аутентифицировать пользователей различными способами. Выбор остается за администратором.

- **Метод А**. Если имя удаленного компьютера, с которого регистрируется пользователь, указано в файле **~/.rhosts**, **~/.shosts**, **/etc/hosts.equiv** или **/etc/shosts.equiv**, то пользователь автоматически получает доступ в систему без проверки пароля. Подобная схема моделирует работу старого демона **rlogind** и, по нашему мнению, неприемлема для широкого применения.
- **Метод Б**. В дополнение к методу А, демон **sshd** может применять шифрование с открытым ключом для проверки адреса удаленного компьютера. Для того чтобы это произошло, открытый ключ компьютера (генерируется при инсталляции пакета) должен находиться в файле **/etc/ssh_known_hosts** локального узла или в файле **~/.ssh/known_hosts** пользователя. Если удаленный компьютер предоставляет соответствующий секретный ключ (обычно хранится в файле **/etc/ssh_host_key**, который недоступен для чтения рядовым пользователям), то пользователю разрешается войти в систему без проверки пароля.

  Мы полагаем, что данный метод сильнее, чем метод А, но все же недостаточно безопасен. Если удаленный компьютер взломан, локальная система также окажется под угрозой.
- **Метод В**. Демон **sshd** может применять шифрование с открытым ключом для идентификации самого пользователя. На этапе регистрации пользователь должен иметь доступ к файлу своего секретного ключа и предоставить пароль для его дешифровки. Ключ можно также создать без пароля, что является разумным решением для автоматической регистрации из удаленных систем.

  Этот метод самый безопасный, но он утомляет пользователей. Кроме того, он делает невозможной регистрацию в системе мобильных пользователей, если только они не возят с собой копию файла с секретным ключом (возможно, на USB-ключе, который должен быть зашифрован).

  Если вы решили использовать пару ключей, широко используйте команду **ssh -v** в процессе поиска ошибок.
- **Метод Г**. Наконец, демон **sshd** может просто попросить пользователя ввести свой регистрационный пароль. Это напоминает утилиту **telnet**, за исключением того, что пароль и все данные, передаваемые в ходе сеанса, подвергаются шифрованию. Основной недостаток этого метода заключается в том, что пароли оказываются относительно слабыми (часто их длина ограничена 8 символами) и есть готовые программы (например, **crack**) для взлома таких паролей. Тем не менее этот метод, как правило, лучше всего подходит для повседневного использования.

Правила аутентификации задаются в файле **/etc/sshd_config**. Он уже заполнен разного рода конфигурационным "мусором", большую часть которого можно проигнорировать. Параметры, касающиеся аутентификации, перечислены в табл. 22.2.

**Таблица 22.2. Параметры аутентификации, задаваемые в файле /etc/sshd_config**

| Параметр | Метод[а] | По умолчанию | Назначение |
|---|---|---|---|
| RhostsAuthentication | А | no | Разрешает регистрацию через файлы **~/.shosts**, **/etc/shosts.equiv** и т.п. |
| RhostsRSAAuthentication | Б | yes | Разрешает регистрацию через файл **~/.shosts** и другие, но также требует от компьютера предоставить ключ |
| IgnoreRhosts | А, Б | no | Задает игнорирование файлов **~/.rhosts** и **hosts.equiv** [б] |
| IgnoreRootRhosts | А, Б | no[в] | Запрещает аутентификацию пользователя **root** через файлы **.rhosts** и **.shosts** |
| RSAAuthentication | В | yes | Разрешает аутентификацию пользователей по методу шифрования с открытым ключом |
| PasswordAuthentication | Г | yes | Разрешает использование обычных регистрационных паролей |

[а] Методы аутентификации, к которым относится параметр.

[б] Файлы **~/.shosts** и **shosts.equiv** продолжают использоваться.

[в] По умолчанию равен значению параметра IgnoreRhosts.

Рекомендуемая нами конфигурация, в которой разрешаются методы В и Г, но не А и Б, такова.

```
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication yes
```

Никогда не следует разрешать суперпользователю регистрироваться в удаленном режиме. Доступ суперпользователя должен осуществляться только с помощью команды **sudo**. Для того чтобы установить этот режим, используйте следующий параметр.

```
PermitRootLogin no
```

Когда вы впервые устанавливаете соединение с новой системой посредством протокола SSH, вам предложат принять открытый ключ для доступа к удаленному узлу (который обычно генерируется в процессе инсталляции системы OpenSSH или сразу после этого). Настоящий параноик может проверить его вручную, но большинство из нас слепо доверяют этому ключу и сохраняют его в файле **~/.ssh/known_hosts**. Протокол SSH больше не вспоминает о серверном ключе, пока он не изменится. К сожалению, небрежность пользователей по отношению к ключам новых систем делает их уязвимыми для атак злоумышленников, если ключ на самом деле был предоставлен системой хакера.

Для устранения этого недостатка была изобретена запись DNS, получившая название SSHFP. В ее основе лежит предположение, что серверный ключ хранится в виде записи DNS. Когда клиент соединяется с неизвестной системой, протокол SSH ищет запись SSHFP, чтобы самостоятельно проверить ключ сервера, а не полагаться на пользователя.

Утилита **sshfp**, доступная на сайте xelerance.com/software/sshfp, генерирует записи SSHFP DNS, либо сканируя удаленный сервер, либо выполняя анализ ранее

принятого ключа из файла **known_hosts**. (Разумеется, в любом случае предполагается, что источник ключа известен и заслуживает доверия.) Использование этой утилиты не представляет трудностей: для генерирования ключа на основе сканирования сети применяется флаг **-s**, а для анализа файла **known_hosts** — флаг **-k** (по умолчанию). Например, следующая команда генерирует BIND-совместимую запись SSHFP для домена solaris.booklab.atrust.com.

```
solaris$ sshfp solaris.booklab.atrust.com
solaris.booklab.atrust.com IN SSHFP 1 1 94a26278ee713a37f6a78110f1ad9bd...
solaris.booklab.atrust.com IN SSHFP 2 1 7cf72d02e3d3fa947712bc56fd0e0a3i...
```

Добавьте эти записи в зонный файл домена (будьте осторожны с именами и командой $ORIGIN), перезагрузите домен и примените команду **dig** для верификации ключа.

```
solaris$ dig solaris.booklab.atrust.com. IN SSHFP | grep SSHFP
; <<>> DiG 9.5.1-P2 <<>> solaris.booklab.atrust.com. IN SSHFP
; solaris.booklab.atrust.com. IN SSHFP
solaris.booklab.atrust.com. 38400 IN SSHFP 1 1 94a26278ee713a37f6a78110f...
solaris.booklab.atrust.com. 38400 IN SSHFP 2 1 7cf72d02e3d3fa947712bc56f...
```

По умолчанию утилита **ssh** не проверяет записи SSHFP. Для того чтобы заставить ее сделать это, добавьте параметр VerifyHostKeyDNS в файл конфигурации **/etc/ssh/ssh_config**. Как и в большинстве случаев, связанных с использованием параметров клиента SSH, при первом обращении к системе вы можете также указать в командной строке аргумент **-o "VerifyHostKeyDNS yes"**.

Утилита SSH имеет много вспомогательных функций, полезных для системных администраторов. Одна из них позволяет безопасно туннелировать соединения TCP с помощью зашифрованного канала SSH, тем самым разрешая устанавливать соединения с небезопасными или защищенными брандмауэрами службами на удаленных сайтах. Эта функциональная возможность повсеместно предоставляется клиентам протокола и допускает простое конфигурирование. На рис. 22.1 показано типичное использование туннеля SSH. Этот рисунок должен помочь разобраться в принципах его функционирования.

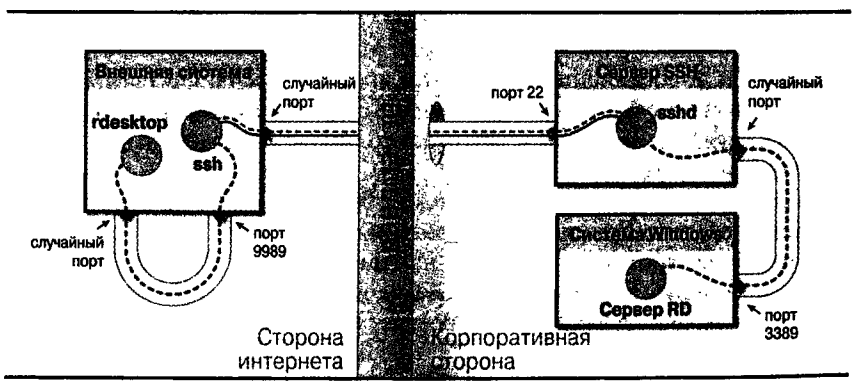



*Рис. 22.1. Туннель SSH для RDP-соединения*

В этом сценарии удаленный пользователь желает установить RDP-соединение (удаленная настольная система) с системой Windows, входящей в корпоративную сеть. Доступ к этому узлу или порту 3389 блокируется брандмауэром, но поскольку пользователь имеет доступ по протоколу SSH, он может маршрутизировать свое соединение через сервер SSH.

Для того чтобы установить соединение через сервер SSH, пользователь должен зарегистрироваться на удаленном сервере SSH с помощью команды **ssh**. В командной строке **ssh** он должен указать произвольный, но конкретный локальный порт (в данном случае 9989), на который утилита **ssh** должна перенаправить безопасный туннель к порту 3389 удаленной Windows-машины. (В стандартной реализации системы OpenSSH для этого достаточно задать аргумент **-L** *локальный_порт:удаленный_узел:удаленный_порт*.) Все порты источника в данном примере помечены как случайные, потому что выбор произвольного порта, с которым необходимо установить соединение, производится программой.

Для доступа к настольной Windows-машине пользователь должен открыть удаленного настольного клиента (в данном случае **rdesktop**) и ввести localhost:9989 как адрес сервера, с которым он хочет установить соединение. Локальная утилита **ssh** получит соединение с портом 9989 и туннелирует трафик по существующему соединению через удаленный сервер **sshd**. В свою очередь, сервер **sshd** направляет соединение на узел Windows.

Разумеется, туннели, подобные этим, также могут оказаться преднамеренными или непреднамеренными лазейками. Системные администраторы должны использовать туннели с осторожностью и обязаны следить за неавторизованным и неправильным использованием этой возможности пользователями.

В последние годы протокол SSH стал целью регулярных атак на пароли методом перебора. Хакеры выполняют повторяющиеся попытки аутентификации как обычные пользователи, такие как root, joe или admin. Свидетельством таких атак являются регистрационные записи о сотнях и тысячах неудачных попыток зарегистрироваться. Для защиты от этих атак лучше всего отключить аутентификацию паролей. Пока, похоже, хакеры сосредоточились на порте 22, поэтому эффективной контрмерой является перенос сервера SSH на другой порт. Однако история показывает, что защита по принципу "безопасность через неясность" ("security through obscurity") редко бывает долговременной. Проверка на ваших системах может выявить слабые пароли, которые могут быть взломаны с помощью прямого перебора.

## Пакет Stunnel

Stunnel — это открытый пакет, написанный Михалом Тройнара (Michal Trojnara), который шифрует TCP-соединения аналогично протоколу SSH. Он использует протокол SSL (Secure Sockets Layer) для создания сквозных туннелей, передающих данные незашифрованной службе и обратно. Известно, что он хорошо работает с небезопасными службами, такими как Telnet, IMAP и POP.

Демон **stunnel** работает как на клиентской, так и на серверной системе. Локальный демон **stunnel** обычно принимает соединения на традиционный порт службы (например, порт 25 для протокола SMTP) и направляет их через протокол SSL к демону **stunnel** на удаленном узле. Демон **stunnel** на удаленном узле принимает соединение, расшифровывает входящие данные и направляет их на удаленный порт, на котором прослушивается сервер. Эта система позволяет незашифрованным службам обеспечивать конфиденциальность и целостность данных, гарантируемые шифрованием, не требуя при этом никаких изменений в программном обеспечении. Достаточно только сконфигурировать клиентское программное обеспечение для поиска служб на локальной системе, а не на сервере, который их предоставляет.

Протокол Telnet является хорошим примером этого метода, потому что он состоит из простого демона, прослушивающего отдельный порт. Для того чтобы установить соединение со службой Telnet с помощью утилиты **stunnel**, сначала необходимо создать сер-

тификат SSL. Утилита Stunnel не зависит от библиотеки SSL, поэтому это можно сделать в любой стандартной операционной системе; мы предпочитаем систему OpenSSL. Для генерирования сертификата необходимо выполнить следующую команду.

```
server$ sudo openssl req -new -x509 -days 365 -nodes -out stunnel.pem
  -keyout stunnel.pem
Generating a 1024 bit RSA private key
.++++++
.............................++++++
writing new private key to 'stunnel.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Colorado
Locality Name (eg, city) [Newbury]:Boulder
Organization Name (eg, company) [My Company Ltd]:Booklab, Inc.
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or server's hostname) []:server.example.com
Email Address []:
```

Эта команда создает самоподписанный и беспарольный сертификат. С одной стороны, отсутствие пароля — это удобно (реальный человек не должен вводить пароль при каждом новом запуске утилиты **stunnel**), но, с другой стороны, возникает определенный риск нарушения системы безопасности. Защитите файл сертификата строгими ограничениями.

Далее следует определить конфигурацию утилиты **stunnel** для сервера и клиента. Стандартный конфигурационный файл называется **/etc/stunnel/stunnel.conf**, но вы можете создать несколько конфигураций, если хотите установить несколько туннелей.

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/run/stunnel/
pid = /stunnel.pid
setuid = nobody
setgid = nobody
debug = 7
output = /var/log/stunnel.log
client = no
[telnets]
accept = 992
connect = 23
```

В конфигурации сервера есть несколько важных аспектов. Инструкция `chroot` ограничивает процесс **stunnel** каталогом `/var/run/stunnel`. Пути для вспомогательных файлов должны быть выражены либо в обычном системном пространстве имен, либо в пространстве имен, ограниченном командой **chroot**. Это зависит от точки, в которой они открываются. В данном случае файл **stunnel.pid** фактически расположен в каталоге **/var/run/stunnel**.

Раздел `[telnets]` содержит две инструкции: инструкция `accept` приказывает утилите **stunnel** принимать соединения на порту 992, а инструкция `connect` пропускает эти соединения через порт 23, т.е. через реальную службу Telnet.

Конфигурация клиента является аналогичной.

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/run/stunnel/
pid = /stunnel.pid
setuid = nobody
setgid = nobody
debug = 7
output = /var/log/stunnel.log
client = yes

[telnets]
accept = 23
connect = server.example.com:992
```

Несколько директив являются обратными по отношению к конфигурации сервера. Инструкция `client = yes` приказывает программе выполнить инициацию соединений с помощью утилиты **stunnel**, а не принимать их. Локальная утилита **stunnel** прослушивает соединения на порту 23 и устанавливает соединения с сервером на порту 992. Имя узла в директиве `connect` должно совпадать с именем, указанным при создании сертификата.

Утилиты **stunnel** на клиентской и серверной сторонах могут запускаться без аргументов командной строки. Если вы выполните команду **netstat -an**, то увидите, что утилита **stunnel** на сервере ожидает соединений на порту 992, а клиентская утилита **stunnel** — на порту 23.

Для доступа к туннелю пользователь должен просто применить команду **telnet** к локальному узлу.

```
client$ telnet localhost 23
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
Red Hat Enterprise Linux WS release 4 (Nahant Update 2)
Kernel 2.6.9-5.EL on an i686
login:
```

Теперь пользователь может безопасно регистрироваться, не опасаясь кражи пароля. Бдительный администратор может использовать оболочки протокола TCP для ограничения соединений на клиентском компьютере только локальным интерфейсом, чтобы не позволить внешним пользователям устанавливать безопасные соединения службы **telnet** с сервером! Утилита **stunnel** — одна из немногих программ, которая имеет встроенную поддержку оболочки и не требует использования команды **tcpd** для ограничения доступа. Инструкции по ее использованию можно найти на сайте `stunnel.org`.

## 22.11. Брандмауэры

Помимо защиты отдельных компьютеров, можно предпринять меры для обеспечения безопасности на сетевом уровне. Основным инструментом сетевой защиты является брандмауэр, физическое устройство или часть программного обеспечения, не допускающие нежелательные пакеты в системы или сети. Брандмауэры в настоящее время используются всюду. Их можно обнаружить как в настольных компьютерах и серверах, так и в маршрутизаторах и корпоративном сетевом оборудовании.

## Брандмауэры, фильтрующие пакеты

Брандмауэр, фильтрующий пакеты, ограничивает виды трафика, проходящего через интернет-шлюз (или через внутренний шлюз, разделяющий домены в пределах организации), основываясь на информации, извлеченной из заголовков пакетов. Это напоминает прохождение таможни, когда вы на своем автомобиле пересекаете границу. Администратор указывает, какие целевые адреса, номера портов и типы протоколов являются допустимыми, а брандмауэр просто отбрасывает (в некоторых случаях также регистрирует) все пакеты, которые не соответствуют заданным критериям.

В системах семейства Linux фильтрация пакетов осуществляется с помощью утилиты **iptables**, в системах Solaris и HP-UX — с помощью программы IPFilter, а в системе AIX — с помощью команды **genfilt**. Подробно они будут рассмотрены чуть позже.

Несмотря на то что эти инструменты способны выполнять сложную фильтрацию и значительно повышают степень безопасности, в целом мы разочарованы использованием систем UNIX и Linux в качестве сетевых маршрутизаторов и особенно в качестве корпоративных маршрутизаторов, играющих роль брандмауэров. Сложность универсальных систем снижает их защищенность и надежность по сравнению со специальными устройствами. Для защиты корпоративных сетей лучше применять специальные брандмауэры, например устройства компании Check Point и Cisco.

## Принципы фильтрации служб

Большинству "известных" служб назначается сетевой порт в файле **/etc/services** или его системно-зависимом эквиваленте. Демоны, которые реализуют соответствующие службы, постоянно опрашивают порты, ожидая поступления запросов на подключение со стороны удаленных компьютеров[7]. В основном, такие порты являются "привилегированными". Это значит, что их номера находятся в диапазоне от 1 до 1023 и что порты могут использоваться только процессами, работающими от имени пользователя **root**. Порты с номерами 1024 и выше являются непривилегированными.

Фильтрация на уровне служб основана на предположении, что клиент (компьютер, инициирующий соединение по протоколу TCP или UDP) будет использовать непривилегированный порт для установления соединения с привилегированным портом сервера. Например, если компьютеру с адресом 192.108.21.200 нужно разрешить только входящие SMTP-соединения, то следует установить фильтр, который позволит посылать TCP-пакеты по этому адресу через порт 25 и отправлять TCP-пакеты с этого адреса через любой порт[8]. Способ инсталляции такого фильтра будет зависеть от типа используемого маршрутизатора.

Некоторые службы, например FTP, еще больше усложняют эту головоломку. При пересылке файла по протоколу FTP на самом деле устанавливается два соединения: для команд и для данных. Клиент инициирует командное соединение, а сервер — информационное.[9] Следовательно, если необходимо получать файлы из Интернета по протоколу FTP, следует разрешить вход в систему через все непривилегированные TCP-

[7] Во многих случаях задачу ожидания берут на себя демоны **inetd** или **xinetd**. Подробнее об этом рассказывается в разделе 32.1.

[8] Порт 25 — это порт SMTP, определенный в файле */etc/services*.

[9] Это относится к традиционному протоколу FTP, известному как "активный протокол FTP". Некоторые системы поддерживают "пассивный протокол FTP", в котором оба соединения инициируются клиентом.

порты, так как неизвестно, какой из них будет использован для установления информационного соединения.

🕮 Подробнее о конфигурировании FTP-сервера рассказывается в разделе 23.4.

Это сводит на нет преимущества фильтрации пакетов, поскольку некоторые общеизвестные службы, печально известные своей незащищенностью, работают с непривилегированными портами (например, служба X11 на порту 6000). Кроме того, любознательные пользователи получают возможность запускать собственные служебные программы (например, сервер `telnet` на нестандартном и непривилегированном порту), к которым они и их друзья могут обращаться из Интернета.

Самый безопасный способ использования фильтра пакетов — использовать протокол передачи данных SSH. В настоящее время в Интернете опубликован его черновик, но он уже широко используется и достиг достаточно высокой степени зрелости. Обычно он используется как компонент протокола SSH, обеспечивающего аутентификацию и шифрование. В отличие от протокола FTP, протокол SFTP использует только один порт и для команд, и для данных, разрешая парадокс фильтрации пакетов. Существует много реализаций протокола SFTP. Мы рекомендуем использовать клиентскую утилиту командной строки SFTP, поставляемую в системе OpenSSH.

Наиболее разумный способ решения проблем с протоколом FTP — разрешить выход по протоколу FTP во внешний мир только с этого изолированного компьютера. Пользователи смогут регистрироваться на нем, когда им понадобится выполнять сетевые операции, запрещенные во внутренней сети. Поскольку копирование всех пользовательских учетных записей на FTP-сервер противоречит сути административного деления в организации, можно создавать учетные записи только по заявкам. Естественно, на FTP-сервере должен быть инсталлирован полный комплект средств защиты.

В некоторых организациях, которые заботятся о безопасности, используется двухступенчатая фильтрация. Один фильтр в этой схеме подключен к Интернету как шлюз, а второй находится между этим шлюзом и остальной частью локальной сети. Идея состоит в том, чтобы оставить внешний шлюз относительно открытым, а внутренний — сделать очень консервативным. Если промежуточный компьютер административно отделен от остальной части сети, появляется возможность реализовать различные службы Интернета с меньшим риском. Частично защищенная сеть обычно называется "демилитаризованной зоной" (DMZ).

Наиболее безопасным способом использования фильтрации пакетов является настройка первоначальной конфигурации, которая не позволяет никаких входящих соединений. Затем можно постепенно ослабить фильтрацию, выяснив полезные службы, которые не работают, и перенеся все службы, доступные в Интернете, в демилитаризованую зону.

## Брандмауэры, осуществляющие инспекцию пакетов с отслеживанием состояния соединений

В основе инспекции пакетов с отслеживанием состояния соединений лежит концепция, которую можно описать с помощью следующего примера. Представьте себе, что вы ищете террориста в переполненном аэропорту, внимательно прислушиваясь ко всем разговорам, происходящим вокруг вас, и понимая их содержание (на всех языках). Брандмауэры, осуществляющие инспекцию пакетов с отслеживанием состояния соединений (stateful inspection firewalls), проверяют весь трафик, проходящий через них, и сравнивают его с текущей сетевой активностью в ожидании события, которое "должно" произойти.

Например, если пакеты, обмениваемые в ходе выполнения последовательности команд протокола FTP, называют порт, который впоследствии должен использоваться для информационного соединения, то брандмауэр должен проверить, что информационное соединение произойдет именно на этом порту. Попытки удаленного сайта соединиться с другими портами заранее считаются фальшивыми и должны пресекаться.

Так что же на самом деле предлагают разработчики под видом инспекции пакетов с учетом состояния протокола. Их продукты либо осуществляют мониторинг ограниченного количества соединений или протоколов, либо распознают конкретный набор "неблагоприятных" ситуаций. В этом нет ничего плохого. Очевидно, что из любой технологии, способной распознавать аномалии трафика, можно извлечь пользу. Однако в данном случае важно помнить, что все эти обещания *преимущественно* являются лишь рекламным ходом.

## Насколько безопасны брандмауэры

Фильтрация пакетов не должна быть основным средством защиты от хакеров. Это всего лишь вспомогательная мера. При наличии брандмауэра порой создается ложное впечатление, будто уровень безопасности является исключительно высоким. Если, доверившись брандмауэру, ослабить бдительность на других "рубежах", это *отрицательно* скажется на защищенности информации.

Каждый компьютер необходимо защищать индивидуально и регулярно проверять с помощью таких средств, как Bro, Snort, Nmap, Nessus и OSSEC. Следует также обучать пользователей правилам безопасной работы в системе. В противном случае вы просто создадите внешне устойчивую, но крайне запутанную внутри систему.

В идеале локальные пользователи должны иметь возможность подключаться к любой службе Интернета, но доступ к локальным службам со стороны интернет-узлов должен быть ограничен демилитаризованной зоной. Например, к серверу архивов может быть разрешен SFTP-доступ, а к почтовому серверу, получающему входящую почту, — доступ только по протоколу SMTP.

Для того чтобы интернет-канал работал с максимальной эффективностью, советуем все же обратить основное внимание на удобство работы и доступность. В конце концов, сеть становится безопасной благодаря бдительности системного администратора, а не благодаря брандмауэру.

# 22.12. Особенности брандмауэров в системе Linux

Обычно мы не рекомендуем использовать компьютер, работающий под управлением Linux (а также UNIX или Windows), в качестве брандмауэра, поскольку в целом операционная система не обеспечивает надлежащий уровень безопасности.[10] Однако лучше инсталлировать усиленную систему Linux, чем вообще ничего, если речь идет о домашнем компьютере или организации, бюджет которой не предусматривает покупку оборудования соответствующего уровня. В любом случае локальный фильтр, такой как команда **iptables**, может оказаться превосходным инструментом для обеспечения безопасности.

Устанавливая систему Linux в качестве брандмауэра, убедитесь, что она включает самые последние обновления и "заплаты", касающиеся брешей в системе защиты.

[10] И все же многие сетевые устройства, например маршрутизаторы компании Linksys, в своих ядрах используют систему Linux и ***iptables***.

Компьютер, функционирующий в качестве брандмауэра, — идеальное место для практической проверки всех рекомендаций, изложенных в этой главе. (В разделе 22.11 собраны сведения, касающиеся брандмауэров в целом. Читателям, не знакомым с концепцией брандмауэра, советуем предварительно прочитать приведенный там материал.)

## Правила, цепочки и таблицы

В ядре Linux версии 2.4 появился новый механизм фильтрации пакетов — Netfilter. Контролирующая его работу команда **iptables** является "наследницей" более старой команды **ipchains**, которая использовалась в ядрах версии 2.2. В команде **iptables** применяется концепция упорядоченной *цепочки* правил, согласно которым проверяются сетевые пакеты. Наборы цепочек образуют *таблицы*, предназначенные для обработки определенных видов трафика.

К примеру, стандартная таблица для команды **iptables** называется **filter**. Цепочки правил этой таблицы используются для фильтрации сетевых пакетов. В таблице **filter** содержатся три стандартные цепочки: FORWARD, INPUT и OUTPUT. Каждый пакет, обрабатываемый ядром, поступает на проверку в одну цепочку.

Правила в цепочке FORWARD применяются ко всем пакетам, которые поступают в один интерфейс и перенаправляются в другой. Правила в цепочках INPUT и OUTPUT предназначены для пакетов, адресованных локальному компьютеру или исходящих от него соответственно. Обычно этих трех цепочек вполне достаточно для фильтрации данных между двумя интерфейсами. В случае необходимости можно определять более сложные конфигурации.

Помимо таблицы **filter**, существуют также таблицы **nat** и **mangle**. В первой содержатся цепочки правил, предназначенные для управления системой NAT (Network Address Translation — трансляция сетевых адресов). Подробнее об этой системе рассказывалось в разделе 14.4, а в разделе 14.12 приводился пример использования таблицы **nat**. В этом разделе мы воспользуемся цепочкой PREROUTING данной таблицы для фильтрации пакетов с поддельными IP-адресами.

Таблица **mangle** содержит цепочки, предназначенные для модификации содержимого сетевых пакетов вне контекста системы NAT или механизма фильтрации пакетов. Эта таблица редко применяется в крупных системах, поэтому мы не будем ее рассматривать.

## Целевые директивы для правил

В каждом правиле цепочки имеется целевая директива, определяющая, что следует делать с пакетами, подчиняющимися этому правилу. Если пакет прошел проверку, его судьба предрешена и другие проверки не выполняются. В качестве директивы разрешается указывать имя другой цепочки.

Правила таблицы **filter** могут содержать следующие целевые директивы: ACCEPT, DROP, REJECT, LOG, MIRROR, QUEUE, REDIRECT, RETURN и ULOG. Директива ACCEPT разрешает пакету следовать своим маршрутом. Директивы DROP и REJECT запрещают пропускать пакет, но первая приводит к "безмолвному" удалению пакета, а вторая — к выдаче ICMP-сообщения об ошибке. Директива LOG позволяет следить за тем, какие правила применяются к пакетам, а директива ULOG включает режим расширенной регистрации.

Директива REDIRECT перенаправляет пакеты прокси-серверу. Она удобна, когда весь трафик веб-узла должен проходить через кеширующую программу, такую как Squid. Директива RETURN объявляет конец пользовательской цепочки. Директива MIRROR меняет

местами IP-адреса отправителя и получателя перед отправкой пакета. Наконец, директива QUEUE передает пакеты локальным программам пользователя через модуль ядра.

## Настройка команды iptables в качестве брандмауэра

Для того чтобы можно было использовать команду **iptables**, следует включить режим перенаправления IP-пакетов и убедиться в том, что многочисленные модули команды загружены в ядро. О том, как включить указанный режим, рассказывалось в разделах 13.3 и 14.12. Во всех дистрибутивах, где есть команда **iptables**, есть и сценарии загрузки ее модулей.

В системе Linux брандмауэр обычно реализуется в виде последовательности команд **iptables**, содержащихся в одном из сценариев запуска системы. Отдельные команды **iptables**, как правило, имеют одну из следующих форм.

```
iptables -F имя_цепочки
iptables -P имя_цепочки директива
iptables -A имя_цепочки -i интерфейс -j директива
```

В первом случае (**-F**) из цепочки удаляются все предыдущие правила. Во втором случае (**-P**) задается стандартная директива цепочки. Мы рекомендуем по умолчанию использовать директиву DROP. В третьем случае (**-A**) заданное правило добавляется к цепочке. Если не указан флаг **-t**, команда применяется к цепочкам таблицы **filter**. Опция **-i** определяет интерфейс, для которого это правило действует. Опция **-j** задает директиву. Команда **iptables** понимает также ряд других опций (табл. 22.3).

**Таблица 22.3. Дополнительные флаги для фильтров команды iptables**

| Опция | Назначение |
|---|---|
| **-p** *протокол* | Соответствие протоколу: **tcp**, **udp** или **icmp** |
| **-s** *исходный_адрес* | Соответствие исходному IP-адресу узла или сети (допускается нотация CIDR) |
| **-d** *целевой_адрес* | Соответствие целевому IP-адресу узла или сети |
| **--sport** *номер_порта* | Соответствие номеру исходного порта (обратите внимание на двойной дефис) |
| **--dport** *номер_порта* | Соответствие номеру целевого порта (обратите внимание на двойной дефис) |
| **--icmp_type** *тип* | Соответствие типу ICMP-сообщения (обратите внимание на двойной дефис) |
| **!** | Инверсия смысла опции |
| **-t** | Задает таблицу, к которой применяется команда (по умолчанию — **filter**) |

## Полный пример

Ниже мы детально разберем конкретный пример. Предполагается, что интерфейс eth1 подключен к Интернету, а интерфейс eth0 — к локальной сети. IP-адрес интерфейса eth1 равен 128.138.101.4, а интерфейса eth0 — 10.1.1.254. В обоих случаях маска сети равна 255.255.255.0. В этом примере выполняется фильтрация пакетов без учета состояния для защиты веб-сервера с IP-адресом 10.1.1.2. Это стандартный метод защиты веб-серверов. В конце примера будет продемонстрирована фильтрация пакетов с учетом состояния для защиты настольных компьютеров.

Нашим первым действием будет инициализация таблицы **filter**. Сначала все цепочки таблицы очищаются, после чего для цепочек INPUT и OUTPUT назначается стандартная директива DROP. Как и в других подобных случаях, наиболее безопасная стратегия заключается в удалении всех пакетов, которые не упомянуты явно.

```
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

Поскольку правила проверяются по порядку, поместим наиболее часто используемые вперед[11]. Первые правила цепочки FORWARD разрешают устанавливать соединения с сетевыми службами по адресу 10.1.1.2. В частности, разрешаются протоколы SSH (порт 22), HTTP (порт 80) и HTTPS (порт 443). Самое первое правило пропускает все пакеты, исходящие из внутренней сети.

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 443 -j ACCEPT
```

Единственный TCP-трафик, который может приниматься узлом, — это трафик протокола SSH, с помощью которого можно управлять брандмауэром. Второе из показанных ниже правил разрешает трафик интерфейса обратной связи внутри брандмауэра. Наши администраторы начинают нервничать, если им не удается с помощью команды **ping** найти стандартный маршрут, поэтому третье правило пропускает ICMP-пакеты ECHO_REQUEST, имеющие локальный IP-адрес отправителя.

```
iptables -A INPUT -i eth0 -d 10.1.1.1 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i lo -d 127.0.0.1 -p ANY -j ACCEPT
iptables -A INPUT -i eth0 -d 10.1.1.1 -p icmp --icmp-type 8 -j ACCEPT
```

Компьютеру, имеющему выход в Интернет, для нормальной работы необходимо, чтобы брандмауэр пропускал определенные типы ICMP-пакетов. Следующие восемь правил пропускают минимальный набор ICMP-пакетов как на сам брандмауэр, так и в защищаемую им сеть.

```
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 5 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 0 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 3 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 5 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 11 -j ACCEPT
```

Теперь добавим правила в цепочку PREROUTING таблицы **nat**. Эти правила не связаны с фильтрацией пакетов, зато полезны для защиты от поддельных IP-адресов. Правила с директивой DROP не нужно дублировать в цепочках INPUT и FORWARDING, поскольку цепочка PREROUTING применяется ко всем пакетам, поступающим на брандмауэр.

```
iptables -t nat -A PREROUTING -i eth1 -s 10.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 172.16.0.0/12 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 192.168.0.0/16 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 127.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 224.0.0.0/4 -j DROP
```

Наконец, завершим цепочки INPUT и FORWARD правилами, которые запрещают все пакеты, не упомянутые явно. Вообще-то, это уже было сделано с помощью команд **iptables -P**, но директива LOG позволяет увидеть, кто к нам "стучится" из Интернета.

[11] Следите за тем, чтобы стремление к повышению производительности не нарушало логику выполнения правил.

```
iptables -A INPUT -i eth1 -j LOG
iptables -A FORWARD -i eth1 -j LOG
```

При желании можно включить систему NAT, чтобы скрывать частные адреса, используемые во внутренней сети. Подробнее об этом рассказывалось в разделе 14.12.

Одной из наиболее интересных возможностей механизма Netfilter является фильтрация пакетов с учетом состояния. Вместо того чтобы пропускать входящий трафик определенных типов, брандмауэр пропускает трафик, выдаваемый в ответ на клиентские запросы. Показанная ниже простейшая цепочка разрешает всем пакетам покидать сеть, но принимаются лишь те пакеты, которые поступают в рамках соединений, установленных узлами нашей сети.

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Для того чтобы команда **iptables** могла отслеживать сложные сетевые сеансы, например сеансы протоколов FTP и IRC, должны быть загружены определенные модули ядра. В противном случае команда просто не будет позволять устанавливать соответствующие соединения. Фильтрация пакетов с учетом состояния способна повысить безопасность системы, но она резко повышает и сложность управления сетью. Убедитесь в том, что она действительно нужна, прежде чем реализовать ее на брандмауэре.

Для отладки цепочек правил, вероятно, лучше всего подходит команда **iptables -L -v**. Она сообщит, сколько раз для того или иного правила было найдено совпадение. Мы часто добавляем временные правила с директивой LOG, если хотим получить дополнительную информацию о проверке пакетов. Более сложные проблемы решаются с помощью анализатора пакетов, такого как **tcpdump**.

# 22.13. Модуль IPFilter для систем UNIX

Большинство разработчиков системы UNIX не имеют собственного программного обеспечения для брандмауэров.[12] Однако его достаточно легко добавить: технологию NAT и функции брандмауэра с отслеживанием состояния соединений для систем UNIX реализует открытый пакет IPFilter, разработанный Дарреном Ридом (Darren Reed). Система Solaris включает его по умолчанию. Кроме того, он доступен в виде надстроек для систем HP-UX, AIX и многих других, включая Linux. Пакет IPFilter можно использовать как загружаемый модуль ядра (что рекомендуется разработчиками) или включать в ядро статически.

Пакет IPFilter является зрелым и полнофункциональным программным обеспечением. Он имеет активное сообщество его пользователей и постоянно совершенствуется. Этот пакет способен отслеживать состояние соединений даже для протоколов без запоминания состояния, таких как UDP и ICMP.

Пакет IPFilter считывает правила фильтрации из файла конфигурации (обычно **/etc/ipf.conf** или **/etc/ipf/ipf.conf**), не заставляя пользователя выполнять несколько таких команд, как **iptables**. Рассмотрим простое правило, которое может встретиться в файле **ipf.conf**.

```
block in all
```

[12] Исключением является компания IBM. Система AIX включает отдельное программное обеспечение для фильтрации пакетов IP Security, хотя оно не выполняет фильтрацию с учетом состояния протокола. Для начала изучите справочную страницу ***genfilt***.

Это правило блокирует весь входящий трафик (т.е. сетевые команды, полученные системой) на всех сетевых интерфейсах. Определенно, это безопасно, но не слишком полезно!

В табл. 22.4 перечислены некоторые возможные условия, которые могут упоминаться в правилах `ipf`.

**Таблица 22.4. Типичные условия ipf**

| Условие | Описание |
|---|---|
| `on` *интерфейс* | Применяет правило к конкретному интерфейсу |
| `proto` *протокол* | Выбирает пакет в соответствии с протоколом: `tcp`, `udp` или `icmp` |
| `from` *ip_источника* | Фильтрует по источнику: `host`, `network` или `any` |
| `to` *ip_получателя* | Фильтрует по получателю: `host`, `network` или `any` |
| `port = `*номер_порта* | Фильтрует по имени порта (из /etc/services) или номеру[a] |
| `flags` *флаг* | Фильтрует по флагам заголовка TCP |
| `icmp-type` *число* | Фильтрует по типу и коду ICMP |
| `keep state` | Сохраняет подробную информацию о сеансе; см. комментарии ниже |

[a] Можно использовать любой оператор сравнения: =, <, >, <=, >=, etc.

Пакет IPFilter оценивает правила в том порядке, в котором они представлены в файле конфигурации. *Последнее* совпадение является привязкой. Например, входящие пакеты, передаваемые через следующий фильтр, всегда проходят через него.

```
block in all
pass in all
```

Правило `block` применяется ко всем пакетам, как и правило `pass`, но при этом правило `pass` проверяется последним. Для того чтобы немедленно применить одно правило и пропустить остальные, следует использовать ключевое слово `quick`.

```
block in quick all
pass in all
```

Промышленные брандмауэры обычно состоят из многих правил, поэтому для поддержания их эффективной работы важно свободно пользоваться ключевым словом `quick`. Без этого каждый пакет будет оценивать каждое правило, что приведет к неоправданным затратам времени.

Возможно, чаще всего брандмауэры используются для управления доступом к конкретной сети или узлу и контроля за их обращениями к другим сетям и узлам, часто по отношению к конкретному порту. Пакет IPFilter имеет мощный синтаксис для управления трафиком на данном уровне детализации. В следующих правилах разрешается входящий трафик в сеть 10.0.0.0/24 на TCP-порты 80 и 443 и UDP-порт 53.

```
block out quick all
pass in quick proto tcp from any to 10.0.0.0/24 port = 80 keep state
pass in quick proto tcp from any to 10.0.0.0/24 port = 443 keep state
pass in quick proto udp from any to 10.0.0.0/24 port = 53 keep state
block in all
```

Ключевые слова `keep state` заслуживают особого внимания. Пакет может отслеживать соединения, отмечая первый пакет в новом сеансе. Например, когда новый пакет поступает на порт 80 в узле 10.0.0.10, программа IPFilter делает запись в таблице состояний и пропускает пакет в сеть. Он также позволяет веб-серверу ответить, даже если

первое правило явно блокирует весь исходящий трафик. Ключевые слова `keep state` также полезны для устройств, которые не предоставляют никаких услуг, но обязаны инициировать соединения. Следующий набор правил разрешает любую передачу данных, инициированную узлом 192.168.10.10. Он блокирует все входящие пакеты, за исключением пакетов, связанных с соединениями, которые уже были инициированы.

```
block in quick all
pass out quick from 192.168.10.10/32 to any keep state
```

Ключевые слова `keep state` относятся и к пакетам UDP и ICMP, но поскольку эти протоколы не запоминают состояния, их механизм довольно необычен: программа IPFilter разрешает ответы на UDP- и ICMP-пакет в течение 60 секунд после того, как входящий пакет был обнаружен фильтром. Например, если UDP-пакет, пришедший от узла 10.0.0.10 с порта 32000, адресован узлу 192.168.10.10 на порт 53, то ответ по протоколу UDP от узла 192.168.10.10 будет разрешен в течение 60 секунд после прохождения через порт. Аналогично ICMP-эхо (ответ на команду **ping**) разрешается после того, как запрос эха был занесен в таблицу состояний.

🕮 Частные адреса и технология NAT описаны в разделе 14.4.

Трансляция сетевых адресов (Network address translation — NAT) — это еще одна функциональная возможность, предлагаемая пакетом IPFilter. Система NAT позволяет крупной сети, использующей частные адреса RFC1918, соединяться с Интернетом с помощью небольшого набора интернет-маршрутизируемых IP-адресов. Устройство NAT отображает трафик, исходящий из частной сети, в один или несколько открытых адресов, посылает запросы через Интернет, а затем перехватывает ответы и переписывает их в терминах локальных IP-адресов.

Для выполнения функций NAT пакет IPFilter использует ключевое слово `map` (вместо `pass` и `block`). В следующем правиле трафик, исходящий из сети 10.0.0.0/24, отображается в текущий маршрутизируемый адрес на интерфейсе `e1000g0`.

```
map e1000g0 10.0.0.0/24 -> 0/32
```

Если адрес интерфейса `e1000g0`, то фильтр должен быть перезагружен, поскольку может оказаться, что интерфейс `e1000g0` арендует динамические IP-адреса по протоколу DHCP. По этой причине функциональные возможности NAT в пакете IPFilter лучше всего использовать на сайтах, имеющих статические адреса на интерфейсе, имеющем выход в Интернет.

Правила пакета IPFilter являются гибкими и настраиваемыми. С помощью дополнительных возможностей, таких как макросы, их можно значительно упростить. Подробности, относящиеся к этим дополнительным возможностям, можно найти на официальном сайте IPFilter `coombs.anu.edu.au/~avalon`.

Пакет IPFilter включает несколько команд, перечисленных в табл. 22.5.

**Таблица 22.5. Команды пакета IPFilter**

| Команда | Функция |
|---|---|
| **ipf** | Управляет списками правил и фильтров |
| **ipfstat** | Получает статистическую информацию о фильтрации пакетов |
| **ipmon** | Отслеживает зарегистрированную информацию о фильтре |
| **ipnat** | Управляет правилами NAT |

Среди команд, перечисленных в табл. 22.5, чаще всего используется команда **ipf**. Она получает на вход файл правил и добавляет правильно сформулированные правила в список фильтров в ядре. Если не указан аргумент **-Fa**, стирающий все существующие правила, команда **ipf** добавляет правила в конец фильтра. Например, для того чтобы стереть существующий набор фильтров в ядре и загрузить правила из файла **ipf.conf**, следует выполнить следующую команду.

```
solaris$ sudo ipf -Fa -f /etc/ipf/ipf.conf
```

Для управления доступом пакет IPFilter использует файлы псевдоустройств из каталога **/dev**, и по умолчанию только пользователь **root** может редактировать список фильтров. Мы рекомендуем не изменять разрешения, установленные по умолчанию, а для управления фильтром использовать команду **sudo**.

При загрузке файла для отладки синтаксических ошибок и выявления других, связанных с конфигурацией, используйте флаг **-v** команды **ipf**.

Пакет IPFilter изначально инсталлирован в ядре операционной системы Solaris, но перед использованием его необходимо активизировать с помощью следующей команды.

```
solaris$ sudo svcadm enable network/ipfilter
```

## 22.14. Виртуальные частные сети

В простейшем виде виртуальная частная сеть (Virtual Private Networks — VPN) — это специальный тип сетевого соединения, эмулирующего непосредственное подключение удаленной сети, даже если в действительности она находится за тысячу километров и скрыта за множеством маршрутизаторов. Для повышения безопасности соединение не только подвергается аутентификации в том или ином виде (обычно с использованием совместного секретного ключа, например пароля), но еще и шифруется. Это называется *защищенным туннелем*.

Приведем пример ситуации, в которой сеть VPN оказывается очень удобной. Предположим, компания имеет офисы в нескольких городах: Чикаго, Боулдере и Майами. Если каждый офис подключен к локальному провайдеру Интернета, то посредством виртуальных частных сетей компания может прозрачным образом (и, главное, весьма безопасно) соединить все офисы через такую ненадежную сеть, как Интернет. Конечно, такого же результата можно достичь и за счет покупки выделенных линий, но это обойдется гораздо дороже.

В качестве еще одного примера можно взять компанию, служащие которой выходят в сеть из дому. Наличие виртуальных частных сетей позволит служащим пользоваться преимуществами своих высокоскоростных и недорогих кабельных модемов и в то же время работать так, будто они непосредственно подключены к корпоративной сети.

Благодаря удобству и популярности такого подхода, многие компании стали предлагать решения, связанные с сетью VPN. Функции поддержки VPN могут быть реализованы в маршрутизаторе, в операционной системе и даже в специализированном сетевом устройстве. Если позволяет бюджет, рассмотрите коммерческие предложения, в избытке имеющиеся на рынке.

Если же вы ограничены в средствах, обратитесь к пакету SSH, который позволяет организовать защищенные туннели (раздел 22.10).

## Туннели IPSEC

Те, кого интересует надежное и недорогое решение в области VPN, должны обратить внимание на IPSEC (Internet Protocol Security — механизм защиты протокола IP). Изначально он был реализован для протокола IPv6, но теперь доступен также и для IPv4. IPSEC — это одобренная организацией IETF система аутентификации и шифрования соединений. Практически все серьезные поставщики VPN-систем оснащают свои продукты по крайней мере режимом совместимости с IPSEC. Системы Linux, Solaris, HP-UX и AIX содержат поддержку технологии IPSec, встроенную в ядро.

Для аутентификации и шифрования механизм IPSec использует сильные криптографические алгоритмы. Аутентификация гарантирует, что пакеты приходят от легального отправителя и по дороге не были искажены, а шифрование предотвращает несанкционированный просмотр содержимого пакета.

В туннельном режиме система шифрует заголовок транспортного уровня, содержащий номера портов отправителя и получателя. К сожалению, этот механизм напрямую конфликтует со способом работы большинства брандмауэров. По этой причине в большинстве современных реализаций по умолчанию используют транспортный режим, в котором шифруется только содержимое пакетов (т.е. передаваемые данные).

Существует одна тонкость, связанная с туннелями IPSEC и параметром MTU передаваемых пакетов. Важно убедиться в том, что пакет, зашифрованный средствами IPSEC, не подвергается фрагментации при прохождении туннеля. Для этого может потребоваться снизить значение MTU для сетевых интерфейсов, расположенных перед туннелем (обычно подходит значение 1400 байт). Подробнее о параметре MTU рассказывалось в разделе 14.2.

## Так ли уж нужны виртуальные частные сети

К сожалению, у виртуальных частных сетей есть и недостатки. Они позволяют организовать достаточно надежный туннель через ненадежный канал связи между двумя конечными точками, но они, как правило, не защищают сами конечные узлы. Например, если создать сеть VPN между корпоративной магистралью и домашним компьютером президента компании, то можно ненароком открыть удобную лазейку для 15-летнего вундеркинда, по ночам наведывающегося в папин кабинет. Хорошо, если он использует эту возможность лишь для игр по сети с одноклассниками. А если нет?

Отсюда вывод: соединения, устанавливаемые через туннели VPN, следует рассматривать как внешние и предоставлять им дополнительные привилегии лишь в случае крайней необходимости, тщательно взвесив все за и против. Можно добавить в свод правил безопасности, принятый в организации, специальный пункт, касающийся работы в сети VPN.

# 22.15. Сертификация и стандарты

Если содержание этой главы ошеломило вас, не беспокойтесь. Компьютерная безопасность — сложная и необъятная тема, которой посвящены бесчисленные книги, вебсайты и журналы. К счастью, для оценки и организации существующей информации было сделано немало. Существуют десятки стандартов и сертификаций, и разумный администратор должен их знать.

Один из основных философских принципов информационной безопасности неформально называется “триадой CIA”.

Эта аббревиатура расшифровывается так.

- Конфиденциальность (confidentiality)
- Целостность (integrity)
- Доступность (availability)

Конфиденциальность означает секретность данных. Доступ к информации должен быть ограничен кругом лиц, которые имеют право ее знать. Аутентификация, контроль доступа и шифрование — это всего лишь несколько компонентов механизма обеспечения конфиденциальности. Если хакер взломает систему и украдет базу данных, содержащую персональную информацию о клиентах, то конфиденциальность будет нарушена.

Целостность связана с аутентификацией информации. Технология обеспечения целостности данных обеспечивает их корректность и защиту от несанкционированных изменений. Она также связана с благонадежностью источников информации. Когда безопасный веб-сайт предоставляет подписанный SSL-сертификат, он подтверждает пользователю не только то, что информация посылается в зашифрованном виде, но и то, что его идентичность удостоверяется авторитетным источником сертификатов (таким как компания VeriSign или агентство Equifax). Целостность данных также гарантируют специальные технологии, например PGP и Kerberos.

Доступность выражает идею о том, что информация должна быть доступной для авторизованных пользователей, когда они в ней нуждаются, но не хотят ее хранить. Перебои, вызванные не действиями злоумышленников, а административными ошибками или сбоями электропитания, также относятся к этой категории проблем. К сожалению, доступность часто игнорируется, пока не возникнут проблемы.

Принципы CIA следует учитывать при разработке, реализации и эксплуатации систем. Старая поговорка гласит: "Безопасность — это процесс".

## Сертификации

Описание принципов CIA, изложенное выше, представляет собой всего лишь краткое введение в более широкую тему, связанную с безопасностью. Крупные корпорации часто нанимают постоянных сотрудников, работа которых заключается в защите информации. Для того чтобы поддерживать высокую степень компетентности в этой области, эти профессионалы посещают тренинги и получают сертификаты. Если вам придется работать с некоторыми из этих сертификатов, приготовьтесь запомнить массу аббревиатур.

Одним из наиболее широко признанных сертификатов является CISSP (Certified Information Systems Security Professional — сертифицированный специалист по безопасности информационных систем). Этот сертификат выдается организацией $(ISC)^2$, полное название которой звучит так: International Information Systems Security Certification Consortium (попробуйте произнести *это* в десять раз быстрее!). Одной из главных особенностей сертификата CISSP является представление организации $(ISC)^2$ об "общей совокупности знаний" ("common body of knowledge — CBK), которая по существу представляет собой самые эффективные методы работы в области информационной безопасности. Комплекс CBK охватывает законодательство, криптографию, аутентификацию, физическую безопасность и многое другое. Это невероятно авторитетный сертификат в среде специалистов по безопасности.

Недостатком сертификата CISSP является излишняя широта охвата и, как следствие, недостаток глубины знаний. В комплекс CBK входит много тем, которые необходимо изучить за короткое время! Для того чтобы решить эту проблему, организация $(ISC)^2$ создала специальные программы CISSP, посвященные архитектуре, технике и управле-

нию. Эти специализированные сертификаты придают глубину более общему сертификату CISSP.

Институт системного администрирования, сетей и безопасности (System Administration, Networking, and Security Institute — SANS Institute) в 1999 году создал набор сертификатов Global Information Assurance Certification (GIAC). Три дюжины разных экзаменов охватывают всю область информационной безопасности тестами, разделенными на пять категорий. Сертификаты ранжируются по сложности — от умеренного уровня GISF с двумя экзаменами до экспертного уровня GSE с 23-часовым экзаменом. Экзамены на получение сертификата GSE печально известны как самые трудные в данной индустрии. Многие экзамены нацелены на техническое вопросы и требуют довольно большого опыта работы.

В заключение упомянем мандат сертифицированного аудитора информационных систем (Certified Information Systems Auditor — CISA), представляющий собой сертификат специалиста по аудиту и процессам, связанным с информационной безопасностью. Курс обучения для получения этого сертификата сосредоточен на бизнес-процессах, процедурах, мониторинге и других вопросах управления. Некоторые специалисты считают сертификат CISA промежуточным и приемлемым для сотрудников, занимающихся безопасностью в организациях. Одним из преимуществ этого сертификата является то, что для его получения достаточно сдать только один экзамен.

Несмотря на то что сертификаты выдаются индивидуально, их признание в деловой среде невозможно отрицать. В настоящее время все больше компаний считают сертификат признаком эксперта. Многие компании предлагают таким сотрудникам более высокую зарплату и продвигают их по карьерной лестнице.

Если вы решили получить сертификат, убедите вашу организацию, чтобы она оплатила ваше обучение.

## Стандарты безопасности

Возрастающая роль информационных систем привела к появлению законов и указов правительства, регулирующих вопросы защиты конфиденциальной информации.

Многие законодательные акты США, такие как HIPAA, FISMA, NERC CIP, и Sarbanes-Oxley Act (SOX), содержат разделы, посвященные информационной безопасности. Несмотря на то что иногда удовлетворение этих требований связано с большими затратами, они привлекли внимание к важным аспектам технологии, которые ранее игнорировались.

🕮 Более широкое обсуждение промышленных и законодательных стандартов, влияющих на информационные технологии, приведено в разделе 32.9.

К сожалению, эти законодательные акты содержат множество юридических терминов, которые иногда трудно понять. Большинство из них не содержит никаких указаний на то, как удовлетворить сформулированные требования. По этой причине, для того чтобы помочь администраторам выполнить требования закона, были разработаны стандарты. Эти стандарты не являются законодательными актами, но следование этим стандартам обычно обеспечивает согласование с законами. Поскольку сразу все стандарты выполнить бывает затруднительно, лучше всего сначала полностью реализовать какой-то один из них.

### ISO 27002

Стандарт ISO/IEC 27002 (бывший ISO 17799) является, вероятно, самым широко признанным в мире. Впервые он был введен в 1995 году как британский стандарт. Тогда он содержал 34 страницы и 11 разделов, которые охватывали широкий круг вопросов — от политических до вопросов физической безопасности и управления доступом. Каждый раздел имел цели, специфичные требования и рекомендации оптимальных практических решений. Этот документ стоит около 200 долл.

Требования не носят технический характер и могут быть выполнены любой организацией самым эффективным способом. С другой стороны, использование общих слов оставляло у читателя ощущение слишком большой гибкости. Критики жаловались на то, что недостаток специфики оставляет организации незащищенными перед атаками.

Тем не менее, этот стандарт является одним из наиболее ценных документов в области информационной безопасности. Он создал мост над пропастью, пролегавшей между вопросами управления и техническими проблемами, и помог обеим сторонам минимизировать риск.

### PCI DSS

Стандарт Payment Card Industry Data Security Standard (PCI DSS) имеет совершенно другую природу. Он возник вследствие возрастающей потребности повысить безопасность обработки кредитных карточек после ряда драматических событий. Например, в июне 2005 года компания CardSystems Services International обнаружила "пропажу" номеров 40 миллионов кредитных карточек.

По оценкам Министерства национальной безопасности США (U.S. Department of Homeland Security) только в 2009 году из-за кражи персональной информации были потеряны 49,3 млрд долл. Разумеется, не все эти потери могут быть напрямую связаны с потерей номеров кредитных карточек, но возрастающая бдительность разработчиков имела благоприятные последствия. ФБР даже связало поиск мошенников, ворующих номера кредитных карточек, с выявлением террористических групп. Этому способствовали конкретные инциденты, происшедшие в Бали и мадридском метро.

Стандарт PCI DSS является результатом совместных усилий компаний Visa и MasterCard, хотя в данный момент он поддерживается только компанией Visa. В отличие от стандарта ISO 27002, он находится в свободном доступе. Этот стандарт полностью сфокусирован на защите данных о владельцах кредитных карточек и имеет 13 разделов, определяющих требования к этой защите.

Поскольку стандарт PCI DSS посвящен обработке карточек, он не является универсальным и не подходит для бизнеса, не имеющего дела с кредитными карточками. Однако компании, собирающие данные о кредитных карточках, обязаны строго следовать этому стандарту, чтобы избежать крупных штрафов и возможного уголовного преследования.

Этот документ находится на сайте `pcisecuritystandards.org`.

### Документы NIST 800

Сотрудники организации National Institute of Standards and Technology (NIST) разработали ряд документов под названием Special Publication (SP) 800, в которых изложили результаты своих исследований, рекомендации и другую информацию, посвященную компьютерной безопасности. Эти документы часто используются для оценки соответствия организаций, имеющих дело с правительственной информацией, требованиям

федерального закона США об управлении информационной безопасностью (Federal Information Security Management Act). Это открытые и доступные стандарты. Они имеют превосходное содержание и широко применяются в промышленности.

Серия SP 800 содержит более 100 документов. Все они доступны на сайте `csrc.nist.gov/publications/PubsSPs.html`. Перечислим те из них, с которых целесообразно начинать изучение стандартов, — NIST 800-12, *An Introduction to Computer Security: The NIST Handbook*; NIST 800-14, *Generally Accepted Principles and Practices for Securing Information Technology Systems*; NIST 800-34 R1, *Contingency Planning Guide for Information Technology Systems*; NIST 800-39, *Managing Risk from Information Systems: An Organizational Perspective*; NIST 800-53 R3, *Recommended Security Controls for Federal Information Systems and Organizations*; NIST 800-123, *Guide to General Server Security*.

### Стандарт Common Criteria

Документ Common Criteria for Information Technology Security Evaluation (известный под названием "Common Criteria") — это стандарт, по которому оценивают уровень безопасности продукции информационных технологий. Эти рекомендации были установлены международным комитетом, состоящим из представителей разных компаний и отраслей промышленности.

Для ознакомления с этим документом и сертифицированными продуктами посетите сайт `commoncriteriaportal.org`.

### OWASP

Open Web Application Security Project (OWASP) — это некоммерческая всемирная организация, занимающаяся повышением безопасности прикладного программного обеспечения.

Она хорошо известна благодаря своему списку "top 10", посвященному рискам веб-приложений и предназначенному для повышения бдительности разработчиков. Текущую версию этого списка и массу других материалов можно найти на сайте `owasp.org`.

## 22.16. Источники информации по вопросам обеспечения безопасности

В битве за безопасность системы половина успеха — знание современных разработок в этой области. Если система оказалась взломанной, это вряд ли стало следствием применения технологической новинки. Скорее всего, маленькая брешь в броне системы широко обсуждалась в какой-нибудь телеконференции или группе новостей.

### Организация CERT

В связи с бурной реакцией общественности на вторжение интернет-"червя", созданного Робертом Моррисом-младшим (Robert Morris, Jr.) в 1988 году, организация DARPA (Defense Advanced Research Projects Agency — Управление перспективных исследовательских программ Министерства обороны США) учредила новую структуру: CERT (Computer Emergency Response Team — группа компьютерной "скорой помощи"). В ее обязанности входит сбор информации, касающейся компьютерной безопасности. До сих пор это наиболее известный консультативный орган, хотя со временем он стал весьма медлительным и бюрократичным. Более того, в самой организации сейчас настаива-

ют на том, что ее название не означает ничего кроме "зарегистрированной служебной марки университета Карнеги-Меллона".

В средине 2003 года организация CERT заключила договор о партнерстве с Управлением компьютерной безопасности Министерства национальной безопасности США (Department of Homeland Security's National Cyber Security Division, NCSD). Непонятно, к лучшему или к худшему, но новая организация изменила структуру списков рассылки.

Объединенная организация под названием US-CERT предлагает четыре списка рассылки. Самым полезным среди них является "Technical Cyber Security Alerts". Подписаться на любой из этих четырех списков рассылки можно на сайте `forms.us-cert.gov/maillists`.

## Сервер SecurityFocus.com и список рассылки BugTraq

Сервер `SecurityFocus.com` специализируется на сборе новостей и различной информации по вопросам безопасности. В новостях публикуются как общие обзоры, так и статьи, посвященные конкретным проблемам. Есть также обширная библиотека полезных документов, удобно отсортированных по тематике.

Программный архив сервера содержит инструментальные средства для множества операционных систем. Программы снабжаются аннотациями и пользовательскими рейтингами. Это наиболее глобальный и исчерпывающий из известных нам архивов подобного рода.

Список рассылки BugTraq — это форум для обсуждения проблем безопасности и путей их устранения. Для того чтобы подписаться на него, посетите страницу `securityfocus.com/archive`. Объем информации, рассылаемый по списку, может оказаться довольно значительным, и отношение "сигнал-шум" довольно плохое. На веб-узле также хранится база данных с отчетами о рассмотренных в форуме BugTraq проблемах.

## Блог Брюса Шнайера

Блог Брюса Шнайера (Bruce Schneier), посвященный вопросам безопасности, является ценным и иногда забавным источником информации о компьютерной безопасности и криптографии. Кроме того, Шнайер является автором широко известных книг *Applied Cryptography* и *Secrets and Lies*. Информацию с его блога можно получать в виде ежемесячного бюллетеня под названием Grypto-Gram. Более подробную информацию вы найдете на сайте `schneier.com/crypro-gram.html`.

## Организация SANS

SANS (System Administration, Networking, and Security Institute — институт системного и сетевого администрирования и проблем безопасности) — это профессиональная организация, которая спонсирует конференции и обучающие семинары, посвященные вопросам безопасности, а также публикует различную информацию по данной тематике. Ее сайт `sans.org` является ценным информационным ресурсом, занимающим промежуточное положение между сайтами SecurityFocus и CERT; он не столь исчерпывающий, как первый, но и не такой скудный, как второй.

Организация SANS рассылает по электронной почте ряд еженедельных и ежемесячных бюллетеней, на которые можно подписаться на ее веб-узле.

## Информационные ресурсы отдельных дистрибутивов

Проблемы, касающиеся безопасности, очень влияют на репутацию, поэтому поставщики операционных систем остро заинтересованы в том, чтобы помочь пользователям сделать свои системы безопасными. Многие крупные поставщики ведут собственные списки рассылки и публикуют в них бюллетени по вопросам безопасности. Эта же информация часто доступна и на веб-узлах. Не редкость, когда защитные "заплаты" распространяются бесплатно, даже если поставщики обычно взимают плату за программную поддержку.

В Интернете есть веб-порталы, например `SecurityFocus.com`, на которых содержится информация, касающаяся конкретных производителей, и имеются ссылки на последние официальные пресс-релизы.

Разработчики системы Ubuntu поддерживают список рассылки, посвященный вопросам безопасности, по адресу:
`https://lists.ubintu.com/mailman/listinfo/ubuntu-security-announce`

Рекомендации по поводу защиты SuSE можно найти по адресу:
`novell.com/linux/security/securitysupport.html`

Для того чтобы подписаться на официальный список рассылки, посвященный вопросам безопасности SuSE, посетите страницу
`suse.com/en/private/support/online_help/mailinglists/index.html`

Объявления о продуктах компании Red Hat, связанных с безопасностью, рассылаются по списку "Enterprise watch", который расположен по адресу: `redhat.com/mailman/listinfo/enterprise-watch-list`.

Несмотря на то что компания Sun Microsystems была приобретена компанией Oracle, ее блог, посвященный вопросам безопасности, продолжает обновляться (`blogs.sun.com/security/category/alerts`). Когда торговая марка и местоположение блога изменятся, вероятно, вы найдете там необходимую ссылку.

Доступ к предложениям компании HP можно получить с помощью сайтов `us-support.external.hp.com` (для Америки и Азии) и `europe-support.external.hp.com` (для Европы).

Информация, связанная с обеспечением безопасности, тщательно скрыта. Для того чтобы найти ее, войдите в область `maintenance/support` и выберите пункт поиска по базе технических знаний. Ссылка, находящаяся среди опций фильтра на этой странице, приведет вас к архиву бюллетеней по вопросам безопасности. (Если вы не зарегистрированы, придется это сделать.) В этой области можно также найти "заплаты" для систем безопасности.

Для того чтобы получать бюллетени по вопросам безопасности, вернитесь на главную страницу в область `maintenance/support` и выберите пункт "Subscribe to proactive notifications and security bulletins". К сожалению, по электронной почте подписаться не получится.

Подписаться на уведомления по вопросам безопасности системы AIX можно с помощью ссылки "My notifications" на странице `ibm.com/systems/support`.

Информация о безопасности продуктов Cisco распространяется в виде практических замечаний, список которых находится по адресу `cisco.com/public/support/tac/fn_index.html`. Для того чтобы подписаться на список рассылки, посвященный вопросам безопасности продуктов Cisco, направьте письмо по адресу `majordomo@cisco.com`, включив в тело сообщения строку "subscribe cust-security-announce".

## Другие списки рассылки и веб-сайты

Перечисленные выше контактные адреса — лишь небольшая часть интернет-ресурсов по вопросам безопасности. Учитывая объем имеющейся сегодня информации, а также скорость, с которой появляются и исчезают различные ресурсы, мы посчитали целесообразным указать несколько "метаресурсов".

Хорошей стартовой площадкой является веб-узел `linuxsecurity.com`, на котором каждый день размещается несколько сообщений по вопросам безопасности системы Linux. Он также содержит постоянно обновляющуюся коллекцию советов по вопросам безопасности системы Linux, регистрирует происходящие события и поддерживает работу групп пользователей.

(IN)SECURE — это свободно распространяемый двумесячный журнал, содержащий информацию о современных тенденциях, а также интервью с выдающимися специалистами в области безопасности. Правда, некоторые статьи в этом журнале следует читать с долей недоверия — убедитесь, что ее автор не занимается беззастенчивой рекламой своей продукции.

Сайт Linux Security (`linuxsecurity.com`) освещает последние новости, касающиеся системы Linux и вопросов безопасности открытых кодов. Для того чтобы извлечь из него максимальную выгоду, подпишитесь на канал RSS.

Linux Weekle News — это симпатичный источник информации о ядре, безопасности, дистрибутивных пакетах и других вопросах. Раздел, посвященный проблемам безопасности, находится по адресу `lwn.net/security`.

# 22.17. Что нужно делать в случае атаки на сервер

Прежде всего, не паникуйте! Скорее всего, к моменту обнаружения взлома большая часть ущерба уже нанесена. Возможно, хакер "гулял" по системе несколько недель или даже месяцев. Вероятность того, что вам удалось выявить факт взлома, происшедшего всего час назад, близка к нулю.

В свете этого мудрая сова приказывает сделать глубокий вдох и начать тщательно продумывать стратегию устранения взлома. Старайтесь не вспугнуть злоумышленника, во всеуслышание заявляя о происшествии или выполняя действия, которые могут насторожить того, кто наблюдал за деятельностью организации на протяжении нескольких недель. Подсказка: в этот момент неплохо выполнить резервное копирование. Надеемся, это не удивит нарушителя?![13]

Стоит также напомнить самому себе о том, что, согласно исследованиям, в 60% инцидентов, связанных с нарушением безопасности, присутствовал "внутренний враг". Не изучив всех фактов, старайтесь не посвящать в проблему лишних людей, кем бы они ни были.

Приведем короткий план, который поможет администратору пережить кризис.

[13] Если создание резервных копий не является "нормальным" действием в вашей организации, то обнаружение взлома окажется самой незначительной из ожидающих вас проблем.

**Этап 1: не паникуйте.** Во многих случаях проблема обнаруживается только спустя какое-то время. Еще один-два часа или дня уже ничего не решают. Однако имеет значение реакция на событие — паническая или рациональная. Часто в результате возникшей паники ситуация только усугубляется уничтожением важной информации, которая позволяет выявить нарушителя.

**Этап 2: определите адекватную реакцию.** Никому не выгодно раздувать инцидент. Будьте благоразумны. Решите, кто из персонала будет участвовать в решении возникшей проблемы и какие ресурсы при этом должны быть задействованы. Остальные будут помогать осуществлять "вынос тела", когда все закончится.

**Этап 3: соберите всю возможную информацию.** Проверьте учетные и журнальные файлы. Попытайтесь определить, где первоначально произошел взлом. Выполните резервное копирование всех систем. Убедитесь в том, что резервные ленты физически защищены от записи, если вы вставляете их в дисковод для чтения.

**Этап 4: оцените нанесенный ущерб.** Определите, какая важная информация (если таковая имеется) "покинула" компанию, и выработайте подходящую стратегию смягчения возможных последствий. Оцените степень будущего риска.

**Этап 5: выдерните шнур.** Если это необходимо и возможно, отключите "взломанные" компьютеры от сети. Перекройте обнаруженные "дыры". Организация CERT предлагает инструкции по обнаружению взлома. Этот документ находится по адресу:

```
cert.org/tech_tips/win-UNIX-system_compromise.html
```

**Этап 6: разработайте стратегию восстановления.** Соберите толковых людей и обсудите план восстановления. Не заносите его в компьютер. Сосредоточьтесь на том, чтобы потушить "пожар" и свести ущерб к минимуму. Избегайте обвинений в чей-либо адрес или нагнетания обстановки. Не забывайте о психологическом ударе, который могут испытать пользователи.

**Этап 7: сообщите о своем плане.** Расскажите пользователям и управляющему персоналу о последствиях взлома, возможных будущих проблемах и предварительной стратегии восстановления. Будьте честны и откровенны. Инциденты, связанные с безопасностью, являются неотъемлемой частью современных сетей. Это не отражение ваших способностей как системного администратора или чего-то другого, чего можно было бы стыдиться. Открытое признание возникшей проблемы — 90% победы, если при этом вы демонстрируете, что у вас есть план выхода из ситуации.

**Этап 8: воплотите план в жизнь.** Вы знаете свои системы и сети лучше, чем кто бы то ни было. Доверьтесь плану и инстинктам. Посоветуйтесь с коллегами из других организаций, чтобы убедиться в правильности выбранного направления.

**Этап 9: сообщите об инциденте в соответствующие органы.** Если в инциденте участвовал "внешний игрок", сообщите об этом случае в организацию CERT. Соответствующий адрес электронной почты — `cert@cert.org`. Предоставьте им как можно больше информации.

Стандартная форма для отчета находится на веб-узле `cert.org`. Ниже приведен список того, что желательно включить в отчет.

- имена, модели "взломанных" компьютеров, а также установленные на них операционные системы;
- список "заплат", которые были установлены в системе на момент инцидента;
- список учетных записей, подвергшихся нападению;
- имена и IP-адреса всех удаленных компьютеров, вовлеченных в инцидент;

- контактная информация, если известны администраторы удаленных систем;
- соответствующие журнальные записи и учетные данные.

Если есть подозрение, что вами обнаружена ранее неизвестная проблема в программном обеспечении, следует также сообщить об этом компании-разработчику.

## 22.18. Рекомендуемая литература


- Barrett, Daniel J., Siverman Richard E., Byrnes Robert G. "Linux Security Cookbook". Sebastobol, CA: O'Reilly Media, 2003.
- Bauer Michael D., "Linux Server Security (2nd Edition). Sebastobol, CA: O'Reilly Media, 2005.
- Bryant, William. "Designing an Authentication System: a Dialogue in Four Scenes." 1988; `web.mit.edu/kerberos/www/dialogue.html`.
- Cheswick William R., Bellovin Steven M., Rubin Aviel D. "Firewalls and Internet Security: Repelling the Wily Hacker (2nd Edition). Reading, MA: Addison-Wiley, 2003.
- Curtin Matt, Ranum Marcus, Robinson P.D. "Internet Firewalls: Frequently Asked Quetions", 2004; `interhack.net/pubs/fwfaq`.
- Farrow Rick, Power Richard "Network defence article series", 1998–2004; `spirit.com/Network`.
- Fraser B. (Ed.) "RFC2196: Site Security Handbook." 1997; `rfc-editor.org`.
- Garfinkel Simson, Spafford Gene, Schwartz Alan. "Practical UNIX and Internet Security (3rd Edition). Sebastobol, CA: O'Reilly Media, 2003.
- Kerby Fred et. al. SANS Intrusion Detection and Response FAQ". SANS. 2009; `sans.org/resources/idfaq`.
- Lyon Gordon Fyodor. "Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning". Nmap Project, 2009.
- Mann Scott, Mitchell Ellen I. "Linux System Security: The Administrator's Guide to Open Source Security Tools (2nd Edition). Upper Saddle River, NJ: Prentice Hall PTR, 2002.
- Morris Robert, Thompson Ken. "Password Security: A Case History". *Communications pf the ACM*, 22 (11): 594–597, November, 1979. Reprinted in "*Unix System Manager's Manual*", 4.3 Berkeley Software Distribution. Universityof California, Berkeley, April 1986.
- Ritchie, Dennis M. "On the Security of UNIX." May 1975. Перепечатано в *UNIX System Manager's Manual,* 4.3 Berkeley Software Distribution. University of California, Berkeley. April 1986.
- Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* New York, NY: Wiley, 1995.
- Steves K. "Building a Bastion Host Using HP-UX 11". HP Consulting, 2002; `tinyurl.com/5sffy2`.
- Thompson, Ken. "Reflections on Trusting Trust." Опубликовано в *ACM Turing Award Lectures: The First Twenty Years 1966-1985.* Reading, MA: ACM Press (Addison-Wesley), 1987.

# 22.19. Упражнения

22.1. Каковы преимущества SSH-аутентификации с использованием паролей Linux по сравнению с SSH-аутентификацией посредством идентификационного пароля и пары ключей. Если один из методов обеспечивает большую безопасность, обязательно ли переходить на него?

22.2. ★ Туннели SSH часто являются единственным средством передачи трафика на удаленный компьютер, к которому у вас нет административного доступа. Прочтите **man**-страницу команды **ssh** и составьте команду, которая организует туннельную пересылку трафика с порта 113 локального компьютера на порт 113 узла `mail.remotenetwork.org`. Точкой перенаправления туннеля тоже должен быть узел `mail.remotenetwork.org`.

22.3. ★ Проанализируйте один из последних преданных огласке инцидентов, связанных с компьютерной безопасностью. Найдите соответствующие "заплаты" или способы защиты компьютеров вашей лаборатории. Укажите, какими источниками информации вы пользовались и какие действия следует предпринять для защиты системы.

22.4. ★★ Предварительно получив разрешение от администратора, инсталлируйте утилиту John the Ripper, которая ищет учетные записи со "слабыми" паролями.

а) модифицируйте исходный код этой утилиты, чтобы она отображала только регистрационные имена, но не пароли.

б) примените утилиту John the Ripper к файлу паролей вашей лаборатории (потребуется получить доступ к файлу **/etc/shadow**) и посмотрите, сколько паролей она сможет взломать.

в) задайте собственный пароль в виде слова из словаря и заставьте утилиту **john** проверить только вашу запись файла **/etc/shadow**. Сколько времени ушло на взлом пароля?

г) попробуйте разные варианты написания (с заглавной буквы, с цифрой в конце, однобуквенный пароль и так далее) и оцените возможности утилиты.

22.5. ★★ Выберите два компьютера в лаборатории и назначьте один из них мишенью, а второй — зондом.

22.6. Инсталлируйте на зондирующем компьютере утилиты **nmap** и Nessus и попытайтесь атаковать компьютер-мишень. Как можно обнаружить факт атаки?

22.7. Создайте на компьютере-мишени брандмауэр с помощью команды **iptables** для защиты от этих атак. Можно ли теперь обнаружить факт атаки? Как именно?

22.8. Какими дополнительными средствами защиты можно воспользоваться? (Необходим доступ с правами суперпользователя.)

22.9. ★★ Используя общие списки рассылки или веб-сайт, идентифицируйте приложение, которое недавно обнаружило уязвимость. Найдите надежный источник информации о "лазейке" и обсудите пути ее устранения.

22.10. ★★ Программы с установленным битом SETUID часто являются необходимыми. А вот SETUID-сценариев интерпретатора команд следует избегать. Почему?

22.11. ★★ Используя команду **tcpdump**, перехватите трафик службы FTP в активном и пассивном режимах. Как необходимость поддерживать анонимный сервер FTP

влияет на функционирование брандмауэра? Что именно должны разрешить правила брандмауэра? (Необходим доступ с правами суперпользователя.)

22.12. ★★ Что разрешают и запрещают показанные ниже правила, сообщаемые командой **iptables**? Что можно сделать для повышения безопасности такой системы? (Подсказка: цепочки OUTPUT и FORWARD можно расширить.)

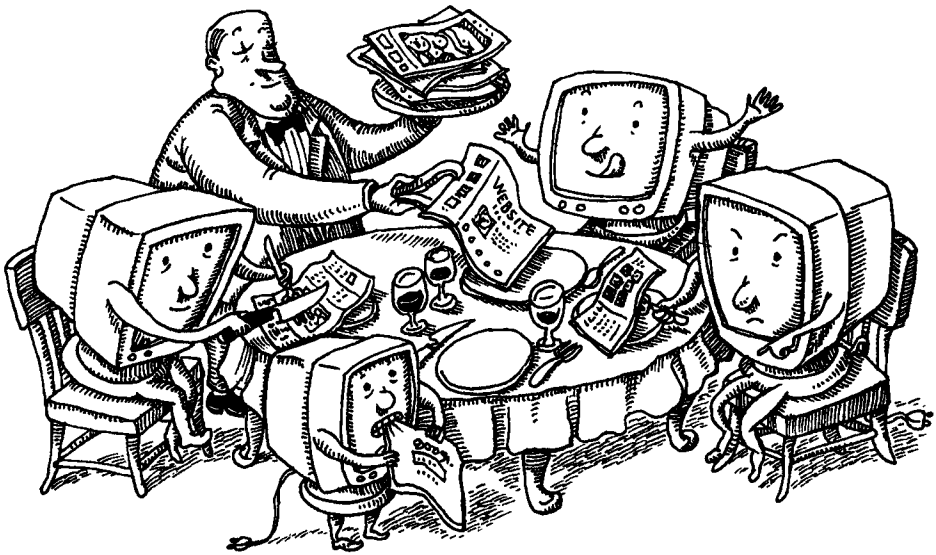
```
Chain INPUT (policy ACCEPT)
target  prot  opt  source    destination
block   all   --   anywhere  anywhere

Chain FORWARD (policy ACCEPT)
target  prot  opt  source    destination
        all   --   anywhere  anywhere

Chain OUTPUT (policy ACCEPT)
target  prot  opt  source    destination

Chain block (1 references)
target  prot  opt  source         destination
ACCEPT  all   --   anywhere       anywhere    state RELATED,
ESTABLISHED
ACCEPT  tcp   --   anywhere       anywhere    state NEW tcp dpt:www
ACCEPT  tcp   --   anywhere       anywhere    state NEW tcp dpt:ssh
ACCEPT  tcp   --   128.138.0.0/16 anywhere    state NEW tcp dpt:
kerberos
ACCEPT  icmp  --   anywhere       anywhere
DROP    all   --   anywhere       anywhere
```

22.13. ★★ Проанализируйте цепочки правил вашего брандмауэра. Какие стратегии защиты они выражают? Все ли надежно в такой системе? (Выполнение этого упражнения может потребовать взаимодействия с администраторами, отвечающими за безопасность организации.)

22.14. ★★★★★ Напишите программу, которая находит сетевые интерфейсы, работающие в "беспорядочном" режиме. Регулярно запускайте эту программу для раннего обнаружения попыток вторжения в сеть. Какую нагрузку на сеть создает программа? Ее нужно запускать на каждом компьютере или достаточно одного? Можете ли вы составить такой пакет, который сообщит о том, что интерфейс находился в "беспорядочном" режиме? (Необходим доступ с правами суперпользователя.)

# Глава 23

# Веб-хостинг




В настоящее время системы UNIX и Linux являются основными платформами для обслуживания веб-содержимого и веб-приложений. Это идеальные системы для этой цели, поскольку они изначально были разработаны как многозадачные системы с вытеснением (preemptive, multitasking systems). Они могут обрабатывать огромный объем веб-запросов и делать это эффективно, безопасно и надежно.

В некоторых аспектах веб-приложения действительно облегчили работу администраторов. Технологии "Web 2.0", такие как AJAX (Asynchronous JavaScript and XML) и динамический HTML, не только предоставили пользователям функциональные возможности локально инсталлированных приложений, но и освободили системных администраторов от проблем, связанных с развертыванием программного обеспечения; из всего программного обеспечения на клиентской стороне требуется лишь веб-браузер.

На серверной стороне типичной конфигурацией является высоко функциональный стек LAMP (Linux, Apache, MySQL и PHP/Perl/Python)[1]. Для приложений, связанных с управлением базами данных, популярные веб-приложения с открытым кодом основаны на языке Ruby. Оба стека являются разумным выбором и легко поддерживаются.

## 23.1. Основы веб-хостинга

Хостинг веб-узла мало чем отличается от других сетевых услуг. Демон прослушивает TCP-порт с номером 80 (в соответствии со стандартом HTTP), принимает запросы на выдачу документов, а затем посылает эти документы пользователю. Многие из этих

[1] Дистрибутивные пакеты, не содержащие операционную систему Linux, называют эту конфигурацию просто AMP. В системе Solaris она называется SAMP, а в среде пользователей системы Windows — WAMP. Поди разбери.

документов генерируются "на лету" во взаимодействии с базами данных и прикладными каркасами, но для основного протокола HTTP это не характерно.

## Обнаружение ресурсов в сети веб

Информация в Интернете организована с помощью архитектуры, определенной организацией Internet Society (ISOC). Это целеустремленная (хотя и управляемая комитетами) организация, которая поддерживает согласованную работу и операционную совместимость в Интернете.

🕮 Информацию об организации Internet Society можно найти в разделе 14.1.

Организация ISOC определяет три способа идентификации ресурсов: унифицированные идентификаторы ресурсов (Uniform Resource Identifier — URI), унифицированные указатели ресурсов (Uniform Resource Locator — URN) и унифицированные имена ресурсов (Uniform Resource Names — URN). Как показано на рис. 23.1, по существу, унифицированные указатели и имена ресурсов представляют собой частный случай унифицированного идентификатора ресурсов.

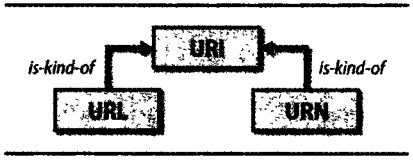



*Рис. 23.1. Классификация унифицированных идентификаторов ресурсов*

В чем же различия между ними?

- Унифицированные указатели ресурсов сообщают вам, как найти ресурс, описывая его механизм первичного доступа (например, `http://admin.com`).
- Унифицированные имена ресурсов идентифицируют ("называют") ресурс без указания на него или сообщают, как получить доступ к нему (например, `urn:isbn:0-13-020601-6`).

Что же тогда называется URI? Если доступ к ресурсу возможен только через Интернет, то используется указатель URL. Если же доступ к ресурсу возможен не только через Интернет, но и с помощью других средств, то используется идентификатор URI.

## Унифицированные указатели ресурсов

Большую часть времени мы работаем с указателями URL, описывающими доступ к объекту с помощью пяти базовых компонентов.

- Протокол или приложение
- Имя узла
- Порт TCP/IP (необязательно)
- Каталог (необязательно)
- Имя файла (необязательно)

В табл. 23.1 перечислены протоколы, которые наиболее часто упоминаются в URL-адресах.

**Таблица 23.1. Протоколы URL-адресов**

| Протокол | Назначение | Пример |
|---|---|---|
| file | Доступ к локальному файлу (выход в Интернет не осуществляется) | file://etc/syslog.conf |
| ftp | Доступ к файлам по протоколу FTP | ftp://ftp.admin.com/adduser.tar.gz |
| http | Доступ к файлам по протоколу HTTP | http://admin.com/index.html |
| https | Доступ к файлам по протоколам HTTP/SSL | https://admin.com/order.shtml |
| ldap | Доступ к службе каталогов LDAP | ldap://ldap.bigfoot.com:389/cn=Herb |
| mailto | Отправка электронного сообщения по указанному адресу | mailto:sa-book@admin.com |

## Принцип работы HTTP

HTTP — это клиент-серверный протокол, не хранящий информацию о состоянии сеанса. Клиент запрашивает у сервера "содержимое" заданного URL-адреса. Сервер отвечает, передавая поток данных или возвращая сообщение об ошибке. Затем клиент может запросить следующий объект.

Поскольку протокол HTTP настолько прост, можно легко подключиться к веб-браузеру с помощью утилиты **telnet**. Для этого достаточно подключиться к порту 80 выбранного веб-сервера. После установления соединения сервер готов принимать HTTP-команды.

Чаще всего используется команда **GET**, которая запрашивает содержимое документа. Обычно она задается в формате **GET /**. В этом случае возвращается корневой документ сервера (как правило, начальная страница). Протокол HTTP чувствителен к регистру символов, поэтому команды следует набирать прописными буквами.

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.atrust.com.
Escape character is '^]'.
GET /
<далее следует содержимое файла, заданного по умолчанию>
Connection closed by foreign host.
```

Более "полный" HTTP-запрос должен включать номер версии протокола HTTP, запрашиваемый узел (необходимый для того, чтобы извлечь файл из виртуального узла, использующего имена) и другую информацию. В этом случае ответ, помимо возвращаемых данных, содержит информативные заголовки.

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.atrust.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.atrust.com

HTTP/1.1 200 OK
Date: Sat, 01 Aug 2009 17:43:10 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Sat, 01 Aug 2009 16:10:22 GMT
Content-Length: 7044
Content-Type: text/html
```

```
<далее следует содержимое файла, заданного по умолчанию>
Connection closed by foreign host.
```

В данном случае мы сообщили серверу, что собираемся передавать данные по протоколу HTT версии 1.1, и назвали виртуальный узел, у которого мы запрашиваем информацию. Сервер вернул нам код состояния (HTTP/1.1 200 OK), текущую дату и время, имя и версию программного обеспечения, на котором выполняется протокол, дату последнего изменения запрашиваемого файла и содержимое запрашиваемого файла. Информация заголовка отделена от содержимого одной пустой строкой.

## Генерирование содержимого "на лету"

Помимо работы со статическими документами, HTTP-сервер способен выдавать пользователям страницы, формируемые "на лету". Например, если требуется отобразить текущие время и температуру, сервер вызывает сценарий, предоставляющий эту информацию. Такие сценарии зачастую создаются средствами CGI (Common Gateway Interface — единый шлюзовой интерфейс).

CGI является не языком программирования, а, скорее, спецификацией, описывающей обмен информацией между HTTP-сервером и другими программами. Чаще всего CGI-сценарии представляют собой программы, написанные на языке Perl, Python или PHP. В действительности подойдет практически любой язык программирования, который поддерживает операции ввода-вывода в режиме реального времени.

### *Встроенные интерпретаторы*

Модель CGI обеспечивает высокую гибкость, благодаря которой разработчик веб-приложения может свободно использовать любой интерпретатор или язык сценариев. К сожалению, запуск отдельного процесса в каждом сценарии может стать ночным кошмаром для загруженного веб-сервера, обрабатывающего значительный объем динамического содержания.

Кроме поддержки дополнительных внешних сценариев CGI, многие веб-серверы определяют модульную архитектуру, позволяющую интерпретаторам сценариев, таким как Perl и PHP, самим встраиваться в веб-сервер. Это связывание значительно увеличивает производительность, поскольку веб-сервер больше не обязан запускать отдельный процесс, чтобы обрабатывать каждый запрос сценария. Архитектура, большей частью, скрыта от разработчиков сценариев. Как только сервер видит файл, имя которого заканчивается специфическим расширением (например, `.pl` или `.php`), он посылает содержимое файла встроенному интерпретатору для выполнения. Некоторые интерпретаторы, которые могут выполняться внутри веб-сервера Apache, перечислены в табл. 23.2.

**Таблица 23.2. Модули встроенных сценариев для веб-сервера Apache**

| Язык | Имя встроенного интерпретатора | Источник информации |
|---|---|---|
| Perl | mod_perl | `perl.apache.org` |
| Python | mod_python | `modpython.org` |
| PHP | mod_php (традиционно)<br>Zend server (коммерческий ускоритель) | `apache.org`<br>`zend.com` |
| Ruby on Rails | Phusion Passenger (он же mod_rails или mod_rack) | `modrails.com` |

### Модуль FastCGI

В некоторых ситуациях удобно использовать модуль FastCGI (`fastcgi.com`). Этот модуль улучшает производительность сценариев, запуская их один раз, а затем оставляя выполняться для обслуживания многих запросов. Такая схема снижает стоимость запуска интерпретатора и анализа сценария при выполнении нескольких запросов. Иногда она работает быстрее, чем запуск интерпретатора внутри самого веб-сервера Apache.

К сожалению, в этом случае сценарии приходится модифицировать, чтобы они соответствовали новому способу взаимодействия с веб-сервером. Основной протокол реализовать легко, но сценарии FastCGI не могут избежать погрешностей при управлении памятью. Неудачные завершения сценариев оставляют в памяти следы, количество которых со временем возрастает. Другая потенциальная опасность заключается в персистентности данных между выполнением запросов; программист должен гарантировать, что эти данные не будут взаимодействовать друг с другом. Разработчики веб-приложений должны оценить, стоит ли повышение производительности за счет сценариев FastCGI дополнительных усилий и риска.

### Безопасность сценариев

Как правило, CGI-сценарии интересуют веб-разработчиков и программистов. Однако один из аспектов сценариев не обходит стороной и системных администраторов — безопасность. Поскольку CGI-сценарии получают доступ к файлам, сетевым соединениям и прочим механизмам перемещения данных, выполнение этих сценариев несет потенциальную угрозу компьютеру, на котором инсталлирован HTTP-сервер. Ведь CGI-сценарий или надстройка позволяет любому пользователю выполнять программы на сервере. Поэтому CGI-сценарии должны подчиняться тем же правилам безопасности, что и другие сетевые программы.

Сайт OWASP (Open Web Application Security Project, `owasp.org`) публикет массу превосходных материалов о веб-безопасности. Общую информацию по вопросам безопасности можно найти в главе 22.

## Серверы приложений

Сложные промышленные приложения могут требовать больше функциональных возможностей, чем может предоставить базовый HTT-сервер. Например, современные страницы Web 2.0 часто содержат компоненты, связанные с динамическими котировками (например, биржевым аппаратом). Несмотря на то что на веб-сервере Apache эту функциональную возможность можно реализовать с помощью таких технологий, как AJAX и JavaScript Object Notation (JSON), некоторые разработчики предпочитают более богатые языки, такие как Java. Обычно для взаимодействия приложения, написанного на языке Java, с другими источниками данных на предприятии используется "сервлет".

Сервлеты (servelet) — это программы, которые выполняются на сервере прикладных программ поверх его платформы. Серверы прикладных программ могут работать независимо или в сочетании с веб-сервером Apache. Большинство серверов прикладных программ разрабатывается "программистами для программистов" и не имеет достаточно мощного механизма отладки, необходимого системным администраторам. В табл. 23.3 перечислены некоторые серверы прикладных программ для систем UNIX/Linux.

Если вам придется столкнуться с необходимостью поддержки одного из этих серверов прикладных программ, поищите документацию, описывающую программы, и по-

тренируйтесь. Поверьте нам, это не та технологию, которую можно освоить "на ходу", как большинство приложений для систем UNIX и Linux. Мы вас предупредили.

**Таблица 23.3. Модули встроенных сценариев для веб-сервера Apache**

| Сервер | Тип | Веб-сайт |
|---|---|---|
| Tomcat | Открытый код | tomcat.apache.org |
| GlassFish | Открытый код | grassfish.dev.java.net |
| JBoss | Открытый код | jboss.org |
| OCJ4 | Коммерческий | oracle.com/technology/tech/java/oc4j |
| WebSphere | Коммерческий | ibm.com/websphere |
| WebLogic | Коммерческий | oracle.com/appserver/weblogic/weblogic-suite.html |
| Jetty | Открытый код | eclipse.org/jetty |

## Распределение нагрузки

Довольно трудно предсказать, сколько обращений (запросов к одному объекту, такому как текстовый файл или изображение) либо операций просмотра страниц (запросов всех объектов на одной просматриваемой странице) может обработать сервер за единицу времени. Все зависит от операционной системы, ее настройки, аппаратной платформы, а также от организации веб-узла (например, содержатся ли там статические HTML-страницы или выполняются запросы к базе данных и математические вычисления).

Только непосредственное определение производительности и уточнение фактических параметров узла, использующего реальное аппаратное обеспечение, позволят ответить на этот вопрос. Иногда подсказку могут дать разработчики, создающие аналогичные узлы с помощью аналогичного аппаратного обеспечения. Но ни в коем случае не стоит полагаться на числа, указываемые поставщиками коммерческих систем. Кроме того, следует помнить о том, что одним из основных факторов является пропускная способность. Отдельная машина, обслуживающая статические HTML-файлы и изображения, может легко выдать столько данных, чтобы вызвать перегрузку канала типа OC3 (155 Мбайт/с).

Иначе говоря, вместо показателя производительности одного сервера лучше сосредоточиться на показателе масштабируемости; перед перегрузкой интерфейса Ethernet веб-сервер обычно интенсивно использует центральный процессор или механизм ввода-вывода. Убедитесь в том, что вы и ваша команда веб-дизайнеров работаете по плану, который позволяет распределять нагрузку узла между несколькими серверами.

Распределение нагрузки повышает как производительность работы, так и избыточность. Существует несколько различных подходов к распределению нагрузки: круговая система DNS, аппаратное распределение нагрузки и программное распределение нагрузки.

Круговая система DNS — это простейшая форма распределения нагрузки. В этой системе нескольким IP-адресам назначается одно имя узла. Когда на сервер имен поступает запрос к веб-серверу с заданным IP-адресом, клиент получает в ответ один IP-адрес. Адреса обрабатываются один за другим в круговой последовательности.

🕮 Информация о системе DNS и ее функционировании изложена в главе 17.

Круговая система DNS используется очень широко. Она применяется даже компанией Google. Например, если вы пошлете инфраструктуре DNS запрос о сервере `www.google.com`, то получите в ответ примерно такие записи.

```
$ dig www.google.com a
…
;; QUESTION SECTION:
;www.google.com. IN A
;; ANSWER SECTION:
www.google.com. 0 IN CNAME www.l.google.com.
www.l.google.com. 65 IN A 74.125.95.104
www.l.google.com. 65 IN A 74.125.95.105
www.l.google.com. 65 IN A 74.125.95.106
www.l.google.com. 65 IN A 74.125.95.147
www.l.google.com. 65 IN A 74.125.95.99
www.l.google.com. 65 IN A 74.125.95.103
```

В данном примере имя `www.google.com` отображается в каноническое имя `www.l.google.com`. Компания Google добавляет этот уровень косвенной адресации, чтобы делегировать ответственность за доставку содержимого внутреннему провайдеру, например Akamai (см. раздел 23.6), не предоставляя управления содержанием (CDN) своему корневому домену.

Клиент DNS может выбрать любую из записей A, возвращенных для сайта `www.l.google.com`. Предполагается, что это делается случайным образом. В противоположность общепринятому мнению, порядок, в котором возвращаются записи A, значения не имеет. В частности, первая запись не является "главной". Поскольку клиенты выбирают адрес случайно, нагрузка на этот сайт примерно равномерно распределяется между шестью серверами.

Недостатком круговой системы DNS является то, что в случае отказа сервера данные DNS необходимо обновить, чтобы можно было перенаправить трафик в обход. Длинные тайм-ауты, связанные с записями A, могут сделать эту операцию сложной и ненадежной. С другой стороны, короткие тайм-ауты мешают кешированию и замедляют поиск сайта в системе DNS, затрачивая дополнительные ресурсы. Обсуждение этой проблемы изложено в разделе 17.2.

В приведенном выше примере записи A можно кешировать за 65 секунд до истечения срока их действия. Это относительно короткий тайм-аут. Если в вашем распоряжении есть резервный сервер, можно установить более длинные тайм-ауты для записей A и просто переназначить IP-адрес поврежденного сервера на резервный сервер.

Аппаратное распределение трафика является довольно простой альтернативой круговой системе DNS, но оно требует наличия резервного кеша. К аппаратному обеспечению, предназначенному для распределения нагрузки, относятся Big-IP Controller компании F5 Networks, веб-коммутаторы Alteon компании Nortel и коммутаторы Content Services Swithes компании Cisco. Эти устройства ориентируются на параметры, задаваемые администратором, в частности на время отклика отдельного сервера и время доступности. Благодаря распределению нагрузки увеличивается производительность и отказоустойчивость сети.

Программное распределение нагрузки не требует использования аппаратного обеспечения; эти программы можно запускать в системе UNIX. Существуют как открытые, так и коммерческие программы для распределения нагрузки. К программам с открытым кодом относятся Linux Virtual Server (`linuxvirtualserver.org`) и модуль для распределения нагрузки между прокси-серверами (mod_proxy_balancer), внедренный на веб-сервере Apache 2.2. Примером коммерческого продукта является программа Zeus (`zeus.com`).

Компания Google на самом деле использует сочетание круговой DNS-системы и аппаратного распределения нагрузки. Более подробно этот механизм описан в Википедии в статье "Google platform".

Помните, что в настоящее время большинство сайтов генерируется автоматически. Эта архитектура предусматривает колоссальную нагрузку на серверы баз данных. При необходимости проконсультируйтесь у администратора ваших баз данных, как лучше организовать распределение нагрузки между многими серверами баз данных.

# 23.2. Инсталляция HTTP-сервера

Установить и поддерживать веб-сервер очень просто! Веб-службы значительно уступают системам электронной почты и службе DNS в плане сложности администрирования.

## Выбор сервера

Существует несколько HTTP-серверов, но чаще всего используют сервер Apache, который отличается высокой гибкостью и производительностью. По состоянию на январь 2010 года, 54% веб-серверов в Интернете работали под управлением Apache, обслуживая 111 миллионов сайтов. Среди остальных сайтов 24% обслуживаются серверами Microsoft. Это распределение рыночных долей между серверами Apache и Microsoft на протяжении последних пяти лет практически не изменилось. Более подробную информацию о рыночных долях этих веб-серверов можно найти на сайте

```
news.netcraft.com/acrhives/web_server_survey.html
```

Полезную информацию для сравнения существующих HTTP-серверов содержит сайт `serverwatch.com/stypes` (выберите в меню **Server Type** команду **web**). Ниже указано несколько факторов, которые следует учитывать при выборе.

- Надежность
- Производительность
- Своевременность обновлений и исправлений ошибок
- Доступность исходного кода
- Уровень коммерциализации или поддержки сообщества
- Цена
- Безопасность и контроль доступа
- Возможность использования в качестве прокси-сервера
- Возможность шифрования данных

Код HTTP-сервера Apache распространяется свободно и доступен на сайте Apache Group `apache.org`. Если вы не желаете заниматься компиляцией кода, можно инсталлировать пакет готовых исполняемых модулей (подсказки в табл. 23.4). Некоторые поставщики не употребляют слово Apache, но, по существу, вы получите именно его.

**Таблица 23.4. Адреса модулей веб-сервера Apache**

| Система | Каталог | Рекомендуемый источник |
|---|---|---|
| Linux | /usr/sbin | Инсталлируется как часть стандартного дистрибутива |
| Solaris | /usr/apache2 | Инсталлируется как часть стандартного дистрибутива |
| HP-UX | /opt/apache | Инсталлируется HP-UX 11i "Web Server suite" |
| AIX | /usr/IBMIHS | Инсталлируется продукт IBM HTTPServer |

## Инсталляция сервера Apache

Если вы решили загрузить исходный код сервера и скомпилировать его самостоятельно, выполните сценарий **configure**, который включен в дистрибутив. Этот сценарий автоматически определяет тип операционной системы, а также устанавливает соответствующие **make**-файлы проекта. Потребуется указать каталог, в котором должен располагаться сервер Apache. Для этого предназначена опция **-prefix**.

```
$ ./configure --prefix=/etc/httpd
```

Если вы не укажете префикс, то по умолчанию будет использоваться каталог **/usr/local/apache2**.

Для того чтобы увидеть список возможных аргументов, выполните команду **config -help**. Большинство этих аргументов содержит опции **--enable-***модуль***=** и **--disable-***модуль***=**, включающие и отключающие функциональные компоненты, существующие внутри веб-сервера.

Задав опцию **--enable-***модуль***=shared** (или **--enable-mods-shared=all**, чтобы сделать совместно используемыми все модули), можно также скомпилировать модули в файлы динамических совместно используемых объектов. Это позволит вам позднее решить, какие модули включить, а какие исключить; во время выполнения загружаются только модули, указанные в конфигурации **httpd**. Эта конфигурация по умолчанию задается для бинарного пакета Apache — все модули компилируются в совместно используемые объекты и динамически загружаются при запуске веб-сервера Apache.

Единственным недостатком совместно используемых библиотек является более долгое время загрузки и небольшое снижение производительности (обычно менее чем на 5%). Для большинства сайтов возможность "на лету" добавлять новые модули и включать существующие модули без перекомпиляции перевешивает снижение производительности.

Полный список стандартных модулей приведен на сайте `httpd.apache.org/docs-2.2/mod`.

Несмотря на то что набор модулей, предлагаемый по умолчанию, является вполне приемлемым, можно также включить модули, указанные в табл. 23.5.

**Таблица 23.5. Полезные модули Apache, которые не включены по умолчанию**

| Модуль | Назначение |
|---|---|
| auth_dbm | Использует базу данных DBM для управления доступом со стороны пользователей/групп (рекомендуется для обеспечения индивидуального доступа пользователей на основе паролей) |
| rewrite | Переписывает URL-адреса на основе регулярных выражений |
| expires | Позволяет включать в документ дату его истечения |
| proxy | Конфигурирует Apache в качестве прокси-сервера |
| mod_ssl | Включает поддержку протокола Secure Socket Layer[a] для HTTPS |

[a] Этот протокол известен также под именем Transport Layer Security или RSL (раздел 23.4).

Аналогичным образом можно отключить модули, перечисленные в табл. 23.6. Отключая неиспользуемые модули, следует исходить из соображений безопасности и повышения производительности.

После завершения сценария **configure** выполните команды **make** и **make install** для фактической компиляции и установки соответствующих файлов.

**Таблица 23.6. Модули Apache, которые могут быть отключены**

| Модуль | Функция |
|---|---|
| asis | Позволяет посылать файлы указанных типов без использования HTTP-заголовков |
| autoindex | Индексирует каталоги, в которых отсутствует начальная HTML-страница (например, **index.html**) |
| env | Позволяет устанавливать специальные переменные среды для CGI-сценариев |
| userdir | Разрешает пользователям иметь собственные HTML-каталоги |



К сожалению, компиляция веб-сервера Apache в системе AIX не такая простая. Советы и приемы для этой инсталляции приведены на сайте `people.apache.org/~trawick/apache-2-on-aix.html`.

## Конфигурирование сервера Apache

После инсталляции сервера необходимо сконфигурировать его с учетом выполняемых функций. Все конфигурационные файлы находятся в каталоге **conf** (например, **/etc/httpd/conf**). Необходимо проверить и настроить файл **httpd.conf**, разделенный на три раздела.

Первый раздел файла **httpd.conf** определяет глобальные настройки, например пул сервера, TCP-порт, через который HTTP-сервер принимает запросы (обычно это порт 80, но можно на одном компьютере запустить несколько HTTP-серверов, подключенных к различным портам), а также параметры для загрузки динамических модулей.

Второй раздел описывает сервер "по умолчанию", который будет отвечать на все запросы, не обработанные определениями `VirtualHost` (раздел 23.3). В нем находятся параметры конфигурации, включающие пользователя и группу, для которых будет работать сервер (не суперпользователь!), и инструкцию `DocumentRoot`, которая определяет корневой каталог для обслуживаемых документов. В этом разделе указывается также ряд специальных установок, связанных, в частности, с обработкой "специальных" URL-адресов, использующих синтаксис *~пользователь* для доступа к рабочему каталогу пользователя.

Глобальные параметры безопасности также устанавливаются во втором разделе конфигурации. Он содержит директивы, которые позволяют управлять доступом на уровне отдельных каталогов (директива `<Directory>`) или файлов (`<File>`). С их помощью можно предотвратить доступ к важным файлам через демон **httpd**. Необходимо определить, как минимум, два уровня управления доступом: на одном уровне охватывается весь каталог документов, а второй применяется только к главному каталогу документов. Для этого достаточно оставить без изменения параметры, установленные для веб-сервера Apache по умолчанию, хотя мы рекомендуем удалить параметр `AllowSymLinks`, чтобы предотвратить обращение демона **httpd** к вашему дереву документов по символическим ссылкам. (Ведь никто не хочет, чтобы кто-нибудь случайно создал символическую ссылку на каталог **/etc**, не так ли?) Более подробные советы по укреплению безопасности веб-сервера Apache можно найти на сайте

`httpd.apache.org/docs-2.2/misc/security_tips.html`.

Третий, и последний, раздел файла конфигурации настраивает виртуальные узлы. Эту тему мы обсудим в отдельном разделе.

Внеся изменения в конфигурацию, проверьте синтаксис конфигурационного файла, выполнив команду **httpd -t**. Если все в порядке, веб-сервер Apache ответит "Syntax OK". Если нет, проверьте опечатки в файле **httpd.conf**.

## Запуск сервера Apache

Для запуска сервера Apache можно запустить демон **httpd** вручную либо воспользоваться сценариями запуска системы. Последний способ предпочтительнее, поскольку он гарантирует, что веб-сервер перезапускается всякий раз при перезагрузке компьютера. Для ручного запуска сервера следует ввести примерно такую команду.

```
% apachectl start
```

Сценарии загрузки описаны в главе 3.

## Анализ регистрационных файлов

Разрабатывая свой веб-сайт, вы, вероятно, захотите собрать статистические показатели об его использовании, например количество обращений к каждой странице, среднесуточное количество запросов, процент ошибочных запросов и объем переданных данных. Убедитесь, что вы используете "комбинированный" формат регистрации (директивы `CustomLog` должны содержать слово `combined`, а не `common`). Комбинированный формат регистрации содержит информацию о домене, ссылающемся на ваш сайт (странице, с которой произошел переход на URL), и пользовательском агенте (браузере клиента и операционной системе).

Файлы регистрации доступа и ошибок находятся в каталоге **logs** сервера Apache. Эти файлы являются удобочитаемыми для человека, но они содержат так много информации, что для ее анализа необходима дополнительная программа. Существуют буквально сотни анализаторов регистрационных файлов, как свободных, так и коммерческих.

Заслуживают внимание два свободно распространяемых анализатора: Analog (`analog.cx`) и AWStats (`awstats.sourceforge.net`). Оба они позволяют получить действительно важную информацию.

Если вас интересует информация о трафике и привычках пользователей вашего веб-сайта, обратитесь к службе Google Analysics на сайте `analytics.google.com`. Эта служба требует, чтобы вы поместили небольшую заглушку на каждую веб-страницу, за которой хотите следить, но зато совершенно бесплатно собирает все данные и выполняет анализ инфраструктуры.[2]

## Высокопроизводительный хостинг

За последние несколько лет стало очевидно, что самым простым способом организации высокопроизводительного хостинга является оптимизация некоторых серверов для обслуживания статического содержимого.

Одним из способов решения этой проблемы является использование веб-сервера или веб-страницы, работающих в адресном пространстве ядра. Они повышают производительность работы, поскольку эти системы не копируют данные из пользовательского адресного пространства и обратно, прежде чем вернуть их инициатору запроса. Однако поскольку они работают в адресном пространстве ядра, возникает дополнительный риск безопасности системы. Мы рекомендуем использовать этот прием с предельной осторожностью.

Веб-сервер, работающий в адресном пространстве ядра, называется TUX. Он взаимодействует с традиционным веб-сервером, например Apache, доступным в некоторых дис-

[2] Разумеется, эта схема подразумевает, что компания Google получает доступ к вашему трафику, что может оказаться и хорошо, и плохо.

трибутивных пакетах системы Linux. Когда есть такая возможность, сервер TUX предоставляет статические страницы, не покидая адресного пространства ядра, как это делает демон **rpc.nfsd** при обслуживании файлов. Несмотря на то что сервер TUX был выпущен для системы Red Hat (в настоящее время компания Red Hat называет его Red Hat Content Accelerator), он распространяется на условиях лицензии GPL, поэтому он может использоваться в других дистрибутивах Linux. Правда, настраивать его не так-то легко. Подробности можно узнать по адресу `http://www.redhat.com/docs/manuals/tux`.

Для кеширования содержимого в адресном пространстве ядра системы Solaris компания SUN выпустила программу Solaris Network Cache and Accelerator (NCA). Программа NCA перехватывает трафик, входящий и исходящий от демона **httpd**, и кеширует статические страницы. Когда поступают последующие запросы на то же самое содержимое, они обслуживаются из кеша без использования демона **httpd**.

## 23.3. Виртуальные интерфейсы

Раньше компьютер с системой UNIX обычно служил сервером для одного веб-узла (например, `acme.com`). По мере роста популярности сети веб, практически каждый пользователь обзавелся собственным веб-узлом, и, как грибы после дождя, стали появляться тысячи новых компаний, занимающихся веб-хостингом.

🕮 Основные конфигурации интерфейса описаны в главе 14.

Провайдеры быстро осознали, что можно добиться существенной экономии средств и ресурсов, если на одном сервере размещать несколько узлов. Это позволило управлять группой узлов, таких как `acme.com`, `ajax.com`, `toadranch.com` и многие другие, используя одно и то же аппаратное обеспечение. На практике такой подход реализуется с помощью *виртуальных интерфейсов*.

Идея, лежащая в основе виртуальных интерфейсов, весьма проста: одиночный компьютер обслуживает больше IP-адресов, чем позволяют его физические сетевые интерфейсы. Каждый из "виртуальных" сетевых интерфейсов может иметь доменное имя, под которым он известен пользователям Интернета. Это позволяет единственному серверу обслуживать сотни веб-узлов.

Виртуальные интерфейсы позволяют демону идентифицировать соединения не только по номеру порта назначения (например, порта 80 для HTTP), но и по целевому IP-адресу. В настоящее время виртуальные интерфейсы получили широкое распространение и оказались полезными не только для веб-хостинга.

Протокол HTTP 1.1 реализует функциональные возможности, подобные виртуальным интерфейсам (официально это называется "виртуальные интерфейсы, не имеющие IP-адреса"), устраняя потребность в назначении уникальных IP-адресов веб-серверам или в конфигурировании специального интерфейса на уровне операционной системы. Этот подход позволяет совместно использовать IP-адреса, что особенно полезно, когда один сервер содержит сотни или тысячи начальных страниц (например, в случае университетских веб-узлов).

Однако такой подход нельзя назвать практичным для коммерческих узлов, поскольку уменьшается степень их масштабируемости (приходится изменять IP-адрес при перемещении узла на другой сервер) и возникает угроза безопасности системы (если доступ к узлу фильтруется брандмауэром на основе IP-адресов). Кроме того, виртуальные узлы,

основанные на именах, требуют, чтобы браузер поддерживал протокол SSL.[3] Видимо, из-за этого ограничения настоящие виртуальные интерфейсы будут применяться еще долго.

## Конфигурирование виртуальных интерфейсов

Настройка виртуального интерфейса проходит в два этапа. Сначала требуется создать виртуальный интерфейс на уровне TCP/IP. Как именно, зависит от версий системы UNIX; инструкции для разных версий системы UNIX приводятся ниже. На втором этапе необходимо сообщить серверу Apache об имеющихся виртуальных интерфейсах. Этот этап также описан ниже.

### *Виртуальные интерфейсы в системе Linux*

Виртуальные интерфейсы Linux обозначаются в формате *интерфейс:экземпляр*. Например, если интерфейс Ethernet называется `eth0`, то связанные с ним виртуальные интерфейсы именуются `eth0:0`, `eth0:1` и т.д. Все интерфейсы конфигурируются с помощью команды **`ifconfig`**. Например, команда

```
$ sudo ifconfig eth0:0 128.138.243.150 netmask 255.255.255.192 up
```

настраивает интерфейс `eth0:0` и закрепляет за ним адрес в сети 128.138.243.128/26. Для того чтобы назначенные виртуальные адреса стали постоянными, необходимо модифицировать сценарии запуска системы.

В системе Red Hat требуется для каждого виртуального интерфейса создать отдельный файл в каталоге **`/etc/sysconfig/network-scripts`**. Например, файл **`ifcfg-eth0:0`**, соответствующий приведенной выше команде **`ifconfig`**, может содержать такие строки.

```
DEVICE=eth0:0
IPADDR=128.138.243.150
NETMASK=255.255.255.192
NETWORK=128.138.243.128
BROADCAST=128.138.243.191
ONBOOT=yes
```

Подход, принятый в системе Ubuntu, аналогичен подходу, используемому в системе Red Hat, за исключением того, что определения интерфейсов должны содержаться в файле **`/etc/network/interfaces`**. Записи, соответствующие интерфейсу `eth0:0`, в нашем примере должны выглядеть следующим образом.

```
Iface eth0:0 inet static
   address 128.138.243.150
   netmask 255.255.255.192
   broadcast 128.138.243.191
```

В системе SuSE можно создать виртуальные интерфейсы либо с помощью программы YaST, либо путем редактирования интерфейсных файлов. Для использования программы YAST сначала на вкладке Global Options раздела Network Settings выберите команду Traditional method with ifup.

[3] Относительно новая функциональная возможность Server Name Indication (SNI) позволяет использовать протокол SSL с виртуальными узлами, но старые браузеры этого делать не могут.

В системе SUSE IP-адреса каждого интерфейса конфигурируются в одном файле. Для редактирования конфигурации поищите в каталоге **/etc/sysconfig/network** файлы, имена которых начинаются словами **ifcgh-***имя_интерфейса*.

Например, в показанном ниже файле определяются интерфейсы eth0 и eth0:0.

```
IPADDR_1=128.138.243.149
NETMASK_1=255.255.255.192
STARTMODE_1="auto"
LABEL_1=0
IPADDR_2=128.138.243.150
NETMASK_2=255.255.255.192
STARTMODE_2="auto"
LABEL_2=1
```

Суффиксы, следующие за именами IPADDR и NETMASK (в данном случае _1 и _2), не обязательно должны быть представлены числами, но для согласованности такое соглашение является вполне приемлемым. Для того чтобы виртуальные интерфейсы распознавались, необходимо отредактировать файл **/etc/sysconfig/network/config** и установить параметр NETWORKMANAGER="no".

### *Виртуальные интерфейсы в системе Solaris*

Система Solaris поддерживает виртуальные интерфейсы ("вспомогательные интерфейсы") посредством концепции физического интерфейса и логического модуля. Например, если hme0 — это имя физического интерфейса, то hme0:1, hme0:2 и так далее — это имена соответствующих виртуальных интерфейсов. По умолчанию с каждым физическим интерфейсом может быть связано до 256 виртуальных сущностей. Если вы хотите изменить это ограничение, используя команду **ndd**, измените параметр ip_addrs_per_if (описание команды **ndd** см. в разделе 14.13).

Для конфигурирования виртуального интерфейса просто примените команду **ifconfig** к одному из виртуальных имен. (Соответствующий физический интерфейс в этот момент уже должен быть "подключен".) В большинстве случаев систему настраивают так, чтобы команда **ifconfig** применялась к виртуальным интерфейсам во время загрузки.

Рассмотрим пример, в котором компьютер под управлением системы Solaris имеет адрес в пространстве частных адресов во внутренней частной виртуальной сети (VPN) и внешний адрес — в Интернете, причем оба адреса связаны с одним и тем же физическим устройством hme0. Для того чтобы эти интерфейсы автоматически конфигурировались во время загрузки, администратор должен отредактировать два файла имен узлов: **/etc/hostname.hme0** и **/etc/hostname.hme0:1**.

```
$ ls -l /etc/host*
-rw-r--r--   1 root   10 Nov 4   10:19   /etc/hostname.hme0
-rw-r--r--   1 root   16 Dec 21  19:34   /etc/hostname.hme0:1
```

Файлы имен узлов могут содержат либо имена узлов из файла **/etc/hosts** либо IP-адреса. В данном случае администратор использовал каждую из этих возможностей.

```
$ cat /etc/hostname.hme0
overkill
$ cat /etc/hostname.hme0:1
206.0.1.133
$ grep overkill /etc/hosts
10.1.2.9 overkill overkill.domain
```

Во время загрузки оба адреса конфигурируются автоматически (вместе со шлейфовым адресом, который мы пропустили).

```
$ ifconfig -a
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu
  1500 inet 10.1.2.9 netmask ffffff00 broadcast 10.1.2.255
hme0:1: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu
  1500 inet 206.0.1.133 netmask ffffff80 broadcast 206.0.1.255
```

### *Виртуальные интерфейсы в системе HP-UX*

В системе HP-UX все виртуальные интерфейсы можно добавлять с помощью команды **ifconfig**. Ее синтаксис очень похож на синтаксис, используемый в системе Solaris. Например, для того чтобы добавить первый интерфейс, следует выполнить такую команду.

```
$ sudo ifconfig lan0:1 192.168.69.1 up
```

### *Виртуальные интерфейсы системы AIX*

В системе AIX, для того чтобы добавить дополнительный IP-адрес для интерфейса, можно создать "псевдоним". Например, для того чтобы добавить 192.168.1.3 как виртуальный IP-адрес интерфейса `en0`, можно использовать команду **ifconfig**.

```
$ sudo ifconfig en0 192.168.1.3 netmask 255.255.255.0 alias
```

Однако этот псевдоним является временным. Для того чтобы создать постоянный виртуальный IP-адрес, следует выполнить команду **chdev**.

```
$ sudo chdev -l en0 -a alias4=192.168.1.3,255.255.255.0
```

## Передача серверу Apache информации о виртуальном интерфейсе

После создания виртуальных интерфейсов с помощью команды **ifconfig** требуется сообщить серверу Apache о том, какие документы должны обрабатываться при попытке подключения клиента к каждому интерфейсу (IP-адресу). Это можно сделать с помощью директивы `VirtualHost` файла **httpd.conf**. Каждому сконфигурированному виртуальному интерфейсу должна соответствовать одна такая директива. Приведем пример.

```
<VirtualHost 128.138.243.150>
  ServerName www.company.com
  ServerAdmin webmaster@www.company.com
  DocumentRoot /var/www/htdocs/company
  ErrorLog logs/www.company.com-error_log
  CustomLog logs/www.company.com-access_log combined
  ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
</VirtualHost>
```

После подключения клиента к виртуальному узлу 128.138.243.150 будут обрабатываться документы из каталога **/var/www/htdocs/company**. Для настройки параметров виртуального узла в раздел `VirtualHost` может включаться почти любая директива веб-сервера Apache. Относительные пути к каталогам, включая пути для директив `DocumentRoot`, `ErrorLog` и `CustomLog`, интерпретируются в контексте `ServerRoot`.

Если виртуальные узлы имеют имена, то множественные имена в системе DNS ссылаются на один и тот же IP-адрес. Конфигурация веб-сервера Apache остается прежней,

но необходимо задать основной IP-адрес, который веб-сервер Apache должен прослушивать в ожидании входящих запросов к именованному виртуальному узлу, и не указывать IP-адрес в разделе `VirtualHost`.

```
NameVirtualHost 128.138.243.150

<VirtualHost *>
  ServerName www.company.com
  ServerAdmin webmaster@www.company.com
  DocumentRoot /var/www/htdocs/company
  ErrorLog logs/www.company.com-error_log
  CustomLog logs/www.company.com-access_log combined
  ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
</VirtualHost>
```

В этой конфигурации веб-сервер Apache ищет заголовки HTTP, чтобы определить требуемый сайт. Сервер прослушивает запросы к сайту `www.company.com` на его основном IP-адресе 128.138.243.150.

# 23.4. Протокол Secure Sockets Layes

Протокол SSL защищает соединения между веб-сайтом и клиентским браузером. Эту технологию используют универсальные указатели ресурса, начинающиеся с префикса `https://`. Протокол SSL использует алгоритмы криптографии для предотвращения перехвата, взлома и подделки сообщений.

Браузер и сервер используют схему аутентификации с помощью сертификата, после которой они переключаются на более быстродействующую схему шифрования для защиты реального соединения.

Протокол SSL выполняется на отдельном уровне, лежащем ниже уровня протокола приложения HTTP. Он просто обеспечивает защиту соединения и не вмешивается в трансакцию HTTP. Благодаря такой аккуратной схеме, протокол SSL может защищать не только HTTP, но и другие протоколы, такие как SMTP и FTP. Более подробную информацию можно найти в Википедии в статье "Secure Sockets Layer."[4]

В первые годы использования протокола SSL большинство ключей симметричного шифрования были относительно слабыми и состояли из 40 бит, потому что правительство США наложило ограничения на экспорт криптографической технологии. После многолетних переговоров и судебных исков, правительство ослабило некоторые ограничения экспорта, позволив реализовать протокол SSL с использованием 128-битовых симметричных ключей.

## Генерирование файла Certificate Signing Request

Владелец веб-сайта, использующий протокол SSL, должен сгенерировать цифровой файл Certificate Signing Request (CSR), содержащий открытый ключ и название компании. Этот "сертификат" должен быть "подписан" доверенным источником, известным как Certificate Authority (CA). Сертификат, подписанный источником сертификатов, содержит открытый ключ сайта, а также подтверждение подлинности источника.

[4] Transport Layer Security (TLS) — это протокол, являющийся преемником протокола SSL. Он реализован во всех современных браузерах. Однако сообщество пользователей сети веб попрежнему называет его SSL.

Веб-браузеры имеют встроенные списки источников CA, подписанные сертификаты которых они будут принимать. Браузер, знающий источник сертификата для вашего сайта, может верифицировать подпись вашего сертификата и получить ваш открытый ключ, тем самым позволив посылать сообщения, которые может расшифровать только ваш сайт. Кроме того, вы действительно можете подписывать ваш собственный сертификат. Получив сертификат от какого-либо непризнанного источника, большинство браузеров сообщает пользователю о том, что этот сертификат является подозрительным. В коммерческих приложениях такая ситуация, очевидно, представляет проблему. Однако, если вы хотите подписывать сертификаты для внутреннего использования и тестирования, изучите документ `httpd.apache.org/docs/2.2/ssl/ssl_faq.html#aboutcerts`.

Вы можете получить подпись сертификата от любого источника сертификатов. Введите в поисковой строке Google слова "SSL certificate" и выберите любую ссылку. Реальное различие между источниками сертификатов заключается в объеме работы, которую они выполняют для проверки вашей идентичности; гарантиях, которые они предоставляют; и количестве браузеров, которые они поддерживают изначально (большинство источников сертификатов поддерживают практически все браузеры).

Создание сертификата, который будет послан источнику сертификатов, является довольно простой процедурой. Необходимо инсталлировать пакет OpenSSL, который по умолчанию есть в большинстве систем. Затем необходимо выполнить следующие действия.

Сначала, создайте 1024-битовый закрытый ключ RSA для своего веб-сервера Apache.

```
$ openssl genrsa -des3 -out server.key 1024
```

Вам предложат войти и подтвердить пароль для шифрования серверного ключа. Скопируйте файл **`server.key`** в безопасное место (которое доступно только суперпользователю) и запомните введенный пароль. Интересующиеся пользователи могут увидеть многочисленные детали ключа с помощью следующей команды.

```
$ openssl rsa -noout -text -in server.key
```

Затем создайте файл Certificate Signing Request (CSR), содержащий серверный ключ, который вы только что сгенерировали.

```
$ openssl req -new -key server.key -out server.csr
```

Когда увидите приглашение ввести "стандартное имя", введите полностью определенное имя домена сервера. Например, если ваш сайт имеет URL-идентификатор `https://company.com`, введите в качестве стандартного имени строку "company.com". Обратите внимание на то, что для каждого узла нужен отдельный сертификат, даже для узла "www.company.com", который отличается от "company.com." Компании обычно регистрируют только стандартное имя; они должны гарантировать, что каждая ссылка по протоколу SSL ссылается именно на этот узел.

Детали сгенерированного файла CSR можно увидеть, выполнив такую команду.

```
$ openssl req -noout -text -in server.csr
```

Теперь можно послать файл **`server.csr`** выбранному вами источнику сертификатов. Сохранять его локальную копию не обязательно. Подписанный файл CSR, возвращенный источником сертификатов CA, должен иметь расширение `.crt`. Поместите подписанный сертификат в каталог, содержащий ваши файлы конфигурации демона **`httpd`**, например в каталог **`/usr/local/apache2/conf/ssl.crt`**.

## Конфигурация веб-сервера Apache для использования протокола SSL

Запросы по протоколу HTTP приходят на порт 80, а запросы по протоколу HTTPS используют порт 443. Трафики протоколов HTTPS и HTTP можно обслуживать одним и тем же процессом веб-сервера Apache. Однако протокол SSL не работает с именованными виртуальными узлами; каждый виртуальный узел должен иметь конкретный IP-адрес. (Это ограничение является следствием внутренней структуры протокола SSL.)

Для того чтобы настроить веб-сервер Apache на использование протокола SSL, сначала примите меры, чтобы модуль SSL был включен в файле **httpd.conf**, найдя или добавив следующую строку.

```
LoadModule ssl_module libexec/mod_ssl.so
```

Затем добавьте директиву `VirtualHost` для порта SSL.

```
<VirtualHost 128.138.243.150:443>
   ServerName www.company.com
   ServerAdmin webmaster@www.company.com
   DocumentRoot /var/www/htdocs/company
   ErrorLog logs/www.company.com-ssl-error_log
   CustomLog logs/www.company.com-ssl-access_log combined
   ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
   SSLEngine on
   SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
   SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server.key
</VirtualHost>
```

Обратите внимание на цифры :443 после IP-адреса и директивы протокола SSL, указывающие веб-серверу Apache, где искать ваш закрытый ключ и подписанный сертификат. Запустив веб-сервер Apache, вы получите сообщение с просьбой ввести пароль для вашего файла **server.key**. Благодаря этому демон **httpd** больше не сможет запускаться автоматически при загрузке компьютера. Если хотите, можно удалить шифрование из вашего закрытого ключа, чтобы устранить необходимость вводить пароль.

```
$ cp server.key server.key.orig
$ openssl rsa -in server.key.orig -out server.key
$ chmod 400 server.key server.key.orig
```

Разумеется, каждый, кто получит копию вашего расшифрованного ключа, затем сможет выдать свой сайт за ваш.

Более подробную информацию о протоколе SSL можно найти на следующих страницах.

```
httpd.apache.org/docs-2.2/ssl/ssl_faq.html
httpd.apache.org/docs/2.2/mod/mod_ssl.html
```

# 23.5. Кеширование и прокси-серверы

Развитие Интернета и увеличение объема находящейся в нем информации по-прежнему происходят очень быстро. Следовательно, пропускная способность глобальных каналов и мощность вычислительных ресурсов, требуемых для их обслуживания, также резко возрастают. Как с этим справиться?

Единственным методом справиться с этим является использование репликации. Независимо от того, на каком уровне — национальном, региональном или корпора-

тивном — находится информация, по мере развития Интернета требуется перемещать данные на ближние узлы. Не имеет смысла передавать одну и ту же популярную веб-страницу, скажем, из Австралии в США миллионы раз в день, используя весьма дорогостоящий международный канал связи. Должен существовать способ сохранять эту информацию после ее однократной передачи по каналу.

К счастью, такой способ есть, хотя бы на уровне сайтов. Веб-прокси позволяет кешировать данные и управлять внешними запросами к содержимому вашего веб-сайта.

Схема работы веб-прокси такова. Клиентские веб-браузеры подключаются к прокси-серверу, чтобы запросить объект из Интернета. Прокси-сервер выполняет запрос от имени клиента (либо извлекает объект из кеша) и возвращает результат клиенту. Прокси-серверы этого типа часто используются для усиления безопасности или фильтрации содержимого.

В системе, где работает прокси-сервер, непосредственный доступ в Интернет через брандмауэр нужен лишь одному компьютеру. В компьютерных классах средних школ прокси-серверы зачастую выполняют фильтрацию содержимого, чтобы дети не могли получить доступ к неподобающим материалам. В настоящее время имеется множество коммерческих и бесплатных прокси-серверов. Некоторые из них основаны исключительно на программном обеспечении, а другие встроены в аппаратные устройства. Обширный список технологий прокси-сервера можно найти на странице `web-caching.com/proxy-caches.com`.

В следующих разделах описывается пакет Squid Internet Object Cache[5], популярный автономный кеш. Мы также кратко обрисуем функциональные возможности прокси, которыми обладает модуль mod_cache веб-сервера Apache.

## Использование кеша Squid и прокси-сервера

Squid — это кеширующий прокси-сервер, поддерживающий несколько протоколов, включая HTTP, FTP и SSL.

Служба прокси — это, конечно, хорошо, но действительно преимуществом сервера Squid являются его кеширующие возможности. Сервер Squid не только кеширует информацию, получаемую в результате выполнения запросов локальных пользователей, но и формирует иерархию серверов[6]. Группы серверов Squid применяют протокол ICP (Internet Cache Protocol — протокол кеширования в Интернете) для передачи друг другу сведений о содержимом своих кеш-буферов.

Это позволяет администратору создать систему, в которой локальные пользователи взаимодействуют с кеширующим сервером организации с целью получения веб-содержимого. Если другой пользователь организации уже запрашивал эти же данные, то при повторном обращении к ним будет возвращена их копия, причем скорость передачи окажется сравнима со скоростью работы локальной сети (обычно 10 или 100 Мбит/с). При отсутствии на локальном сервере Squid требуемой информации она запрашивается у регионального кеширующего сервера. Как и в случае локального сервера, если какой-либо пользователь регионального сервера уже запрашивал объект раньше, то этот объект будет немедленно взят из кеша. При отсутствии объекта сервер обратится к "вышестоя-

[5] Почему "Squid" (кальмар — Примеч. ред.)? По версии авторов, "потому что все хорошие имена уже были заняты".

[6] К сожалению, некоторые узлы помечают все свои страницы как некешируемые, что препятствует эффективной работе сервера Squid. Кроме того, сервер не кеширует динамически формируемые страницы.

щему" серверу (обслуживающему страну или континент) и т.д. В результате производительность обработки запросов существенно повышается, и пользователи это чувствуют.

Применение пакета Squid экономически выгодно. Поскольку пользователи часто запрашивают в Интернете одну и ту же информацию, в крупных организациях наблюдается значительное дублирование внешних веб-запросов. Эксперименты показали, что установка кеширующего сервера приводит к уменьшению внешнего трафика на 40%.

Для эффективного использования сервера Squid, вероятно, потребуется заставить пользователей использовать кеш. Либо конфигурируйте прокси по умолчанию с помощью службы Active Directory (в среде Windows), либо настройте ваш маршрутизатор на перенаправление всего веб-трафика в кеш Squid с помощью протокола Web Cache Communication Protocol (WCCP).

## Инсталляция сервера Squid

Сервер Squid довольно легко инсталлировать и конфигурировать. Поскольку серверу нужно свободное пространство для кеша, потребуется выделенный компьютер, имеющий достаточный объем оперативной и дисковой памяти. Приемлемая конфигурация такова: 32 Гбайт ОЗУ и 8 Тбайт на жестком диске.

Пакет доступен в виде бинарных скомпилированных модулей. Кроме того, свежую копию пакета Squid можно загрузить с узла `squid-cache.org`. Во втором случае после распаковки инсталляционного дистрибутива запустите сценарий **`configure`**, находящийся в начальном каталоге. По умолчанию предполагается, что пакет будет установлен в каталог **`/usr/local/squid`**. Если требуется задать другой каталог, воспользуйтесь опцией **`--prefix=`***`каталог`* сценария **`configure`**. После завершения сценария выполните команды **`make`** и **`make install`**.

Далее потребуется модифицировать файл конфигурации **`squid.conf`**. Обратитесь к файлу **`QUICKSTART`**, находящемуся в дистрибутивном каталоге, для получения перечня изменений, которым должен подвергнуться имеющийся файл **`squid.conf`**.

Нужно также выполнить команду **`squid -z`**, чтобы сформировать и предварительно очистить структуру каталогов, в которой будут храниться кешированные веб-страницы. В конце можно запустить сервер вручную с помощью сценария **`RunCache`**, хотя обычно он вызывается из сценариев запуска системы, чтобы сервер Squid автоматически запускался на этапе начальной загрузки.

Для тестирования сервера Squid необходимо указать его в качестве прокси-сервера для веб-браузера.

## Настройка обратного прокси с помощью веб-сервера Apache

Для обеспечения безопасности и равномерного распределения нагрузки иногда полезно создавать веб-прокси для обработки *входящих* запросов (т.е. запросов, которые поступают от браузеров из Интернета к вашему серверу). Поскольку эта ситуация является противоположной по отношению к обычному использованию веб-прокси (подразумевающему обработку запросов, исходящих от браузеров в вашей сети), такая инсталляция называется обратным прокси (reverse proxy).

🕮 Сети DMZ описаны в главе 22.

Одна из распространенных конфигураций подразумевает размещение обратного прокси в вашей сети DMZ, чтобы иметь доступ к запросам, поступающим из Интернета,

к вашим службам, например электронной почте, основанной на использовании сети веб. Затем прокси передает эти запросы соответствующим внутренним серверам. Этот подход имеет несколько преимуществ.

- Это исключает соблазн установить прямые соединения с серверами, не принадлежащими сети DMZ.
- Достаточно настроить только один DMZ-сервер, а не по одному серверу для каждой службы, доступной извне.
- Можно управлять доступными указателями URL с центрального пункта, повышая безопасность.
- Входящие запросы можно регистрировать для мониторинга и анализа.

Конфигурирование веб-сервера Apache для обеспечения обратного прокси является относительно простым. В разделе `VirtualHost` файла **`httpd.conf`** веб-сервера Apache следует использовать директивы `ProxyPass` и `ProxyPassReverse`.

- Директива `ProxyPass` отображает удаленный указатель URL в пространство URL локального сервера, так что эта часть локального адресного пространства выглядит как зеркало удаленного сервера. (В этом сценарии "локальный" сервер — это DMZ-компьютер, а "удаленный" сервер — это сервер вашей внутренней сети.)
- Директива `ProxyPassReverse` скрывает реальный сервер с помощью "ретуширования" внешних HTTP-заголовков, проходящих через прокси.

Ниже приведен фрагмент конфигурации обратного прокси, который необходимо поместить в систему UNIX DMZ перед сервером Microsoft Outlook Web Access (OWA), обеспечивающим работу электронной почты с помощью сети веб.

```
<Location /rpc>
   ProxyPass https://wm.monkeypaw.com/rpc
   ProxyPassReverse https://wm.monkeypaw.com/rpc
   SSLRequireSSL
</Location>

<Location /exchange>
   ProxyPass https://wm.monkeypaw.com/exchange
   ProxyPassReverse https://wm.monkeypaw.com/exchange
   SSLRequireSSL
</Location>

<Location /exchweb>
   ProxyPass https://wm.monkeypaw.com/exchweb
   ProxyPassReverse https://wm.monkeypaw.com/exchweb
   SSLRequireSSL
</Location>

<Location /public>
   ProxyPass https://wm.monkeypaw.com/public
   ProxyPassReverse https://wm.monkeypaw.com/public
   SSLRequireSSL
</Location>

<Location /oma>
   ProxyPass https://wm.monkeypaw.com/oma
   ProxyPassReverse https://wm.monkeypaw.com/oma
   SSLRequireSSL
```

```
</Location>

<Location /Microsoft-Server-ActiveSync>
  ProxyPass https://wm.monkeypaw.com/Microsoft-Server-ActiveSync
  ProxyPassReverse https://wm.monkeypaw.com/Microsoft-Server-ActiveSync
  SSLRequireSSL
</Location>
```

В этом примере прокси-службы обеспечиваются только для нескольких указателей URL верхнего уровня: `/rpc`, `/exchange`, `/exchweb`, `/public`, `/oma` и `/Microsoft-Server-ActiveSync`. По соображениям безопасности желательно ограничить запросы, проходящие через прокси.

# 23.6. Расширение возможностей

Внезапная популярность сайта в сети веб может стать ночным кошмаром для системного администратора. Популярный блок или объявление на сайте `digg.com` может увеличить ваш веб-трафик в несколько раз. Даже "реальный" рост популярности может быстро исчерпать пропускную способность вашего локального сервера или сетевого соединения. Но не бойтесь; для решения этих проблем есть много решений.

## Облачные вычисления

🕮 Облачные вычисления описаны в главе 24.

Облачный хостинг предоставляет вам доступ к виртуальному экземпляру операционной системы, выбранной вами, без необходимости устанавливать аппаратное обеспечение в вашей организации. Фактически в этой схеме аппаратное обеспечение и его поддержка являются полностью абстрактными — вы можете иметь только самое общее представление о том, где на самом деле работает ваш виртуальный экземпляр.

Существует много провайдеров облачного хостинга, но компания Amazon является первопроходцем и остается рыночным лидером со своими службами Amazon Web Services (AWS), работающими на сайте `aws.amazon.com`. Менее чем за пять минут вы можете запустить новый экземпляр системы Linux или UNIX. Вы регистрируетесь с помощью утилиты **ssh**, чтобы управлять этой системой так, будто это ваш собственный информационный центр. А лучше всего то, что эта услуга невероятно дешевая (в настоящее время около 10 центов за час работы одного экземпляра в США для службы самого низкого уровня).

Несколько служб можно разместить на верхнем уровне облака, чтобы автоматически подключать или отключать серверы. Собственная служба масштабирования компании Amazon называется Auto Scaling. Интегрированные службы масштабирования продает компания RightScale (`rightscale.com`).

## Хостинг совместного размещения серверов

Совместное размещение серверов — еще один способ организации хостинга ваших систем в удаленном информационном центре, но в этом случае вы обычно или владеете серверным аппаратным обеспечением, или арендуете его. Этот способ предпочтителен, если стандарты или регламенты запрещают использовать облачные информационные центры (например, при использовании стандарта защиты информации в индустрии платежных карт PCI DSS) или необходимо специальное оборудование. Некоторые прило-

жения, связанные с интенсивным вводом и выводом данных, также лучше работают на специальном оборудовании, хотя виртуальный мир их быстро догоняет.

🕮 Более подробно уровни информационных центров описаны в главе 27.

Существуют сотни провайдеров совместного размещения серверов. Выберите один из них на требуемом вам ярусе в соответствии с рекомендациями организации Uptime Institute (`uptimeinstitute.org`). Коммерческие приложения обычно размещаются в информационных центрах третьего или четвертого уровня.

## Сети для распределения контента

Большинство контента в Интернете является статическим: изображения, документы, пакеты программного обеспечения. Помещая копии этих статических компонентов ближе к пользователям (в терминах сети), вы можете уменьшить или вообще исключить необходимость обращаться к данным из оригинального источника и выключить его из сети.

Система компьютеров, обеспечивающая эту возможность, называется сетью для распределения контента (content distribution network — CDN). Межконтинентальные сетевые соединения обычно перегружены, поэтому сети CDN особенно важны для обеспечения быстрого доступа к популярному контенту на других континентах.

Большинство сетей CDN действует как коммерческие службы, финансируемые провайдерами контента, желающими обеспечить доступность и оперативность своих сайтов без масштабирования их инфраструктуры. Наиболее успешной оказалась платформа CDN, управляемая компанией Akamai Technologies (`akamai.com`). Крупнейшими конкурентами этой компании являются недавно появившиеся компании Limelight (`limelightnetworks.com`) и Disney-owned EdgeCast (`edgecast.com`).

При правильной реализации использование сети CDN не представляет для конечного пользователя никаких трудностей. Некоторые объекты могут доставляться из относительно близкого сервера или кеша, а другие — непосредственно из источника. Однако все это стоит больших затрат. Включать службу CDN в свой план хостинга могут только богатые организации.

# 23.7. Упражнения

23.1. Сконфигурируйте виртуальный интерфейс на рабочей станции. Выполните команду **`ifconfig`** до и после этого и сравните результат. Можно ли обнаружить виртуальный интерфейс с помощью команды **`ping`**, выполненной на другом компьютере той же подсети? А другой сети? Объясните ваш ответ. (Необходим доступ с правами суперпользователя.)

23.2. Используя свой браузер, зайдите на популярный и содержательный сайт, например `abcnews.com`, и посмотрите на источник страницы (View⇨Page в браузере Firefox или Page⇨View в браузере IE). С помощью команды **`dig`** просмотрите записи DNS для узлов, заданных в указателе URL отдельного объекта. Можете ли вы определить, какие объекты хранятся в сети для распределения контента?

23.3. ✯ С помощью анализатора пакетов (**`tcpdump`**) прослушайте трафик двустороннего HTTP-диалога, связанного с передачей информации на сервер (например, с заполнением полей формы или поля поиска). Покажите, каким образом браузер посылает данные веб-серверу. (Необходим доступ с правами суперпользователя.)

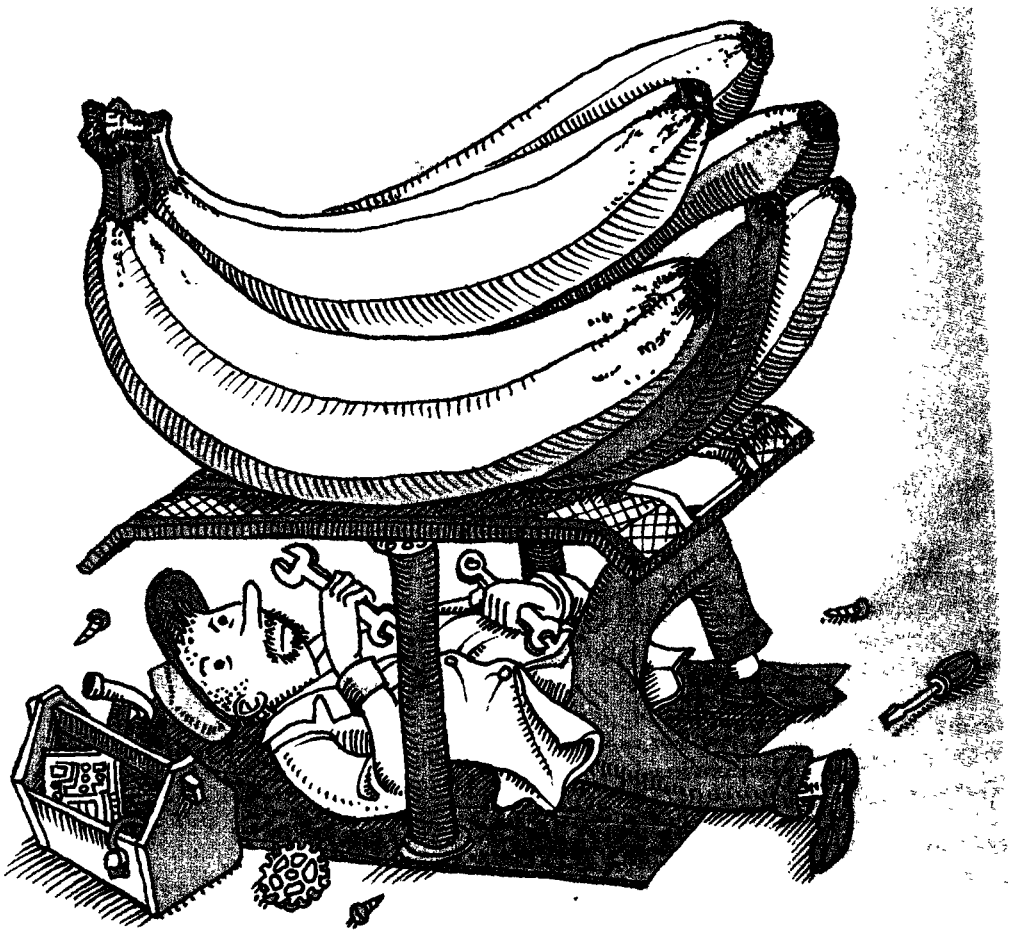
23.4. ★ Используйте анализатор пакетов для перехвата трафика, возникающего при открытии часто посещаемой веб-страницы, например начальной страницы портала `amazon.com` или `cnn.com`. Сколько отдельных TCP-соединений приходится открывать? Кто инициирует их? Является ли это наиболее эффективным способом использования протокола TCP? (Необходим доступ с правами суперпользователя.)

23.5. ★ Найдите журнальные файлы веб-сервера и проанализируйте их. Запросы каких типов поступали серверу за последние несколько часов? Какие ошибки произошли за этот период? Что можно сказать о безопасности сервера на основании содержимого журнальных файлов? (Может потребоваться доступ с правами суперпользователя.)

23.6. ★★ Инсталлируйте сервер Apache и создайте несколько веб-страниц. Проверьте с других компьютеров, работает ли сервер. Найдите журнальные файлы Apache, которые позволяют узнать, какие браузеры обращались к серверу. Настройте сервер Apache на выдачу некоторых веб-страниц через виртуальный интерфейс, созданный в упр. 23.1. (Необходим доступ с правами суперпользователя.)

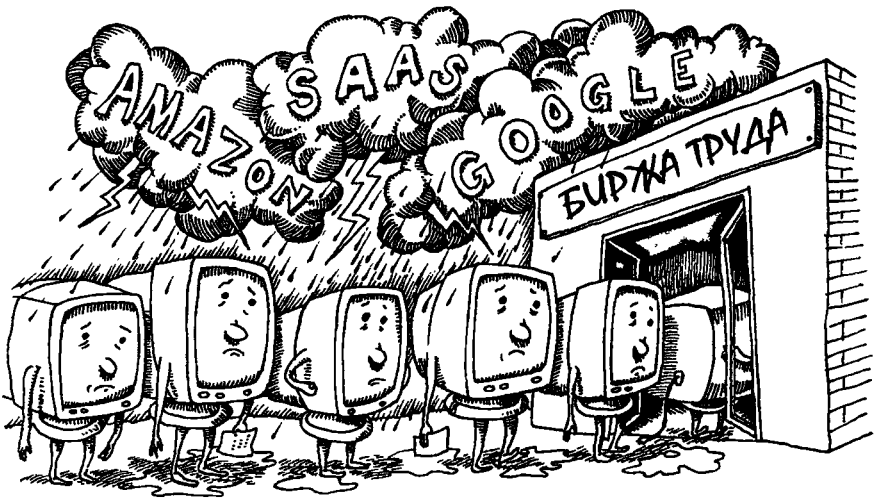
# ЧАСТЬ III

# Разное



# Глава 24

# Виртуализация




В то время как корпоративные центры обработки данных продолжают увеличивать количество серверов для удовлетворения информационных аппетитов современного бизнеса, системные администраторы стараются разрешить технический парадокс: как управлять существующими системами более эффективно, чтобы экономить электроэнергию, площадь и стоимость охлаждения, продолжая обслуживать сотни пользователей?

Разработчики программного обеспечения никогда не поощряли администраторов выполнять свои приложения на других платформах, ссылаясь на потенциальную несовместимость, а в некоторых случаях даже угрожая прекратить техническую поддержку. В результате возникло огромное количество узкоспециализированных серверов. По последним оценкам коэффициент использования среднестатистического сервера колеблется от 5 до 15% и продолжает снижаться по мере увеличения производительности серверов.

Выйти из этого затруднительного положения позволяет виртуализация: параллельное использование разных независимых операционных систем на одном и том же аппаратном обеспечении. Администраторы могут рассматривать каждую виртуальную машину как уникальный сервер, удовлетворяющий капризных поставщиков (в большинстве случаев) и одновременно уменьшающий затраты на работу центра обработки данных. Виртуализацию поддерживают многие платформы, а разработка специальных инструкций центрального процессора, предназначенных для поддержки виртуализации, и преобладание многоядерных процессоров резко повысили производительность. Виртуальные серверы легко инсталлировать и проще сопровождать (в расчете на один сервер), чем физические машины.

Со временем реализация виртуализации сильно изменилась, но основные концепции остаются прежними. Компания IBM использовала виртуальные машины еще в первых мейнфреймах, исследуя в 1960-х годах концепцию разделения времени и позволяя совместно использовать процессор и ресурсы для хранения данных с помощью абстрактно-

го уровня. Аналогичная технология, разработанная компанией IBM, была использована в мейнфреймах в средине 1970-х годов, пока в 1980-х не начался клиент-серверный бум. Эта технология оставалась доминантной на протяжении 1980–1990-х годов, пока проблемы, связанные с высокой стоимостью обслуживания и сложностью управления огромными фермами серверов, не возродили интерес к виртуализации современных систем. По общему мнению, путь для современного виртуального безумия проложила компания VMware, создав в 1999 году платформу для виртуализации на основе архитектуры Intel x86.

Сегодня технология виртуализации — это процветающий бизнес, открывающий перед разработчиками множество рыночных возможностей. Компания VMware остается явным лидером и предлагает продукты, предназначенные для бизнеса любого масштаба, а также программное обеспечение для управления организациями с высокой степенью виртуализации. Сообщество разработчиков программ с открытым исходным кодом ответило на этот вызов проектом Xen, получившим коммерческую поддержку компании XenSource, которая в настоящее время называется Citrix.

Выпустив систему Solaris 10, компания Sun предложила мощную технологию, основанную на концепциях зон и контейнеров и способную запускать более 8000 виртуальных систем на базе одной инсталляции Solaris. На этом рынке всего несколько игроков и десятки конкурирующих продуктов, каждый из которых занимает свою нишу.

🕮 Сети хранения данных описаны в главе 8.

Несмотря на то что главной темой этой главы является серверная виртуализация, аналогичные концепции можно применить ко многим другим областям информационной инфраструктуры, включая сети, хранилища данных, приложения и даже настольные системы. Например, при использовании сетей для хранения данных или хранилищ, объединенных в сеть, пулы дисков можно рассматривать как службу, при необходимости создавая дополнительный объем памяти. Применение виртуализации к настольным системам может оказаться полезным как для системных администраторов, так и для пользователей, поскольку каждому пользователю предоставляется среда, настроенная по его требованиям.

Широкие возможности виртуализации создали дополнительные сложности для нерасторопных администраторов систем UNIX и Linux. Их пугает возможность выбора между десятками платформ и конфигураций и необходимость определения долгосрочной стратегии. В этой главе мы начнем с определения терминов, используемых в технологиях виртуализации, продолжим обсуждением преимуществ виртуализации, подскажем, как принять наилучшее решение, удовлетворяющее ваши потребности, и, наконец, дадим несколько практических советов, касающихся наиболее популярного программного обеспечения для виртуализации, работающего в исследуемых нами операционных системах.

## 24.1. Виртуальный жаргон

Виртуализация имеет свои термины и концепции. Овладение этим языком является первым шагом к овладению разнообразными функциональными возможностями.

Предполагается, что операционные системы управляются аппаратным обеспечением, поэтому одновременное функционирование двух систем вызывает конфликты, связанные с совместным использованием ресурсов. Серверная виртуализация — это абстракция вычислительных ресурсов, позволяющая операционным системам работать, ничего не зная о базовом аппаратном обеспечении. Программное обеспечение для виртуализации

"дробит" физические ресурсы, такие как диски, память и центральный процессор, динамически предоставляя их для использования несколькими виртуальными машинами.

Администраторы системы UNIX должны понимать три разные парадигмы: полная виртуализация, паравиртуализация и виртуализация на уровне операционной системы. Каждая модель по-своему решает проблемы выделения ресурсов и доступа к аппаратному обеспечению, и каждая из них имеет свои преимущества и недостатки.

## Полная виртуализация

В настоящее время наиболее широко признанной является парадигма полной виртуализации. В этой модели операционная система не знает, что она функционирует на виртуализированной платформе, а между виртуальными машинами ("гостями") и аппаратным обеспечением инсталлируется гипервизор ("hypervisor"), известный также как монитор виртуальной машины.

Такие гипервизоры называются автономными (bare-metal), потому что они управляют аппаратным обеспечением. Гипервизор обеспечивает уровень эмуляции для всех аппаратных устройств узла. Гостевая операционная система не модифицируется. Гости выполняют прямые запросы к виртуализированному аппаратному обеспечению, и любые привилегированные инструкции, которые гостевые ядра пытаются выполнить, обрабатываются гипервизором.

Автономная виртуализация — наиболее безопасный вид виртуализации, потому что гостевые операционные системы изолированы от базового аппаратного обеспечения. Кроме того, не требуется вносить никаких изменений в ядра операционных систем, и гости могут выбирать разные базовые архитектуры. Если имеется программное обеспечение для виртуализации, то гость может функционировать на любой архитектуре процессоров. (Однако трансляция инструкций центрального процессора приводит к умеренному снижению производительности.)

Примером полной виртуализации является популярная технология VMware. Общая структура систем, созданных по технологии VMware, представлена на рис. 24.1.

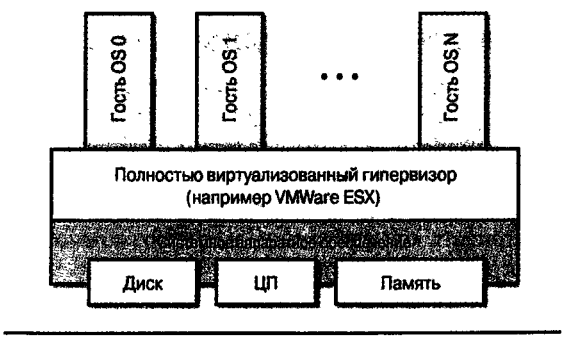



*Рис. 24.1. Архитектура полной виртуализации*

## Паравиртуализация

Паравиртуализация — это технология, используемая виртуальной платформой с открытым исходным кодом Xen. Как и полная виртуализация, паравиртуализация позволяет нескольким операционным системам согласованно функционировать на одной

машине. Однако ядра каждой операционной системы должны быть модифицированы, чтобы поддерживать "гипервызовы" ("hypercalls"), т.е. перевод определенных инструкций, предназначенных для центрального процессора. Приложения, действующие в адресном пространстве пользователя, не требуют модификации и выполняются на Xen-машинах естественным образом. Гипервизор в паравиртуализации используется точно так же, как и в полной виртуализации.

Уровень трансляции в паравиртуализированной системе связан с меньшими накладными расходами ресурсов, поэтому паравиртуализация ведет к номинальному повышению производительности. Однако необходимость модифицировать гостевые операционные системы резко снижает эффект и является основной причиной того, почему технология паравиртуализации Xen слабо поддерживается за пределами сообщества Linux и других ядер с открытым исходным кодом.

Среда паравиртуализации продемонстрирована на рис. 24.2. Она похожа на полностью виртуализированную систему, изображенную на рис. 24.1, в которой гостевые операционные системы взаимодействуют с гипервизором посредством определенного интерфейса и первый гость является привилегированным.

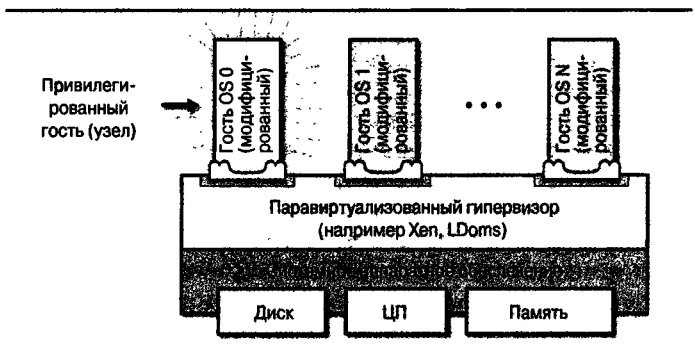



*Рис. 24.2. Архитектура паравиртуализации*

## Виртуализация на основе операционной системы

Системы виртуализации на основе операционной системы резко отличаются от предыдущих двух моделей. Вместо создания множественных виртуальных машин внутри физической системы, виртуализация на основе операционной системы позволяет операционной системе создать множественные изолированные окружения прикладных программ, относящихся к одному и тому же ядру. Виртуализацию на основе операционной системы следует рассматривать как функциональную возможность ядра, а не отдельный уровень абстракции программного обеспечения.

Поскольку в этой схеме нет реального уровня трансляции или виртуализации, накладные расходы ресурсов при таком способе виртуализации очень низкие. Большинство реализаций обеспечивает практически естественную производительность. К сожалению, этот тип виртуализации препятствует использованию нескольких операционных систем, поскольку все гости совместно используют одно и то же ядро (или "контейнеры", как принято говорить в данном контексте).[1] Разделы рабочих нагрузок (workload partitions)

[1] Это не совсем так. Контейнеры системы Solaris имеют функциональную возможность под названием "фирменные зоны" ("branded zones"), позволяющие выполнять бинарные модули системы Linux в ядре системы Solaris.

в системе AIX, а также контейнеры и зоны в системе Solaris представляют собой примеры виртуализации на основе операционной системы.

Архитектура виртуализации на основе операционной системы представлена на рис. 24.3.

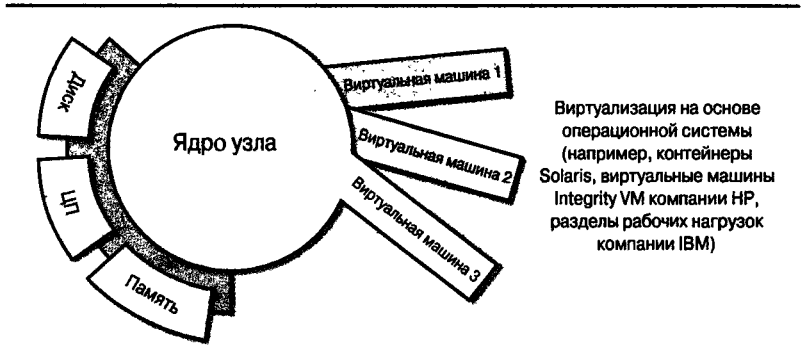



*Рис. 24.3. Архитектура виртуализации на основе операционной системы*

## Естественная виртуализация

Пытаясь обеспечить преимущество своему аппаратному обеспечению, ведущие компании–производители микропроцессоров AMD и Intel ведут острую конкурентную борьбу за лучшую поддержку виртуализации с помощью аппаратного обеспечения, т.е. за естественную ("native") виртуализацию. Обе компании предлагают центральные процессоры, которые содержат инструкции по виртуализации, что исключает необходимость в уровне трансляции для полной виртуализации или паравиртуализации. В настоящее время все основные игроки на рынке виртуализации могут воспользоваться этими возможностями центральных процессоров.

## Облачные вычисления

В дополнение к традиционной виртуализации, относительно недавно появилась технология, которую неформально (и в некоторой степени недоброжелательно) называют облачными вычислениями и считают альтернативой локальным серверным фермам. Облачные вычисления предлагают вычислительные мощности в качестве службы, которая оплачивается по привлекательно низкой цене на почасовой основе. Большая часть очевидных преимуществ этой технологии заключается в преобразовании ресурсов серверов в инфраструктуру, аналогичную коммунальным службам. Администраторы и разработчики никогда не видят реальное аппаратное оборудование, которое они используют, и ничего не знают о его структуре. Название этой технологии происходит от традиционного использования очертания облака для обозначения Интернета на сетевых диаграммах.

Поскольку эта книга предназначена для системных администраторов, мы рассмотрим облачные вычисления на уровне сервера, но приложения также могут перемещаться в облако (эта технология называется "программное обеспечение как служба" (software-as-a-service — SAAS)). Все программы для настольных систем, реализующие разнообразные функции — от рассылки электронной почты до деловой активности, — можно перепоручить сторонним организациям и ими можно управлять независимо.

Облачные службы обычно сопровождаются интерфейсом для управления, который настраивает мощности по требованию и позволяет резервировать новые системы одним щелчком мыши. Служба Elastic Compute Cloud (EC2) компании Amazon — это наиболее зрелая служба этого типа, относящаяся к первому поколению. Она пользуется популярностью среди компаний, предлагающих веб-платформы нового поколения. Любите ли вы его или ненавидите, но принцип предоставления информационных сервисов как коммунальных услуг (utility computing) привлекает скупых менеджеров как дешевая альтернатива центрам обработки данных и локализованной серверной инфраструктуре. Фанатики информационных технологий верят, что облачные технологии в различных формах — это будущее вычислительных систем.

Облачные вычисления основаны на идеях, некоторые из которых легли в основу виртуализации, но ее следует рассматривать как отдельный набор технологий со своими особенностями.

## Динамическая миграция

Последняя концепция, которую нам следует рассмотреть, — это возможность миграции виртуальных машин с одной физической машины на другую. Большинство программного обеспечения для виртуализации позволяют переносить виртуальные машины с одной функционирующей системы на другую, причем в некоторых случаях это происходит без прерывания обслуживания или потери соединения. Эта функциональная возможность называется динамической миграцией (live migration). Она полезна для равномерного распределения нагрузки, восстановления работы после сбоя, поддержки сервера и обеспечения общей гибкости системы.

## Сравнение технологий виртуализации

Несмотря на то что разные технологии виртуализации основаны на разных концепциях, все они в конце концов приводят к одинаковым результатам. Администраторы обращаются к виртуальным системам точно так же, как это происходит в обычном режиме работы в сети. Основные отличия заключаются в том, что проблемы с аппаратным оборудованием сказываются на всех операционных системах одновременно (поскольку все они совместно используют эти устройства), а проблемы, связанные с содержанием ресурсов, должны устраняться на том же уровне, на котором реализована виртуализация (т.е. в гипервизоре).

# 24.2. Преимущества виртуализации

Услышав столько комплиментов в адрес виртуальных вычислений, удивительно осознавать, что на ее разработку и коммерческую адаптацию ушло так много лет. Основными факторами, способствующими принятию виртуальных технологий, являются экономия средств, снижение потребления электроэнергии, упрощенное обеспечение бесперебойной работы и высокая техническая маневренность.

Стоимость является основным фактором во всех новых проектах в области информационных технологий, и, оценивая технологии виртуализации, бизнесмены видят краткосрочную выгоду, поскольку им не придется покупать несколько серверов.

Вместо приобретения новых серверов для реализации новых приложений, администраторы могут запустить новые виртуальные машины и сэкономить деньги на покупке физических устройств и их дальнейшем обслуживании. Кроме того, резко снижаются

затраты на охлаждение, поскольку виртуальные серверы не генерируют тепла, что также приводит к дополнительной экономии.

Упрощается работа центров обработки данных и удешевляется их обслуживание. Некоторые организации объединяют на одном виртуальном узле до 30 серверов, благодаря чему они экономят место для стоек.

Помимо прочего, виртуализация снижает нагрузку на окружающую среду. По некоторым оценкам, прожорливыми центрами обработки данных потребляется около одного процента электричества, производимого в мире.[2] Современные многоядерные центральные процессоры используются более эффективно, когда на них одновременно работает несколько виртуальных машин.

Бесперебойность бизнеса, т.е. способность компании переживать физический и логические кризисы с минимальными потерями, — это досадная и дорогостоящая проблема для системных администраторов. Сложные подходы к восстановлению системы после сбоя упрощаются, если виртуальные серверы могут перемещаться с одной физической машины на другую с помощью одной команды. Технологии такой миграции, поддерживаемые большинством платформ виртуализации, позволяют обеспечивать независимость приложений от физического местоположения.

Поскольку виртуальные серверы могут независимо обращаться к гипервизорам, которые их поддерживают, управление серверами разрывает связь с физической реальностью и может полностью описываться сценариями. Системные администраторы могут быстро удовлетворять потребности заказчиков в новых системах и приложениях, используя шаблонное обеспечение серверов. Сценарии могут автоматизировать и упростить задачи управления типичными виртуальными системами. Загрузку, прекращение работы и миграцию виртуального сервера можно автоматизировать с помощью сценариев оболочки и даже задать расписание его работы с помощью демона **cron**. Снятые с производства операционные системы и приложения можно переносить с устаревшего аппаратного обеспечения на более современные архитектуры.

Виртуализация увеличивает доступность. Динамическая миграция позволяет снимать физические серверы с обслуживания без прерывания работы службы. Обновление аппаратного обеспечения также не нарушает бизнес-процессы. Когда приходит время заменить устаревшую машину, виртуальная система немедленно мигрирует без болезненной модификации, инсталляции, тестирования и цикла переключений.

Виртуализация обеспечивает строгое разделение между средами для разработки, тестирования, размещения и производства даже в небольших компаниях. Исторически сложилось так, что поддержка этих раздельных сред для многих компаний была слишком дорогим удовольствием, хотя этого могут требовать законы и стандарты. Отдельные среды могут быть выгодными; например, специалисты, занимающиеся обеспечением качества продукции, могут легко восстанавливать базовую конфигурацию среды для тестирования.

С точки зрения немедленной отдачи, лишь немногие технологии предлагают так много преимуществ, как серверная виртуализация. Однако, как мы увидим в следующем разделе, виртуализация — это не панацея.

## 24.3. Практичный подход

Необходимо тщательно планировать, контролировать и реализовывать переход в виртуализованную среду. Несогласованный подход приведет к появлению неупорядо-

[2] Эта оценка приведена Джонатаном Куми (Jonathan Koomey) в его превосходном исследовании "Estimating total power consumption by servers in the U.S. and the world".

ченного набора неустойчивых и неуправляемых реализаций, которые принесут больше вреда, чем пользы. Более того, доверие заинтересованных лиц легко потерять: ошибки, сделанные на ранних этапах, могут усложнить будущие попытки перетянуть сопротивляющихся пользователей на новые платформы. Тише едешь — дальше будешь.

Важно правильно выбрать системы, в которых одни приложения виртуализировать удобнее, чем другие. Службы, которые уже интенсивно используются, лучше оставить на физической системе, по крайней мере на начальном этапе. Среди других служб, которые целесообразно оставить в покое, перечислим следующие.

- Серверы интенсивного резервирования ресурсов или регистрационные узлы.
- Приложения с высокой пропускной способностью, например системы для обнаружения взлома.
- Загруженные серверы для управления базами данных с вводом и выводом.
- Лицензионные приложения с аппаратной защитой от копирования.
- Приложения с особыми требованиями к аппаратному обеспечению, например медицинские системы или некоторые системы для сбора научных данных.

Хорошими кандидатами на виртуализацию являются следующие серверы.

- Веб-серверы с выходом в Интернет, посылающие запросы межплатформенному программному обеспечению или базе данных.
- Мало используемые изолированные серверы прикладных программ.
- Системы для разработки программ, например серверы сборки или контроля версий.
- Узлы для проверки качества продукции и среды для разворачивания программ.
- Системы ядерной инфраструктуры, такие как каталоги LDAP, серверы DHCP и DNS, сервер времени и шлюзы SSH.

Сначала организуйте миграцию небольшого количества менее важных систем. Это повысит доверие организации и обогатит опыт администратора. Очевидными претендентами на миграцию являются новейшие приложения, которые могут изначально иметь встроенные возможности для виртуализации. После стабилизации среды можно продолжить регулярный перенос систем. Для крупных организаций разумный темп миграции составляет от 25 до 50 серверов в год.

Планируйте соответствующую поддержку инфраструктуры в новой среде. Планы по миграции должны поддерживаться соответствующими объемами дисковой памяти и сетевыми ресурсами. Если несколько систем с одного узла окажутся в разных физических сетях, запланируйте соединение сетевых интерфейсов. Предусмотрите соответствующее оснащение для систем, которые будут использовать память в сети SAN. Правильно размещайте подобные системы на одном и том же физическом аппаратном обеспечении, чтобы упростить инфраструктуру. В заключение примите меры, чтобы каждая виртуальная машина имела запасную позицию, на которую ее можно перенести, если на первичной системе возникнут проблемы.

Не запускайте все важные службы на одном и том же физическом аппаратном обеспечении и не перегружайте системы слишком большим количеством виртуальных машин. Благодаря быстрому совершенствованию серверного оборудования администраторы имеют широкие возможности для виртуализации. Многоядерная многопроцессорная архитектура — очевидный выбор для виртуальных машин, поскольку они уменьшают потребность в переключателях контекста и облегчают выделение ресурсов центрального процессора. Для виртуальных сред основные производители выпускают новые лезвий-

ные серверы (blade server) и предлагают быстродействующие средства ввода и вывода, а также большие объемы памяти. Полупроводниковые дисковые устройства отлично подходят для виртуализации, потому что они обеспечивают быстрый доступ и потребляют мало электроэнергии.

# 24.4. Виртуализация с помощью системы Linux

За звание чемпиона по виртуализации системы Linux соперничают две платформы: Xen и KVM. В одном углу ринга — платформа Xen, устоявшаяся, хорошо документированная, с широкой поддержкой от авторитетных дистрибьюторов. В другом углу ринга — платформа KVM, которую Линус Торвальдс (Linus Torvalds) включил в ядро системы Linux. Количество ее поклонников возрастает. Кроме того, она получила поддержку со стороны систем Ubuntu и Red Hat.

В этом разделе мы не станем вмешиваться в их состязание, а сосредоточимся на вопросах администрирования систем, связанных с каждой из этих технологий.

## Введение в платформу Xen

Изначально Linux-ориентированная платформа Xen была разработана Яном Праттом (Ian Pratt) как исследовательский проект Кембриджского университета (University of Cambridge). Со временем она стала мощной платформой для виртуализации, которая, благодаря производительности, безопасности и дешевизне, оказалась способной конкурировать даже с коммерческими гигантами. Накладные расходы, связанные с эксплуатацией паравиртуального гипервизора виртуальной машины Xen, составляют 0,1–3,5%, что намного меньше, чем у полностью виртуализованных платформ.

Поскольку гипервизор Xen является программой с открытым исходным кодом, существует множество инструментов с разными уровнями поддержки ее функциональных возможностей. Исходный код платформы Xen можно загрузить на сайте `xen.org`. Кроме того, она входит во многие дистрибутивные пакеты.

Xen — это автономный гипервизор, который функционирует непосредственно на физическом аппаратном обеспечении. Работающая виртуальная машина называется доменом. Всегда существует по крайней мере один домен, который называется нулевым (или dom0). Нулевой домен имеет полный доступ к аппаратному обеспечению, управляет другими доменами и запускает драйверы всех устройств. Непривилегированные домены называются domU. Всеми доменами, включая dom0, управляет гипервизор Xen, ответственный за работу центрального процессора и управление памятью. Архитектуру платформы Xen дополняют демоны, инструменты и библиотеки, обеспечивающие взаимодействие между доменами domU, dom0 и гипервизором.

Несколько инструментов управления упрощают решение общих задач управления платформой Xen, таких как загрузка и остановка, конфигурирование и создание гостей. Xen Tools — это коллекция сценариев на языке Perl, которые упрощают создание доменов domU. Сценарий MLN, или Manage Large Networks, — это еще один сценарий на языке Perl, создающий виртуальные сети на основе ясных и понятных файлов конфигурации. ConVirt — это превосходный инструмент с графическим пользовательским интерфейсом для управления гостями. Он обеспечивает динамическую миграцию по принципу "drag-and-drop", мультисерверную поддержку без агентов, доступ к приборным интерфейсам и их конфигурирование, а также шаблонное создание новых виртуальных машин. Для поклонников командной строки предусмотрена также встроенная утилита `xm`.

Дистрибутивные пакеты Linux по-разному поддерживают платформу Xen. Компания Red Hat сначала включала платформу Xen в свои дистрибутивные пакеты, пока не отказалась от нее в пользу конкурирующего программного обеспечения KVM. Платформа Xen хорошо поддерживается системой SUSE Linux, особенно в версии Enterprise 11. Разработчики канонической системы Ubuntu Linux выбрали странный подход к платформе Xen, нерешительно поддерживая ее вплоть до появления версии 8.10, в которой они отказались от платформы Xen и предпочли программное обеспечение KVM (хотя платформа Xen по-прежнему упоминается в документации). После инсталляции способ использования платформы Xen мало чем отличается в разных дистрибутивных пакетах. В общем, для крупного развертывания виртуализации на основе платформы Xen мы рекомендуем использовать системы Red Hat или SUSE.

## Основы платформы Xen

Сервер Linux Xen требует большого количества демонов, сценариев, конфигурационных файлов и инструментов. Большинство элементов этой головоломки перечислено в табл. 24.1. Конфигурационный файл каждого домена на платформе Xen в каталоге **/etc/xen** содержит информацию о том, какие виртуальные ресурсы доступны для домена domU, например дисковые устройства, центральный процессор, память и сетевые интерфейсы. Каждый домен domU имеет свой конфигурационный файл. Формат этого файла очень гибкий и позволяет администраторам точно контролировать ограничения, накладываемые на каждого гостя. Если в подкаталог **auto** добавлена символическая ссылка на конфигурационный файл домена domU, то гостевая операционная система будет автоматически запускаться во время загрузки.

**Таблица 24.1. Компоненты платформы Xen**

| Путь | Назначение |
|---|---|
| **/etc/xen** | Каталог основной конфигурации |
| **xend-config.sxp** | Высокоуровневый файл конфигурации демона **xend** |
| **auto** | Конфигурационные файлы гостевой операционной системы для автоматического запуска во время загрузки |
| **scripts** | Вспомогательные сценарии, создающие сетевые интерфейсы и т.д. |
| **/var/log/xen** | Регистрационные файлы платформы Xen |
| **/usr/sbin/xend** | Главный управляющий демон платформы Xen |
| **/usr/sbin/xm** | Инструмент для управления гостевым доменом на платформе Xen |

Демон **xend** обеспечивает создание доменов domU и миграцию, а также решает другие задачи управления. Он должен всегда находиться в работе и, как правило, запускается во время загрузки. Его файл конфигурации **/etc/xen/xend-config.sxp** задает параметры соединения для гипервизора и ограничения ресурсов для домена dom0. Кроме того, он конфигурирует функциональные возможности для динамической миграции.

🕮 Разреженные файлы упоминаются также в разделе 10.4.

Диски гостевых доменов обычно хранятся в виртуальных блочных устройствах (virtual block devices — VBD) в домене dom0. Устройство VBD может быть соединено со специальным ресурсом, например физическим дисковым запоминающим устройством или логическим томом. Кроме того, оно может представлять собой закольцованный файл (loopback file), также известный как файловое устройство VBD (file-backed VBD), соз-

данное программой **dd**. Если используется специальный диск, то производительность такого устройства оказывается более высокой, но файлы являются более гибкими и могут управляться с помощью обычных команд системы Linux (таких, как **mv** или **cp**) в нулевом домене. Файлы, на основе которых создается такое устройство, являются разреженными и при необходимости могут увеличиваться.

Если в системе возникает узкое место с точки зрения производительности, то файловое устройство VBD обычно оказывается лучшим выбором. Если вы передумаете, устройство VBD легко переключить на специальный диск.

Аналогично виртуальные сетевые интерфейсы (VIF) можно установить многими способами. По умолчанию используется мостовой режим (bridged mode), в котором каждый гостевой домен является узлом той же сети, что и главный узел. Маршрутизированный режим и режим NAT конфигурируют гостевые домены как частные сети, доступные друг для друга и нулевого домена, но скрытые от остальной части сети. Более сложные конфигурации включают связанные сетевые интерфейсы и сети VLAN для гостей из разных сетей. Если ни одна из этих возможностей не удовлетворяет вашим требованиям, можно настроить сценарии платформы Xen для реализации практически любых пожеланий.

## Инсталляция гостя на платформе Xen с помощью программы virt-install

Для простой инсталляции гостя существует инструмент **virt-install**, являющийся частью приложения **virt-manager** для системы Red Hat.[3] Инструмент **virt-install** — это утилита, запускаемая из командной строки для инсталляции операционной системы. Она принимает средства инсталляции из разных источников, например устройства сетевой файловой системы NFS, физического CD- или DVD-привода или местоположения HTTP.

Например, инсталляция гостевого домена может быть выполнена с помощью следующей инструкции.

```
redhat$ sudo virt-install -n chef -f /vm/chef.img -l http://example.com/myos
  -r 512 --nographics
```

Это типичный гостевой домен на платформе Xen с именем "chef," дисковым устройством VBD, размещенным в каталоге **/vm/chef.img**, и средой инсталляции, полученной с помощью протокола HTTP. Этот экземпляр имеет 512 двоичных Мбайт (MiB) оперативной памяти и не использует во время инсталляции графическую поддержку X Windows. Утилита **virt-install** загружает файлы, необходимые для начала инсталляции, а затем запускает процесс инсталлятора.

Вы увидите пустой экран и пройдете через все стандартные этапы инсталляции системы Linux в текстовом режиме, включая конфигурирование сети и выбор пакета. После инсталляции гостевой домен перезагружается и готов к использованию. Для отсоединения от гостевой консоли и возвращения в домен dom0 выполните команду <Control-]>.

🕮 Более подробно система VNC обсуждается в разделе 30.2.

Несмотря на то что в данном примере волшебное заклинание **virt-install** осуществляет текстовую инсталляцию, возможна также графическая поддержка этого процесса с помощью системы Virtual Network Computing (VNC).

Конфигурация домена хранится в файле **/etc/xen/chef**. Этот файл выглядит следующим образом.

[3] Для поддержки инструмента ***virt-install*** в системе Ubuntu инсталлируйте пакет ***python-virtinst***.

```
name = "chef"
uuid = "a85e20f4-d11b-d4f7-1429-7339b1d0d051"
maxmem = 512
memory = 512
vcpus = 1
bootloader = "/usr/bin/pygrub"
on_poweroff = "destroy"
on_reboot = "restart"
on_crash = "restart"
vfb = [ ]
disk = [ "tap:aio:/vm/chef.dsk,xvda,w" ]
vif = [ "mac=00:16:3e:1e:57:79,bridge=xenbr0" ]
```

Как видим, по умолчанию сетевой интерфейс установлен в мостовой режим. В этом случае устройство VBD представляет собой "блочный" файл, обеспечивающий более высокую производительность, чем стандартный кольцевой файл. Файл образа диска, допускающий запись, предоставлен гостю под именем **/dev/xvda**. Это специфическое определение дискового устройства `tap:aio` рекомендовано разработчиками платформы Xen по соображениям производительности.

Инструмент **xm** удобен для ежедневного управления виртуальными машинами, например для их запуска и остановки, присоединения к их консолям и исследования текущего состояния. Ниже показана процедура запуска гостевого домена с последующим присоединением к консоли **chef**. Идентификаторы присвоены в порядке возрастания по мере создания гостевых доменов. При перезагрузке узла они восстанавливаются.

```
redhat$ sudo xm list
Name                ID   Mem(MiB)   VCPUs   State   Time(s)
Domain-0             0       2502       2   r-----   397.2
chef                19        512       1   -b----    12.8
redhat$ sudo xm console 19
```

Для того чтобы ввести в действие любую настройку гостевого домена, например присоединение другого диска или перевод сети из мостового режима в режим NAT, необходимо отредактировать гостевой конфигурационный файл в каталоге **/etc/xen** и перезагрузить гостя. Подробную информацию о дополнительных параметрах гостевых доменов можно найти на справочной странице `xmdomain.cfg`.

## Динамическая миграция на платформе Xen

Динамическая миграция — это процесс переноса домена domU с одного физического узла на другой без прекращения работы службы. С практической точки зрения это один из наиболее удобных и самых магических трюков виртуализации, который могут выполнить системные администраторы. Поскольку поддерживаются открытые сетевые соединения, то ни сеанс протокола SSH, ни одно активное соединение HTTP не будут потеряны. Хорошим поводом для динамической миграции является добавление аппаратного обеспечения, обновление операционной системы и перегрузка физических серверов.

Для успешного осуществления миграции важно, чтобы запоминающие устройства использовались совместно. Любое запоминающее устройство, необходимое для домена domU, например файл образа диска, на котором хранится виртуальная машина, должно быть доступным для обоих узловых серверов. Динамическую миграцию виртуальных машин с файловой поддержкой выполнить легче всего, потому что они обычно содержатся в одном переносимом файле. Однако совместное использование файлов между система-

ми возможно также в сетях SAN и NAS, а также в протоколах NFS и iSCSI. Несмотря на то что устройство VBD используется совместно несколькими пользователями, примите меры, чтобы домен domU единовременно функционировал только на одном физическом сервере. Файловые системы Linux не поддерживают непосредственный параллельный доступ к нескольким узлам.

Кроме того, поскольку IP- и MAC-адреса виртуальной машины следуют за ней от одного узла к другому, каждый сервер должен находиться на том же самом уровне 2 и в той же самой IP-подсети.

Как только виртуальная машина посылает трафик по сети, сетевое аппаратное обеспечение узнает новое местоположение MAC-адреса.

Если все эти основные требования удовлетворены, все, что нужно для выполнения миграции, — внести новые изменения в конфигурационный файл гипервизора **/etc/xen/xend-config.sxp**. Соответствующие параметры перечислены в табл. 24.2; в стандартном процессе инсталляции платформы все они закомментированы. Внеся изменения, запустите демон xend заново, выполнив команду **/etc/init.d/xend restart**.

**Таблица 24.2. Параметры динамической миграции в конфигурационном файле xend**

| Путь | Назначение |
|---|---|
| `xend-relocation-server` | Включает механизм миграции; установить равным `yes` |
| `xend-relocation-port` | Сетевой порт, используемый для выполнения миграции |
| `xend-relocation-address` | Интерфейс, прослушиваемый для миграционных соединений. Если не задан, платформа Xen прослушивает все интерфейсы в домене dom0 |
| `xend-relocation-hosts-allow` | Узлы, из которых разрешены соединения[a] |

[a] Этот параметр никогда нельзя оставлять пустым; в противном случае будут разрешены все соединения со всех узлов.

В процессе миграции виртуальной машины между узлами образ памяти домена domU передается по сети в незашифрованном виде. Если гость хранит в памяти конфиденциальные данные, администраторы обязаны обеспечивать секретность. Перед выполнением миграции файл конфигурации гостя должен находиться как на сервере источника, так и на сервере назначения. Если местоположение файлов образа диска на разных узлах разное (например, если один сервер монтирует совместно используемое запоминающее устройство в каталоге **/xen**, а другой — в каталоге **/vm**), то это различие должно отражаться в параметре `disk =` в файле конфигурации домена.

Сама миграция выполняется очень просто.

```
redhat$ sudo xm migrate --live chef server2.example.com
```

Предполагая, что гостевой домен chef функционирует, команда переносит его на другой узел платформы Xen, **server2.example.com**. Из-за того что флаг **--live** пропущен, он останавливает свою работу перед миграцией. Поучительно применить команду **ping** к IP-адресу домена chef во время миграции, чтобы увидеть отброшенные пакеты.

## Платформа KVM

KVM (Kernel-based Virtual Machine) — это инструмент полной виртуализации, включаемый в магистральное ядро системы Linux начиная с версии 2.6.20. Необходимым условием его применения является наличие расширений центрального процессора Intel

VT и AMD-V для поддержки виртуализации.[4] В операционной системе Ubuntu эта технология виртуализации предусмотрена по умолчанию, а компания Red Hat переключилась с платформы Xen на программу KVM после поглощения компании Qumranet, разработавшей программу KVM. Поскольку виртуализация KVM поддерживается аппаратным обеспечением центрального процессора, поддерживаются многие гостевые операционные системы, включая Windows. Для работы этого программного обеспечения необходима также модифицированная версия эмулятора процессора QEMU.

На платформе KVM ядро операционной системы Linux функционирует как гипервизор; управление памятью и диспетчеризация выполняются с помощью ядра узла, а гостевые машины представляют собой обычные процессы Linux. Этот уникальный подход к виртуализации сулит огромные преимущества. Например, сложность, порождаемая многоядерными процессорами, устраняется с помощью ядра системы, и для их поддержки не требуется вносить никаких изменений в гипервизор.

Команды Linux, такие как **top**, **ps** и **kill**, управляют виртуальными машинами так, будто они являются обычными процессами. Интеграция с системой Linux является безупречной.

Следует предупредить администраторов о том, что KVM — относительно новая технология, поэтому перед использованием ее следует тщательно протестировать. На сайте KVM зарегистрировано множество проблем, связанных с несовместимостью и возникающих при запуске гостей, представляющих собой разные типы операционных систем. Сообщается также о частом прерывании миграции между разными версиями программного обеспечения KVM. Об этом следует помнить.

## Инсталляция платформы KVM и ее использование

Несмотря на то что технологии, лежащие в основе платформ Xen и KVM, принципиально отличаются друг от друга, инструменты, инсталлирующие гостевые операционные системы и управляющие ими, похожи друг на друга. Как и на платформе Xen, для создания новых гостей по технологии KVM можно использовать программу **virt-install**, а для управления ими — команду **virsh**.[5] Для работы этих утилит необходима библиотека **libvirt**, разработанная компанией Red Hat.

Перед началом инсталляции необходимо сконфигурировать узел так, чтобы он поддерживал работу сетей в гостевых операционных системах.[6] В большинстве конфигураций для организации моста, обеспечивающего сетевую связность узлов в каждой гостевой системе, используется один физический интерфейс. В системе Red Hat конфигурационные файлы сетевого устройства хранятся в каталоге **/etc/sysconfig/network-scripts**. Требуется два файла устройства: для моста и физического устройства.

В приведенном ниже примере **peth0** — это физическое устройство, а **eth0** — мост.

**/etc/sysconfig/network-scripts/peth0**

```
DEVICE=peth0
```

[4] Имеет ли ваш центральный процессор эти расширения? Выполните следующую команду **egrep '(vmx|svm)' /proc/cpuinfo**. Если эта команда ничего не выведет на экран, значит, расширений нет. В некоторых операционных системах, для того чтобы увидеть расширения, необходимо включить их в настройках BIOS.

[5] При желании команду ***virsh*** можно использовать и для управления доменом domU на платформе Xen.

[6] Это в равной степени относится и к платформе Xen, но демон ***xend*** берет эту работу на себя, создавая интерфейсы в фоновом режиме.

```
ONBOOT=yes
BRIDGE=eth0
HWADDR=XX:XX:XX:XX:XX:XX
```

**/etc/sysconfig/network-scripts/eth0**

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
TYPE=Bridge
```

Здесь устройство **eth0** получает IP-адрес с помощью протокола DHCP.

Флаги, передаваемые программе **virt-install**, могут немного отличаться от флагов, используемых при инсталляции платформы Xen. Например, флаг **--hvm** означает, что для виртуализации гостя должно использоваться аппаратное обеспечение, а не паравиртуализация. Кроме того, аргумент **--connect** гарантирует, что по умолчанию будет выбран правильный гипервизор, потому что утилита **virt-install** поддерживает несколько гипервизоров. В заключение, чтобы получить выгоду от возможностей платформы KVM, рекомендуется использовать файл **-accelerate**. Следовательно, пример полной команды для инсталлирования гостевого сервера Ubuntu с диска CD-ROM должен выглядеть следующим образом.

```
ubuntu$ sudo virt-install --connect qemu:///system -n UbuntuHardy
    -r 512 -f ~/ubuntu-hardy.img -s 12 -c /dev/dvd --os-type linux
    --accelerate --hvm -vnc

Would you like to enable graphics support? (yes or no)
```

Если инсталляционный диск DVD для системы Ubuntu вставлен в дисковод, эта команда запускает процесс инсталляции и сохраняет гостя в файле **~/ubuntu-hardy.img**, позволяя увеличивать его размер до 12 Гбайт. Поскольку мы не указали ни флаг **-nographics**, ни флаг **--vnc**, утилита **virt-install** спрашивает, включать ли графику.

Утилита **virsh** запускает свою собственную оболочку, которая выполняет команды. Для того чтобы открыть эту оболочку, наберите команду **virsh --connect qemu:///system**. Следующая серия команд демонстрирует некоторые основные функциональные возможности утилиты **virsh**. Для того чтобы увидеть полный список этих функций, наберите команду **help** в оболочке или просмотрите справочную страницу.

```
ubuntu$ sudo virsh --connect qemu:///system
virsh # list --all
  Id Name State
----------------------------------------
3   UbuntuHardy running
7   Fedora running
-   Windows2003Server shut off

virsh # start Windows2003Server
Domain WindowsServer started

virsh # shutdown FedoraExample
Domain FedoraExample is being shutdown

virsh # quit
```

Похоже, что динамическая миграция на платформе KVM пока находится на стадии разработки; ее реализации в разных версиях резко отличаются друг от друга. Миграции между системами с разной архитектурой центрального процессора могут потребовать

специальных "заплат". Мы не рекомендуем полагаться на механизм динамической миграции в платформе KVM в промышленной среде, пока эта технология не станет достаточно устойчивой.

## 24.5. Зоны и контейнеры системы Solaris

Компания Sun раньше многих реализовала виртуализацию на уровне операционной системы, предложив в системе Solaris 10 (которая относится примерно к 2005 году) концепцию зон и контейнеров. Обширная интерактивная документация и сообщество активных пользователей привели к широкой адаптации и признанию этой технологии со стороны бизнеса. Гибкость и богатый набор инструментов для управления также способствуют продажам средств виртуализации для системы Solaris.

Зоны и контейнеры нельзя назвать средствами виртуализации, характерными исключительно для системы Solaris. Например, проект xVM включает в себя гипервизор LDOM для виртуальных машин, основанный на технологии Xen, наряду с другими мощными средствами управления для развертывания и контроля за большим количеством гостевых систем. Технология, основанная на аппаратном обеспечении компании Sun (наряду с системами многих других производителей), может выполнять физическое разделение устройств на электрическом уровне, позволяя нескольким операционным системам параллельно работать на одном и том же шасси. В этой книге мы не собираемся обсуждать дополнительные технологии, но читателям стоит заглянуть на сайты, посвященные многочисленным устройствам компании Sun.

Зоны и контейнеры отличаются от других инструментов виртуализации, поэтому мы начнем с быстрого обзора, который поможет нам не углубляться в некоторые темы.

Термины "зона" и "контейнер" во многом являются синонимами. В самом строгом смысле, зоны — это защищенная среда выполнения, а контейнер — это зона плюс управление ресурсами. На практике эти термины эквивалентны, и именно так мы будем их использовать в данной главе.

Все системы Solaris имеют "глобальную зону", которая запускает ядро и все процессы в системе, *включая* процессы из других зон. Неглобальная зона — это виртуальная система Solaris, работающая наряду с глобальной зоной. Сетевой трафик и выполнение процессов в неглобальной зоне скрыты от других неглобальных зон, но все процессы являются видимыми в глобальной зоне. С целью обеспечения безопасности важно ограничить доступ системных администраторов в глобальную зону.

Существует два типа зон: целостная (whole-root) и разреженная (sparse). Целостная зона содержит копии файлов собственной операционной системы и независимых файловых систем, требуя при этом намного больше места для хранения на запоминающих устройствах. Разреженная зона совместно с глобальной использует несколько своих файловых систем, монтируя их только для чтения.

Пулы ресурсов являются коллекциями системных ресурсов, таких как процессоры, которые можно распределять между зонами. Во всех системах существует по крайней мере один стандартный пул ресурсов. Все зоны, включая глобальную, должны иметь хотя бы один пул ресурсов. Можно также создавать несколько пулов ресурсов для распределения доступных системных ресурсов между функционирующими зонами.

Пул ресурсов состоит из, как минимум, одного набора ресурсов (который в настоящее время ограничен частью ресурсов центрального процессора) и алгоритма их распределения. Несколько зон могут совместно использовать один пул ресурсов. В этом случае

распределитель может определять, как совместно использовать центральный процессор среди зон, использующих пул ресурсов.

Зоны поддерживают множество разных алгоритмов распределения ресурсов для разных ситуаций, но мы сосредоточимся на самом популярном — "справедливом разделе" ("fair share scheduling").

Рассмотрим конкретный пример. Представьте себе систему с двумя физическими центральными процессорами. Эта система собирается запустить две виртуальные системы Solaris: одну со специализированным приложением, требующим по крайней мере одного полноценного центрального процессора, а другую — с облегченным веб-сервером, не требующим особенных ресурсов. Реализация этой архитектуры в системе Solaris представлена на рис. 24.4.

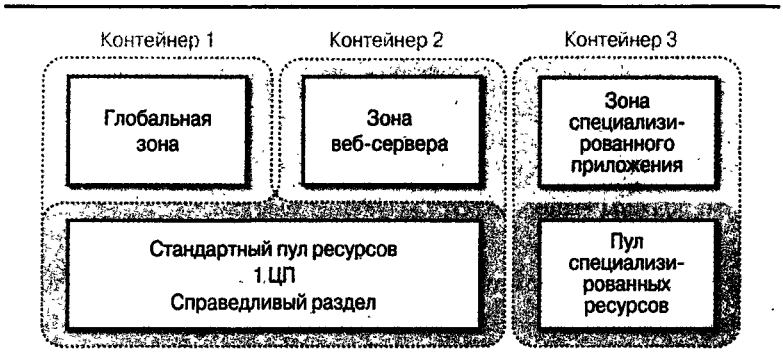



*Рис. 24.4. Пример контейнеров в системе Solaris*

На рис. 24.4 показаны три контейнера: контейнер для специализированного приложения, веб-сервер и оригинальный (глобальный) экземпляр системы Solaris. Кроме того, существует три зоны: для лицензионного приложения, веб-сервера и глобальная. Помимо них, есть два пула ресурсов: по одному для каждого центрального процессора.

Веб-сервер и глобальная зона совместно используют стандартный пул ресурсов, содержащий один центральный процессор и реализующий алгоритм справедливого раздела. Каждая из этих зон имеет свою долю (которые не изображены на рисунке), т.е. ресурсы центрального процессора разделяются поровну между глобальной зоной и зоной веб-сервера. Специализированное приложение использует отдельный пул ресурсов со специальным центральным процессором.

Система Solaris имеет несколько утилит командной строки для управления контейнерами, зонами и ресурсами. Наиболее важно то, что каждая зона и пул ресурсов имеют инструмент конфигурации и инструмент управления: **zonecfg**, **zoneadm**, **poolcfg** и **pooladm**.

С помощью команды **pooladm** можно создать пул ресурсов до его присвоения зоне. Для включения пулов используется флаг **-e**, а затем выполняется команда **pooladm** без аргументов, чтобы увидеть текущее состояние пула.

```
solaris$ sudo pooladm -e
solaris$ sudo pooladm

system default
         string    system.comment
```

```
int       system.version 1
boolean   system.bind-default true
string    system.poold.objectives wt-load
...
```

Мы сократили довольно длинный вывод; команда продолжает печатать информацию о текущем состоянии пула и доступности центрального процессора. Стандартный пул называется pool_default. Он содержит все доступные ресурсы центрального процессора.

Команда **poolcfg** создает новый пул. В серии команд, представленных ниже, мы выделяем отдельный центральный процессор для набора специализированных ресурсов, присваиваем этот набор ресурсов новому пулу ресурсов и активизируем процессор.

```
solaris$ sudo poolcfg -c 'create pset proprietary-pset (uint pset.min=1;
   uint pset.max=1)'
solaris$ sudo poolcfg -c 'create pool proprietary-pool'
solaris$ sudo pooladm -c
```

Теперь мы создаем зону с помощью команд **zoneadm** и **zonecfg**. Выполнение команды **zonecfg** открывает новую оболочку конфигурации для зоны. Следовательно, этот инструмент поддерживает оболочечные функциональные свойства, такие как автозаполнение при нажатии клавиши <Tab> и "горячие" клавиши для перемещения курсора.

Как минимум, должны выполняться следующие условия.

- Зона создана.
- Для зоны задан путь к запоминающему устройству для зонных файлов и файловых систем.
- Для зоны задан независимый IP-адрес.
- Зоне присвоен один из активных системных интерфейсов.
- Зоне выделен созданный выше пул ресурсов.

Рассмотрим команды, выполняющие эти действия.

```
solaris$ sudo zonecfg -z proprietary-zone
zonecfg:proprietary-zone> create
zonecfg:proprietary-zone> set zonepath=/zones/proprietary-zone
zonecfg:proprietary-zone> set autoboot=true
zonecfg:proprietary-zone> add net
zonecfg:proprietary-zone:net> set address=192.168.10.123
zonecfg:proprietary-zone:net> set physical=e1000g0
zonecfg:proprietary-zone:net> end
zonecfg:proprietary-zone> set pool=proprietary-pool
zonecfg:proprietary-zone> verify
zonecfg:proprietary-zone> commit
zonecfg:proprietary-zone> exit
```

Обратите внимание на то, что команда **zonecfg** просит вас указать объект, с которым вы в настоящее время работаете. В этот момент происходит конфигурирование зоны, но на самом деле она еще не инсталлирована и не готова к работе. Это полноценная система Solaris, которой нужен пакет инсталляции, как любой нормальной операционной системе. Утилита **zoneadm** инсталлирует, загружает и выполняет другие операции над зоной.

```
solaris$ sudo zoneadm -z proprietary-zone install
Preparing to install zone <proprietary-zone>.
Creating list of files to copy from the global zone.
...
```

```
solaris$ sudo zoneadm -z proprietary-zone boot
solaris$ sudo zoneadm list
global
proprietary-zone
```

После запуска зоны вызов команды **zlogin -C proprietary-zone** приводит к соединению с ее консолью. Процесс загрузки зоны напоминает загрузку физической системы, поэтому соединение с утилитой **zlogin** перед загрузкой приводит к выводу на экран всей информации во время процесса загрузки. Необходимо выполнить обычное конфигурирование новой системы, например выбрать язык и метод аутентификации.

Осталось создать зону для нашего веб-сервера. Поскольку зона веб-сервера делит стандартный пул ресурсов с глобальной зоной, нет необходимости создавать новый пул. Вместо этого мы просто создадим зону с помощью утилиты **zonecfg**, как показано выше, но с параметром **pool=pool_default**.

Концепция зон и контейнеров намного глубже, чем мы описали в этом разделе. Она предусматривает функциональные возможности для миграции зон между физическими системами (хотя динамическая миграция не поддерживается), ZFS-ресурсы для небольших целостных зон и "фирменные зоны" ("branded zones"), поддерживающие выполнение бинарных модулей, принадлежащих другим платформам (например, Linux), в ядре системы Solaris.

## 24.6. Разделы рабочей нагрузки в системе AIX

Компания IBM уже давно занимается виртуализацией. Ее сотрудники изобрели эту концепцию еще в 1960-х годах, но не реализовали ее, пока в конце 2007 года не появилась версия AIX 6.1. Любая система, способная выполнить программную платформу AIX 6, поддерживает разделы рабочей нагрузки (WPAR). (Эта технология отличается от различных реализаций логического разделения, предлагавшихся компанией IBM в версиях, предшествующих версии 4.3.)

Разделы WPAR работают в изолированной среде выполнения. Процессы могут взаимодействовать только с процессами, находящимися в том же самом разделе. Сигналы и события в глобальной среде не влияют на разделы и наоборот. Разделы WPAR могут иметь собственные сетевые адреса.

Разделы WPAR бывают двух видов: системные и прикладные.

- Системный раздел WPAR совместно с глобальным окружением использует только ядро системы AIX (по существу, узел). Приложение, работающее в системе WPAR, функционирует так, будто оно запущено в независимом экземпляре системы AIX с собственным демоном **inetd** для обеспечения сетевой автономии.
- Прикладной раздел WPAR выполняет отдельное приложение в изолированной среде. Он использует все файловые системы совместно с глобальным окружением и не может предоставить возможности для удаленного доступа. После выполнения приложения прикладной раздел WPAR продолжает существовать.

Компания IBM предоставляет несколько разных инструментов для управления разделами WPAR, продолжая традицию обеспечения удобного управления, характерную для системы. Здесь мы обсудим интерфейс командной строки, но администраторы должны знать, что в типичной ситуации для системы AIX существует полноценный интерфейс SMIT и программа WPAR Manager, представляющая собой веб-интерфейс для централизованного управления несколькими серверами и их разделами рабочей нагрузки.

Мы создаем систему WPAR с помощью команды **mkwpar**. Для нее нужен только один аргумент **-n**, чтобы задать имя системы. Таким образом, команда **mkwpar -n mario** создает раздел с именем "mario." Команда **mkwpar** создает файловые системы, инсталлирует соответствующие наборы файлов и готовит подсистемы и службы. Этот процесс занимает мало времени; после его завершения для запуска системы WPAR необходимо выполнить команду **startwpar mario**.

Каталог **/wpars** в глобальной среде содержит файловые системы для раздела **mario**. Разделы WPAR из глобальной среды можно перечислить с помощью команды **lswpar**.

```
aix$ sudo lswpar
Name   State   Type   Hostname   Directory
-------------------------------------------------------
mario  A       S      mario      /wpars/mario
```

Для того чтобы соединить корень с консолью, используется команда **clogin mario**. Что может быть проще?

Этот простой пример не учитывает множество важных фактов и возможностей для настройки. Например, новое программное обеспечение нельзя инсталлировать в разделе mario, потому что файловые системы **/usr** и **/opt** по умолчанию монтируются только для чтения, чтобы сэкономить память запоминающего устройства. Кроме того, не доступен ни один сетевой интерфейс. Упрощенная процедура, описанная выше, также игнорирует превосходные возможности, которые компания IBM предоставляет для управления ресурсами и которые упрощают контроль над использованием центрального процессора, памяти, виртуальной памяти и других ресурсов разделом WPAR.

Для создания более полезного экземпляра можно снабдить команду **mkwpar** дополнительными аргументами. Приведенная ниже процедура создает раздел WPAR со следующими атрибутами.

- Имя WPAR — mario (**-n**).
- Настройки для разрешения имен наследуются у глобального экземпляра (-r).
- Созданы частные, допускающие запись файловые системы **/opt** и **/usr** (**-l**).
- Раздел WPAR использует IP-адрес 192.168.10.15 на интерфейсе en0 из глобального раздела WPAR (-N).
- Доля центрального процессора, выделенного данному разделу WPAR, колеблется от 5 (минимум) до 15% (условный максимум); абсолютный предел составляет 25% (**-R**).

```
aix$ sudo mkwpar -n mario -r -l -N interface=en3 address=192.168.10.15
  netmask=255.255.255.0 broadcast=192.168.10.255 -R active=yes
  CPU=5%-15%,25%
```

Этот вызов более подробный по сравнению с предыдущим вызовов команды **mkwpar** и выполняется дольше из-за дублирования файловых систем **/usr** и **/opt**.

Для модификации раздела после его создания следует использовать команду **chwpar**. С помощью команды **stopwpar** можно прекратить работу раздела, а с помощью команды **rmwpar** — удалить его.

Прикладные разделы WPAR обрабатываются аналогично. Вместо команды **mkwpar** для создания прикладного раздела WPAR используется команда **wparexec**. Ее параметры в целом идентичны параметрам команды **mkwpar**, но вместо имени следует указать приложение, которое выполняется как аргумент. Например, для того чтобы запустить веб-сервер Apache в своем прикладном разделе WPAR, следует просто выполнить команду **wparexec /usr/local/sbin/httpd**.

# 24.7. Программное обеспечение Integrity Virtual Machines в системе HP-UX

Если целью сотрудников компании HP было запутать администраторов неорганизованной массой возможных подходов к виртуализации, то они ее достигли. Программное обеспечение Integrity Virtual Machines распределяет ресурсы аппаратного обеспечения среди нескольких гостевых операционных систем на отдельном низкоуровневом сервере. Для более крупных систем с несколькими ядрами компания HP разработала более мощное программное обеспечение Virtual Partitions или vPars. Кроме того, служба разделения аппаратного обеспечения nPartitions обеспечивает истинное разделение ресурсов среди работающих серверов на электрическом уровне. Эти технологии совместно с программным обеспечением компании HP для кластеризации получили обобщенное название Virtual Server Environment.

В этом разделе рассматривается только программное обеспечение Integrity Virtual Machines (IVM). IVM — это технология полной виртуализации. Немодифицированные версии систем Windows и Linux могут выполнять его на узле HP-UX. Каждый гость получает заранее сконфигурированную долю времени центрального процессора. Память и ресурсы запоминающих устройств также допускают настройку. Можно конфигурировать неограниченное количество гостей, но количество функционирующих виртуальных машин ограничено числом 256.

Сеть для гостевых машин состоит из трех компонентов.

- Адаптер физической сети в узле, который называется pNIC.
- Адаптер гостевой сети, который называется виртуальным интерфейсом NIC или vNIC.
- Виртуальный коммутатор, или vswitch, создающий сеть между узлом и одним или несколькими гостями.

Сложность различных сетевых конфигураций, возникающих при инсталляции виртуальных машин Integrity, ошеломляет, но в то же время это обеспечивает гибкость системы. Для иллюстрации создадим отдельный виртуальный коммутатор, выполняющий отображение в ту же сеть, в которой находится узел. Это простейшая конфигурация, которая эквивалентна мостовым сетям, упомянутым ранее.

Почти как на платформе Хеп, узел может предоставлять гостям ресурсы запоминающего устройства в виде физических дисков, DVD, лентопротяжных устройств или файлов операционной системы узла. Кроме того, как и на платформе Xen, достичь максимальной производительности работы запоминающего устройства иногда довольно сложно.

Инструкции по сложной конфигурации сети и методам оптимизации запоминающих устройств можно найти в руководстве *Installation, Configuration, and Administration Guide* для системы HP-UX.

## Создание и инсталляция виртуальных машин

Команды, создающие виртуальные машины, инсталлирующие их и управляющие ими, являются мощными и в то же время простыми. Имя каждой команды начинается с префикса **hpvm**, а остальная часть имени описывает требуемое действие. Например, команда для создания новой виртуальной машины называется **hpvmcreate**.

Для управления конфигурацией гостевых операционных систем в каждой команде используются различные аргументы и нет статических файлов, которыми мог бы управлять администратор. Изменения в настройке гостя осуществляются с помощью команды **hpvmmodify**.

Прежде чем создать виртуальную машину, следует обеспечить доступ к ресурсам ее запоминающих устройств и сети. В приведенном ниже примере сначала мы создаем файловую систему на одном из физических дисков для хранения гостя, а затем виртуальный коммутатор для обеспечения сетевой связности гостя. Кратко опишем процесс создания виртуальной машины.

- Создаем ресурсы запоминающих устройств для файлов гостя.
- Создаем виртуальный коммутатор для сетевой связности.
- Создаем виртуальную машину.
- Запускаем и инсталлируем виртуальную машину.

В серии приведенных ниже команд мы используем команду **mkfs** для создания файловой системы на физическом устройстве **disk3**, произвольном диске, который оказался доступным в нашей лабораторной системе. В результате он будет хранить гостевую операционную систему, приложения и данные. Файл монтируется в каталоге **/vdev**, и, в заключение, команда **hpvmdevmgmt** (не путайте ее с командой **hpwndkwlvyfm**) создает запоминающее устройство с именем **disk1**.

```
hp-ux$ sudo mkfs -F vxfs -o largefiles /dev/disk/disk3
hp-ux$ sudo mount /dev/disk/disk3 /vdev/vm0disk/
hp-ux$ sudo hpvmdevmgmt -S 8G /vdev/vm0disk/disk1
```

На следующем шаге создается виртуальный коммутатор для гостя. Один виртуальный коммутатор может использоваться несколькими гостями.

```
hp-ux$ lanscan
Hardware    Station        Crd  Hdw   Net-Interface  NM  MAC    HP-DLPI   DLPI
Path        Address        In#  State NamePPA        ID  Type   Support   Mjr#
0/0/3/0     0x00306EEA9237 0    UP    lan0 snap0     1   ETHER  Yes       119
0/1/2/0     0x00306EEA720D 1    UP    lan1 snap1     2   ETHER  Yes       119
hp-ux$ sudo hpvmnet -c -S vm0switch -n 0
hp-ux$ sudo hpvmnet -b -S vm0switch
```

Здесь удобная команда **lanscan** находит все системные сетевые интерфейсы. Нам необходим правильный идентификатор для сетевого интерфейса в правильной сети; в данном случае это идентификатор **0** в сети **lan0**. Используя команду **hpvmnet** и имя **lan0**, мы создаем коммутатор, а затем запускаем его с помощью следующей команды, передавая ей аргумент **-b**.

Создав необходимые ресурсы, можно приступать к созданию самой виртуальной машины.

```
hp-ux$ sudo hpvmcreate -P vm0 -O hpux -r 2G
   -a network:lan:vswitch:vm0switch -a disk:scsi::file:/vdev/vm0disk/disk1
hp-ux$ sudo hpvmstart -P vm0
```

Эта команда создает виртуальную машину с именем vm0, выделяет два двоичных гигабайта памяти, использует виртуальный коммутатор vm0switch и выбирает запоминающее устройство, созданное нами ранее. Затем новая виртуальная машина запускается с помощью команды **hpvmstart**.

Инсталляция виртуальной машины идентична инсталляции физической машины с консоли. Соединение с консолью осуществляется с помощью команды **hpvmconsole**

`-P vm0`, а отсоединение — с помощью комбинации клавиш <Ctrl+B>. Для проверки статуса гостя используется команда **`hpvmstatus -P vm0`**.

## 24.8. VMware: операционная система со своими собственными правами

Компания VMware — крупнейший игрок на авангардном рынке средств виртуализации и первый разработчик методов виртуализации капризной платформы x86. Она разработала технологию для выполнения семнадцати проблематичных инструкций, которые ранее препятствовали распространению виртуализации. Выпуск программного обеспечения VMware Workstation в 1999 году открыл возможности для более эффективных вычислений, реализация которых остается актуальной проблемой и в наши дни.

Платформа VMware — это третий коммерческий продукт, заслуживающий внимания при выборе технологии виртуализации для сайта. Администраторов систем UNIX и Linux в первую очередь должны заинтересовать платформы ESX и ESXi — автономные гипервизоры для архитектуры Intel x86. Платформа ESXi распространяется свободно, но некоторые полезные функциональные возможности, такие как доступ к консоли, из нее удалены. Платформа ESX предназначена для предприятий, использующих инсталляции, управляемые сценариями, средства для слежения с помощью протокола SNMP и поддержку загрузки с устройства сети для хранения данных (SAN).

Кроме продуктов ESX, компания VMware выпустила несколько мощных и современных программ, облегчающих централизованное развертывание и управление виртуальными машинами.

Разработчики компании VMware создали самую зрелую технологию динамической миграции, которую нам довелось видеть. К сожалению, их интерфейс управления клиентом в данный момент работает только в среде Windows. В общем, продукция компании VMware относится к следующему поколению информационных сред, подробное описание которых выходит за рамки нашей книги.

Системы HP-UX и AIX не могут работать как виртуальные машины VMware, поскольку они используют специализированную архитектуру процессора, которую программное обеспечение VMware не эмулирует.

Разумеется, система Linux поддерживается хорошо. Система Solaris также может функционировать под управлением программы VMware, поскольку ее исходный код охватывает платформы SPARC и x86.

## 24.9. Веб-службы компании Amazon

Все крутые парни занимаются облачными вычислениями, и компания Amazon Web Services (AWS) впереди всех. Начиная с 2006 года компания Amazon начала выпускать на рынок инфраструктуру, в основе которой лежит сайт `amazon.com`, продавая доступ к набору прикладных интерфейсов и веб-служб. Эти службы представляют собой чрезвычайно мощную, масштабируемую и широко доступную вычислительную платформу для всех, кому необходимы дешевые и быстродействующие вычислительные мощности или ресурсы.

Основной пакет услуг, предлагаемых компанией AWS администраторам системы UNIX, включает в себя следующие службы.

- EC2, Elastic Compute Cloud, — платформа для масштабируемых вычислений. "Экземпляр" платформы EC2 представляет собой расположенный в Интернете сервер, который запускает операционную систему по вашему выбору и находится под вашим полным контролем. Вы можете добавлять или удалять экземпляры по своему желанию. Платформа EC2 основывается на платформе Xen и поддерживает много разных операционных систем.
- EBS, Elastic Block Store, — постоянное дископодобное устройство для экземпляров EC2. Концепция EBS похожа на хранилище SAN. Это устройство позволяет сохранять экземпляры платформы EC2 и сберегать ее состояние между разными вызовами.
- S3, Simple Storage Services, — широко доступная долгосрочная инфраструктура хранения. В отличие от платформы EBS, она не предназначена для монтирования в качестве файловой системы, а вместо этого хранит и извлекает объекты с помощью прикладного программного интерфейса.

Эти службы обеспечивают администраторам беспрецедентную гибкость и масштабируемость за счет потери части контроля над аппаратным обеспечением и сетевой конфигурацией.

Перенося службы в облако, администраторы и разработчики должны учитывать последствия для безопасности. Конфиденциальные данные должны оставаться в физическим центре обработки данных, особенно если его деятельность регулируется законами, такими как Sarbanes-Oxley или HIPAA (в США). Требования регуляторов могут исключать (но могут и не исключать!) использование облачных вычислений, но пока суды заполняют пробелы в законодательстве, лучше не играть с огнем.

Где же тогда служба AWS может оказаться полезной? С точки зрения стоимости, доступности и динамической масштабируемости, служба AWS находится вне конкуренции как платформа для веб-хостинга.

Стоимость использования экземпляров EC2 "до востребования" в настоящее время колеблется от 0,09 до 0,68 долл. за час, в зависимости от вычислительной мощности экземпляра. Стоимость использования хранилища S3 составляет 0,15 долл. за гигабайт в месяц. Эти гроши складываются в приличную сумму (работа самого дешевого из доступных экземпляров платформы EC2 будет стоить около 379 долл. в год в рамках трехлетнего контракта), но если учесть стоимость электроэнергии, систем охлаждения и эксплуатационные издержки, то итоговая сумма покажется более привлекательной, чем расходы на содержание собственных серверов.

С учетом неограниченных ресурсов, облако также является привлекательным в качестве распределенной вычислительной платформы. Фактически служба AWS может с пользой использоваться для хостинга электронной почты, DNS или других услуг, которые обычно предоставляются центрами обработки данных.

Служба AWS — это ядро набора интерфейсов прикладного программирования SOAP, но компания Amazon предоставляет несколько простых оболочек командной строки, написанных на языке Java, а также графический пользовательский веб-интерфейс и надстройки для браузера Firefox. Служба AWS может запускать в качестве гостевых операционных систем как систему Linux, так и систему Windows.

Ниже перечислены этапы создания и запуска постоянного запоминающего устройства.

- Инсталляция и конфигурирование среды выполнения на языке Java (проследите, чтобы переменная окружения JAVA_HOME ссылалась на правильное место).
- Создание учетных записей S3 и EC2 на веб-сайте Amazon Web Services.

- Загрузка и инсталляция инструментов для платформы EC2.
- Создание экземпляра EC2 из образа диска Amazon Machine Image (AMI), представляющего собой образ диска сконфигурированной операционной системы, возможно, с инсталлированным дополнительным программным обеспечением. Вы можете выбрать эти программы из обширного списка доступного программного обеспечения или развернуть свое собственное.
- Создание тома EBS и присоединение его к вашему экземпляру.

Веб-сайт AWS содержит страницы для регистрации учетных записей и все необходимые модули для загрузки. Для того чтобы приступить к использованию службы AWS, загрузите инструменты командной строки и идентификаторы доступа, которые состоят из сертификата и закрытого ключа.

Создайте каталог с именем **~/.ec2** и перенесите туда загруженный файл сертификата, файл ключа и загруженные инструменты. Все команды EC2 будут находиться в каталоге **~/.ec2/bin**, а библиотечные модули — в каталоге **~/.ec2/lib**. Для того чтобы настроить окружение для более легкого использования, добавьте в сценарий регистрации следующие инструкции. (Например, для пользователя bash файл будет называться **~/.bash_login** или **~/.bash_profile**.)

```
export EC2_HOME=~/.ec2
export PATH=$PATH:$EC2_HOME/bin
export EC2_PRIVATE_KEY=$EC2_HOME/pk-<long string value>.pem
export EC2_CERT=$EC2_HOME/cert-<long string value>.pem
export JAVA_HOME=/path/to/java
```

В заключение, прежде чем сделать выбор образа и загрузить его, создайте пару ключей, которые вы собираетесь использовать для обеспечения доступа к этому образу. Новую пару ключей создает команда **ec2-add-keypair**. Все новые образы, которые вы создадите, будут автоматически конфигурироваться для использования нового открытого ключа для аутентификации SSH на корневой учетной записи.

```
ubuntu$ ec2-add-keypair my-keypair
KEYPAIR my-keypair b0:65:11:df:05:43:3b:f7:42:93:fb:0e:7f:63:22:13:ff:88:e5:ae
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAoiJxHIHjuXOqeEoKaeluj8ny55INuWS5hOQVBxfuhEwG7kttz
kiuF8B7U4C4
...
82827HZO/9cCok6FP8loOAR8GIJvDzvWozZ7hdRhc/i6isWBiMTDQQUItk79fI9atk7P
-----END RSA PRIVATE KEY-----
```

Этот ключ понадобится в будущем, поэтому сохраните все, кроме строки, начинающейся словом KEYPAIR, в файл, находящийся в каталоге **~/.ec2**, и установите параметр режима доступа равным 600. Никогда не допускайте совместное использование файла, содержащего закрытый ключ, ведь именно в нем содержатся ключи от облаков!

Настало время выбрать образ AMI. Существует огромный список образов AMI, одни из них созданы компанией Amazon, а другие предоставлены (или проданы) членами сообщества. Можно также самостоятельно создать образ AMI для собственного использования и сконфигурировать его с учетом особенностей вашего окружения или инсталлировать в него заранее все необходимые приложения.

Выбрав образ, обратите внимание на его идентификатор. Команда, приведенная ниже, выводит на экран список образов AMI, созданных компанией Amazon. Добавив в переменную PATH имя каталога с инструментами, можно выполнять команды платформы EC2 где угодно.

```
ubuntu$ ec2-describe-images -o amazon
IMAGE ami-ba4eaad3 /aws-quickstart/phpquickstart.manifest.xml
  amazon available public i386 machine
IMAGE ami-b44bafdd /aws-quickstart/rubyquickstart.manifest.xml
  amazon available public i386 machine aki-a71cf9ce ari-a51cf9cc
IMAGE ami-1c54b075 /aws-quickstart/tomcatquickstart.manifest.xml
  amazon available public i386 machine aki-a71cf9ce ari-a51cf9cc
...
```

Имя образа обычно имеет краткое описание, помогающее понять его назначение и конфигурацию, но подробная информация находится на сайте. Для того чтобы препроцессор PHP быстро запустил образ AMI из приведенного выше списка, выполните следующую команду.

```
ubuntu$ ec2-run-instances ami-ba4eaad3 -k my-keypair
ubuntu$ ec2-describe-instances
RESERVATION  r-56aa053f  default
INSTANCE i-1343fb7a  ami-ba4eaad3  pending  my-keypair  0
  m1.small  2008-12-22T01:43:27+0000  us-east-1c
```

Результаты работы команды **ec2-describe-instances** отражают тот факт, что экземпляр еще загружается (статус "pending") и что он использует пару ключей с именем **my-keypair**.

Следует подчеркнуть, что, согласно этим результатам, экземпляр работает в зоне доступности us-east-1c. Компания Amazon разделила свои системы на отдельные зоны доступности, поэтому пользователь, желающий получить гарантии, что несколько экземпляров будут выполняться в физически разделенных центрах обработки данных, может послать запрос в командной строке. Это значение также необходимо для присоединения тома EBS.

В заключение отметим, что, согласно данной команде, экземпляр имеет тип m1.small. Этот код сообщает объем ресурсов, доступных для системы. Служба Amazon имеет несколько стандартных профилей; m1.small — это стандартный профиль, включающий единственный 32-битовый центральный процессор EC2, память объемом 1,7 Гбайт и 160 Гбайт памяти на диске (непостоянном). Разумеется, реальное аппаратное обеспечение, на котором работает ваш сервер, вряд ли имеет такие параметры; это просто способ описать ваши ресурсы.

Запустив экземпляр платформы, необходимо провести авторизацию всех сетевых портов, к которым вам нужен доступ. Для этого используется команда EC2 **ec2-authorize**.

```
ubuntu$ ec2-authorize default -p 22
```

В данном случае порт 22 (для протокола SSH) авторизован для всех узлов в стандартной группе. (Группы — это механизм для управления коллекциями экземпляров. Если не указано иное, новые экземпляры включаются в стандартную группу.) После авторизации порта 22 можно, наконец-то, соединиться с вашим новым экземпляром. Для этого сначала найдите имя узла, затем соединитесь с ним с помощью утилиты **ssh**, как с обычным узлом в Интернете (потому что он таким и является!).

```
ubuntu$ ec2-describe-instances
RESERVATION   r-56aa053f   default
INSTANCE   i-1343fb7a   ami-ba4eaad3   ec2-67-202-24-235.compute-
   1.amazonaws.com   domU-12-31-39-02-5E-55.compute-1.internal
   runninmy-keypair   0   m1.small   2008-12-22T01:43:27+0000
   us-east-1c
```

```
ubuntu$ ssh -i ~/.ec2/id_rsa-my-keypair
   root@ec2-67-202-24-235.compute-1.amazonaws.com
```

Эта команда использует пару ключей, сохраненную ранее, для соединения с корневой учетной записью нового экземпляра. Для обеспечения безопасности мы рекомендуем отключить корневой доступ по протоколу SSH.

Осталась только одна проблема — экземпляры не являются персистентными. Иначе говоря, когда экземпляр прекращает работу, любые изменения состояния его дисков или памяти не сохраняются. Тома запоминающих устройств для новых экземпляров предоставляет устройство EBS. В данном случае мы создаем том и присоединяем его к работающему узлу.

```
ubuntu$ ec2-create-volume -s 1 -z us-east-1c
VOLUME   vol-5de80c34   1   us-east-1c   creating   2008-12-22T02:02:
   53+0000
ubuntu$ ec2-attach-volume vol-5de80c34 -i i-1343fb7a -d /dev/sdf
   ATTACHMENT vol-5de80c34   i-1343fb7a   /dev/sdf   attaching 2008-
     12-22T02:04:01+0000
ubuntu$ ec2-describe-volumes
VOLUME    vol-5de80c34   1   us-east-1c   in-use   2008-12-22T02:02:
   53+0000
ATTACHMENT   vol-5de80c34   i-1343fb7a   /dev/sdf   attached 2008-12-
   22T02:04:01+0000
```

🕮 Более подробно создание файловых систем описано в главе 8.

Эти команды создают новый том запоминающего устройства EBS в зоне доступности us-east-1c и присоединяют его как устройство **/dev/sdf** к ранее созданному экземпляру. Для того чтобы создать файловую систему и приступить к использованию этого тома, зарегистрируйте его в экземпляре платформы и работайте так, будто вы на самом деле создали файловую систему на физическом диске.

Помните, что служба AWS взимает почасовую плату за работу каждого экземпляра, поэтому неиспользуемые экземпляры и тома следует удалять. Прежде чем завершить работу ненужного экземпляра, отсоедините от него все тома EBS и аккуратно остановите работу системы, как это обычно делают с физически существующими узлами.

```
ubuntu$ ec2-detach-volume vol-5de80c34
ATTACHMENT   vol-5de80c34   i-1343fb7a   /dev/sdf   detaching 2008-
   12-22T02:04:01+0000
ubuntu$ ec2-terminate-instances i-1343fb7a
INSTANCE   i-1343fb7a   running   shutting-down
ubuntu$ ec2-delete-volume vol-5de80c34
VOLUME vol-5de80c34
```

В дополнение к интерфейсу командной строки, компания Amazon предлагает управляющий веб-интерфейс, который может инициировать те же самые команды, что и утилиты командной строки, например запуск и останов экземпляров, присоединение томов запоминающих устройств и выделение IP-адресов. Надстройка ElasticFox для веб-браузера Firefox обеспечивает те же функциональные возможности непосредственно в самом браузере.

Компания Amazon регулярно предлагает новые функциональные возможности и продукты для своих веб-служб. Например, автоматическое масштабирование экземпляров EC2 предусматривает автоматический запуск новых серверов, чтобы предотвратить их отключение при высокой нагрузке. Служба CloudWatch проводит мониторинг показате-

лей, таких как степень использования центрального процессора и дисковых механизмов ввода-вывода, чтобы быстро реагировать на изменяющиеся условия. Обратите внимание на блог службы AWS по адресу: `aws.typepad.com`. Он содержит информацию о новых функциональных возможностях и усовершенствованиях.

## 24.10. Рекомендуемая литература

- Веб-сайт `virtualization.info` — превосходный источник новостей и слухов о виртуализации и облачных вычислениях.
- Troy Ryan. *VMware Cookbook: A Real-World Guide to Effective VMware Use.* Sebastopol, CA: O'Reilly Media, 2009.
- Crawford, Luke. *The Book of Xen: A Practical Guide for the System Administrator.* San Francisco, CA: No Starch Press, 2009.
- Hess, Kenneth. *Practical Virtualization Solutions: Virtualization from the Trenches.* Upper Saddle River, NJ: Prentice Hall PTR, 2009.

## 24.11. Упражнения

24.1. Сравните разные подходы к визуализации. К какой категории относится платформа KVM? Чем технология облачных вычислений отличается от виртуализации?

24.2. Современные процессоры Intel и AMD имеют специальные инструкции, улучшающие поддержку виртуализации. Что это за инструкции и какие специальные функции они выполняют? Выберите конкретную операционную систему и модель процессора и опишите, по крайней мере, два способа, позволяющих определить, поддерживаются ли инструкции виртуализации.

24.3. Какие новые функциональные возможности стала поддерживать служба Amazon Web Services с момента написания этой главы? Улучшают ли они существующую инфраструктуру или представляют собой совершенно новые службы?

24.4. ✯ Создайте учетную запись на службе Amazon Web Services и пару открытых ключей. Установите окружение Java и создайте экземпляр платформы EC2. Имеете ли вы прямой доступ к консоли? Что из этого следует? Предположим, что созданный вами экземпляр платформы предназначен для хранения конфиденциальных данных, какие шаги следует предпринять, чтобы убедить клиента в том, что данные в облаке хорошо защищены?

24.5. ✯ Крупное предприятие планирует развернуть новую систему для управления взаимоотношениями с клиентами (CRM), состоящую из зарезервированных прикладного и интерфейсного серверов, промежуточных серверов и сервера для управления базой данных. Какие из этих компонентов системы CRM следует виртуализировать? Объясните свое решение.

# Глава 25

# Система X Window System

Основой для большинства графических пользовательских сред для систем UNIX и Linux является система X Window (ее еще называют X11 или просто X). X — это прямая наследница оконной системы, называемой *W* (это действительно так), которая разрабатывалась под эгидой проекта Project Athena в Массачусетсском технологическом институте (MIT) в начале 80-х годов прошлого века. Система X Window System версии 10, вышедшая в 1985 году, была первой, которой удалось достичь широкого распространения, а вскоре после нее вышла версия 11 (X11). Благодаря относительно либеральным условиям лицензионного соглашения, систему X стали переносить на другие платформы, появились различные варианты ее реализации. Как и в случае с протоколом TCP/IP, элегантная архитектура и гибкость системы X позволили ей стать самым популярным в мире графическим интерфейсом пользователя (GUI), отличающимся от интерфейса системы Windows.

В 1998 году группа X Consortium, входящая в состав MIT, выработала общее направление развития протокола X. В течение последующих десяти лет эта группа, а также ее последователи постоянно работали над обновлениями протокола. Самой последней и большой версией является X11R7.5 — теперь к номерам версий добавляются новые числа, а не увеличиваются существующие.

Версия XFree86 являлась фактической реализацией X-сервера для многих платформ до тех пор, пока в 2004 году не изменились условия лицензионного соглашения, вследствие чего многие дистрибьюторы были вынуждены переключиться на ветвь XFree86, которая не подпадала под условия нового лицензионного соглашения. Эта ветвь поддерживается некоммерческой организацией X.Org Foundation и сегодня является доминирующей реализацией. Кроме того, версию X.Org Server можно переносить на платформу Windows для использования в среде совместимости Cygwin Linux. (Для Windows

существует несколько коммерческих версий X-сервера; более подробно об этом можно прочитать в разделе 30.2.)

В этой главе будет рассмотрена версия X.Org, которая используется во всех наших примерах систем, за исключением HP-UX. Реализации версий X.Org и XFree86 имеют расхождения в плане архитектуры, однако многие административные функции одинаковы. Чтобы использовать команды для версии XFree86, можно менять в них записи "xorg" на "xf86".

Система Solaris (начиная с версии 10) включала как X.Org Server, так и Xsun (еще одну реализацию системы X)[1]. Сервер Xsun применяется в системах SPARC, работающих под управлением Solaris 10, а системы x86 обычно используют реализацию сервера X.Org. Но в настоящее время X.Org уже поддерживает архитектуру SPARC, и проектом OpenSolaris заявлено, что сервер X.Org будет единственным вариантом поддержки платформы X в будущем. Поэтому здесь мы не будем рассматривать реализацию Xsun.

По умолчанию система AIX не включает среду X Window System. Чтобы инсталлировать ее, выполните команду `smitty easy-install`, укажите в качестве исходной библиотеки `lpp`, а затем выберите либо вариант CDE (для традиционной и "святой" для IBM платформы Motif), либо вариант KDE (для более современных платформ)[2]. Вы получите версию среды X.Org, высокая степень настраиваемости которой позволяет придать ей вид старых-добрых систем X эры Motif. Но под "ретро-оболочкой" вы обнаружите поддержку среды X11R7.5.

В системе X Window System можно выделить несколько ключевых компонентов. Она содержит диспетчер дисплеев (display manager), главной задачей которого является аутентификация пользователей, их регистрация и запуск исходной среды для сценариев запуска системы. Диспетчер дисплеев запускает X-сервер (X server), который определяет абстрактный интерфейс для растровых изображений и устройств ввода (например, для клавиатуры и мыши). Сценарии запуска запускают диспетчер окон (window manager), который позволяет пользователям перемещать окна, сворачивать, восстанавливать и изменять их размеры, а также управлять отдельными виртуальными рабочими столами. Наконец, на самом нижнем уровне приложения связываются с библиотекой графических интерфейсных элементов (widget library), которая реализует механизмы высокоуровневого пользовательского интерфейса, такие как кнопки и меню. На рис. 25.1 можно проследить взаимосвязь между диспетчером дисплеев, X-сервером и приложениями клиентов.

X-сервер понимает только базовый набор рисуемых примитивов в сетевом API; он не определяет программный интерфейс для высокоуровневых объектов, таких как кнопки, текстовые окна, меню и ползунки. Это обусловлено двумя причинами. Во-первых, это позволяет X-серверу работать на совершенно другом компьютере из приложения клиента. Во-вторых, X-сервер позволит серверу поддерживать разнообразные диспетчеры окон и наборы библиотек графических интерфейсных элементов.

У разработчиков приложений имеются свои библиотеки графических интерфейсных элементов и стандарты пользовательских интерфейсов. К сожалению, часто выбор зависит от "религиозной" приверженности, а не от того, какие задачи нужно решать разработчикам в действительности. Несмотря на удобства, которые открываются бла-

[1] Реализация Xsun включала поддержку технологии Display PostScript, которой когда-то пророчили стать языком отображения будущего.

[2] Совместная инсталляция обеих сред возможна, но не рекомендуется. Больше о настольных средах см. в разделе 25.5.

годаря свободе выбора, агностицизм пользовательского интерфейса системы X и сдача лидерских позиций в области дизайна в течение долгих лет сдерживали разработку качественных пользовательских интерфейсов. К счастью, степень завершенности основных X-сред заметно возросла. Настольные среды KDE и GNOME существенно оживляют современные веб-браузеры, дружественные к пользователю программы управления файлами, и новые мультимедийные возможности.

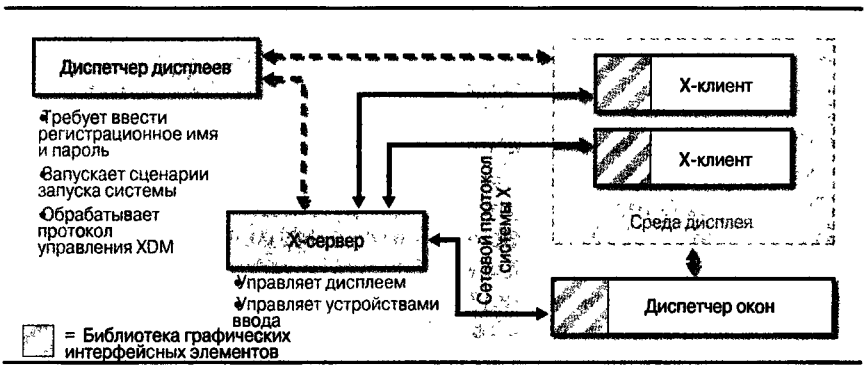



*Рис. 25.1. Модель "клиент/сервер" в системе X*

В этой главе мы поговорим о том, как запускаются программы на дистанционном дисплее и как выполняется процесс аутентификации. Кроме того, мы рассмотрим вопрос конфигурирования сервера X.Org, а также поиска и устранения ошибок в конфигурации. В конце главы мы вкратце поговорим о некоторых доступных диспетчерах окон и средах рабочего стола.

# 25.1. Диспетчер дисплеев

Диспетчер дисплеев представляет себя с помощью экрана регистрации, и обычно именно его пользователь видит первым, садясь за компьютер. Он не является обязательным; многие пользователи отключают эту утилиту и запускают систему X из текстовой консоли или из сценария `.login`, выполняя команду **`startx`** (она является оболочкой для программы **`xinit`**, которая запускает X-сервер).

Исходный диспетчер дисплеев называется **`xdm`** (для диспетчера дисплеев системы X), однако современные варианты, такие как **`gdm`** (диспетчер дисплеев для GNOME) и **`kdm`** (диспетчер дисплеев для KDE), предлагают дополнительный набор функций и имеют более привлекательный вид. Диспетчер дисплеев может позволить выполнять дистанционную регистрацию на других X-серверах посредством протокола XDMCP. Он может также поддерживать аутентификацию дисплея (см. раздел "Аутентификация клиентов" далее в этой главе).

Конфигурационные файлы, находящиеся в подкаталогах **`xdm`**, **`gdm`** или **`kdm`** каталога **`/etc/X11`**, определяют, как будет работать утилита **`xdm`**. К примеру, файл **`Xservers`** можно отредактировать таким образом, чтобы изменить номер дисплея, используемого для данного сервера, если множество серверов будет работать на других виртуальных терминалах. Или же можно изменить компоновку сервера с помощью опции **`-layout`**, если вы определили компоновку, подходящую для множества систем.

🕮 Подробные сведения о PAM можно найти в разделе 22.5.

Как правило, диспетчер дисплеев предлагает ввести имя пользователя и пароль. Пароль пользователя проверяется в соответствии с конфигурацией PAM (Pluggable Authentication Modules — подключаемые модули аутентификации), заданной в файле `.etc/pam.d/xdm` (или `gdm/kdm`, если вы используете диспетчеры дисплеев GNOME или KDE). На экране регистрации может быть предложено использование нескольких альтернативных сред рабочего стола, включающих важную безотказную опцию, о которой поговорим немного позже.

В заключение диспетчер дисплеев должен выполнить сценарий интерпретатора команд XSession, который настраивает среду рабочего стола пользователя. Сценарий XSession, который, как правило, находится в каталоге `/etc/X11/{xdm,gdm,kdm}`, является общесистемным сценарием запуска. Он задает стандартные настройки приложений, инсталлирует стандартные клавиатурные привязки и выбирает языковые настройки. Затем сценарий XSession выполняет личный пользовательский сценарий запуска, который обычно называется `~/.xsession`. Он запускает диспетчер дисплеев, панель задач, вспомогательные аплеты, а также может запускать и другие программы. GNOME и KDE имеют собственные сценарии запуска, которые конфигурируют рабочий стол пользователя в соответствии со средствами конфигурации GNOME и KDE. При этой схеме ошибок возникает меньше, чем при редактировании пользователями собственных сценариев запуска.

Когда работа сценария `~/.xsession` завершена, пользователь выводится из системы и диспетчер дисплеев снова предлагает ввести имя пользователя и пароль. Таким образом, сценарий `~/.xsession` должен запускать все программы в фоновом режиме (добавляя символ `&` в конце каждой команды), кроме самой последней, которой обычно является диспетчер окон. (Если все команды в сценарии `~/.xsession` будут работать в фоновом режиме, работа сценария будет тут же прекращена и пользователь будет выведен из системы сразу же после входа в нее.) Если же диспетчер окон будет работать в качестве приоритетного процесса, пользователь сможет выйти из системы только после того, как будет завершена работа диспетчера окон.

Безотказная опция позволяет пользователям входить в систему с целью устранения сбоев в проблемных сценариях запуска. Как правило, эту опцию можно выбирать на экране регистрации диспетчера дисплеев. В этом случае открывается только простое окно терминала; после закрытия окна пользователь выходит из системы. Благодаря этому пользователь может самостоятельно устранить возникшие неполадки, не обращаясь за помощью к администратору.

Как правило, самой распространенной проблемой при запуске системы является как раз то, что процессу не присваивается высокий уровень приоритета и он остается работать в фоновом режиме. Однако это не единственная проблема, которая может возникнуть. Если причину возникновения проблемы установить сложно, можно просмотреть файл `~/.xsession-errors`, в котором содержатся результаты выполнения команд в сценарии `~/.xsession`. Постарайтесь найти в этом файле ошибки или какое-либо необычное поведение системы. Если и это не поможет, переместите сценарий `~/.xsession` в какое-нибудь изолированное место и попробуйте войти в систему без него. Затем постепенно восстанавливайте одну-две строки до тех пор, пока не будет найдена "проблемная" строка.

## 25.2. Процесс запуска X-приложения

Процедура, необходимая для запуска X-приложения, на первый взгляд может показаться слишком сложной. Тем не менее вскоре вы обнаружите, что клиент-серверная модель дисплеев предлагает высокую степень гибкости. Так как информация об обнов-

лении дисплея передается по сети, приложение (клиент) может работать на совершенно другом компьютере, а не на том, где оно отображает свой графический интерфейс пользователя (на сервере). X-сервер может соединяться со многими другими приложениями, каждое из которых будет работать на отдельном компьютере.

Чтобы эта схема работала, клиенты должны знать, к какому дисплею нужно подключаться и какой экран нужно занимать на этом дисплее. После соединения с дисплеем клиенты должны пройти аутентификацию на X-сервере, гарантируя, что человек, сидящий перед дисплеем, имеет право устанавливать соединение.

🕮 Подробные сведения о протоколе SSH даны в разделе 22.10.

Тем не менее даже после прохождения аутентификации система X не обязательно будет полностью защищена. Еще один безопасный способ управления соединениями заключается в использовании протокола SSH (см. раздел "Перенаправление соединений с помощью протокола SSH" далее в этой главе). Мы настоятельно рекомендуем использовать протокол SSH для соединений системы X с Интернетом. В любом случае это будет нелишним для локальной передачи данных.

## Переменная окружения DISPLAY

X-приложения обращаются к переменной окружения **DISPLAY** для того, чтобы узнать, где они будут отображать свои данные. Эта переменная содержит имя или IP-адрес сервера, номер дисплея (он указывает на определенный экземпляр X-сервера, с которым нужно соединиться) и необязательный номер экрана (для дисплеев с несколькими мониторами). Когда приложения работают на том же компьютере, на котором отображаются их интерфейсы, для простоты записи большинство этих параметров можно опустить.

Следующий пример показывает и формат информации, представленной на дисплее, и синтаксис интерпретатора команд **bash**, используемого для задания переменной окружения.

```
client$ DISPLAY=имя_сервера.домен.com:10.2; export DISPLAY
```

Эта запись расшифровывается так: X-приложение находится по адресу *имя_сервера.домен*.**com**, дисплей 10, экран 2. Приложения устанавливают TCP-соединение с сервером на порту с номером 6000 плюс номер дисплея (в данном примере это порт 6010), на котором X-сервер, обрабатывающий данный дисплей, должен вести прослушивание.

Следует иметь в виду, что каждый процесс имеет собственные переменные окружения. Если переменную окружения **DISPLAY** задать для интерпретатора команд (оболочки), то ее значение будет передано только тем программам, которые будут запускаться этим интерпретатором. Если выполнить приведенные выше команды в одной программе **xterm**, а затем попытаться запустить любимое X-приложение в другой программе, то приложение не будет иметь доступа к созданной вами переменной **DISPLAY**.

Обратить внимание нужно и на другой момент. Дело в том, что хотя X-приложения посылают свои графические выходные данные на обозначенный X-сервер, они по-прежнему имеют локальные каналы **stdout** и **stderr**. На терминальное окно, из которого было запущено X-приложение, могут направляться некоторые выходные данные об ошибках.

Если и клиент, и сервер находятся в вашей организации, вы можете убрать из переменной **DISPLAY** полное доменное имя сервера, в зависимости от того, каким образом был сконфигурирован распознаватель вашего сервера имен. Поскольку большинство си-

стем работает на одном X-сервере, дисплей обычно имеет нулевой номер. Номер экрана можно опустить, в результате чего будет подразумеваться номер 0. Итак, чаще всего удобнее будет присваивать переменной **DISPLAY** значение *имя_сервера*:**0**.

О конфигурировании распознавателя DNS рассказывалось в разделе 17.3.

Если приложение клиента будет работать на том же компьютере, что и X-сервер, переменную **DISPLAY** можно еще больше упростить, опустив имя компьютера. Это не просто косметическое изменение: если указать пустое имя, то для связи с X-сервером библиотеки клиента будут использовать сокет домена UNIX, а не сокет сети. Кроме того, что этот тип соединения является более быстрым и эффективным, он позволяет обойти любые ограничения, вводимые брандмауэром в локальной системе, которые блокируют внешние соединения с системой X. Наиболее простым из возможных значений переменной окружения **DISPLAY** является нулевое значение (**:0**).

## Аутентификация клиентов

Несмотря на то что X-среда обычно считается относительно незащищенной, любая мера предосторожности поможет защититься от несанкционированного доступа. Ранее, до того как вопрос безопасности системы стал крайне важным, X-серверы с помощью команды **xhost** принимали соединения от любого клиента, работающего на компьютере, который помечался как безопасный. Однако в связи с тем, что любой пользователь, работающий на этом компьютере, мог соединиться с вашим дисплеем и причинить вред (случайно или преднамеренно), от метода предоставления доступа посредством команды **xhost** отказались навсегда. Больше о нем и мы не будем упоминать.

Наиболее распространенным альтернативным вариантом защиты является аутентификация с использованием "магических" cookie-наборов (magic cookie). Если вы знакомы с работой в Интернете, то cookie-наборы вам тоже должны быть знакомы; в данном контексте cookie-наборы используются для аутентификации X-соединений. Суть в том, что диспетчер дисплеев X в самом начале процедуры регистрации в системе генерирует большое случайное число, которое и называется *cookie-набором*. Cookie-набор сервера записывается в файл **~/.Xauthority**, который находится в домашнем каталоге пользователя. Соединение могут выполнять любые клиенты, которым известен cookie-набор. Пользователи могут выполнять команду **xauth** для того, чтобы просматривать существующие cookie-наборы и добавлять в этот файл собственные cookie-наборы.

Как работает эта схема, проще всего продемонстрировать на примере. Предположим, что вы задали вашу переменную **DISPLAY** в системе клиента для отображения X-приложений на компьютере, за которым работаете. Однако когда вы запускаете программу, на экран выводится ошибка, которая выглядит примерно так.

```
client$ xprogram -display server:0
Xlib: connection to "server:0.0" refused by server
xprogram:  unable to open display 'server:0'
```

Это сообщение говорит о том, что у клиента нет подходящего cookie-набора, поэтому дистанционный сервер отклонил данное соединение. Чтобы получить подходящий cookie-набор, нужно зарегистрироваться на сервере (это уже может быть сделано, если вы пытаетесь отобразить программу прямо на нем) и вывести список cookie-наборов сервера, выполнив команду **xauth list**.

```
server$ xauth list
server:0  MIT-MAGIC-COOKIE-1  f9d888df6077819ef4d788fab778dc9f
```

```
server/unix:0  MIT-MAGIC-COOKIE-1  f9d888df6077819ef4d788fab778dc9f
localhost:0  MIT-MAGIC-COOKIE-1  cb6cbf9e5c24128749feddd47f0e0779
```

Каждый сетевой интерфейс на сервере имеет свою запись. В данном примере мы имеем cookie-набор для Ethernet, для сокета домена UNIX, который используется для локальных соединений, и для сетевого интерфейса обратной связи localhost.

Самый простой способ "обзавестись" cookie-набором на клиенте (при условии, что не используется сетевой протокол SSH, который сам заботится о cookie-наборах) заключается в применении знакомой всем операции вырезания и вставки. Многие эмуляторы терминалов (например, **xterm**[3]) позволяют выделять текст с помощью мыши и вставлять его в другом окне, обычно посредством щелчка средней кнопкой мыши. В этом случае удобно то, что команда **xauth add** принимает в качестве ввода тот же формат, который отображает команда **xauth list**. Добавить cookie-набор к клиенту можно следующим образом.

```
client$ xauth add server:0 MIT-MAGIC-COOKIE-1
9d888df6077819ef4d788fab778dc9f
```

Чтобы убедиться в том, что cookie-набор был добавлен правильно, нужно выполнить команду **xauth list** на клиенте. Если переменная окружения **DISPLAY** задана и добавлен действительный "магический" cookie-набор, приложения смогут отображать свои данные на сервере.

Если у вас возникнут проблемы с работой cookie-наборов, можно на время вернуться к процедуре аутентификации с использованием команды **xhost** — просто для того, чтобы посмотреть, не возникают ли какие-то проблемы в ней (например, ограничения, выдвигаемые брандмауэрами или локальной сетью, которые не предоставляют клиенту доступ к серверу). Чтобы отключить аутентификацию **xhost** после выполненного вами теста, нужно выполнить команду **xhost -** (т.е. **xhost** со знаком дефиса в качестве единственного аргумента).

## Перенаправление соединений с помощью протокола SSH

"Магические" cookie-наборы усиливают защиту, однако использовать одни лишь cookie-наборы недостаточно. Любой пользователь, который может получить cookie-набор вашего дисплея, сможет подключиться к нему и запустить программы, которые будут отслеживать ваши действия. Даже без вашего cookie-набора X-протокол будет передавать данные по сети без шифрования, в результате чего буквально каждый пользователь сможет их перехватывать.

🕮 Подробные сведения о протоколе SSH даны в разделе 22.10.

Усилить защиту можно с помощью протокола SSH (secure shell — защищенный интерпретатор команд). Этот протокол обеспечивает аутентификацию и шифрование на терминале. Тем не менее SSH может также передавать произвольные сетевые данные, включая данные протокола X, по защищенному каналу. Механизм перенаправления, используемый в системе X, подобен обобщенному перенаправлению портов с помощью SSH, однако поскольку SSH знает об особенностях системы X, вы получаете дополнительные преимущества, включая псевдодисплей на дистанционном компьютере и "договорную" передачу "магических" cookie-наборов.

Протокол SSH обычно используется для передачи данных между компьютером, на котором работает X-сервер, и компьютером, на котором функционируют X-программы.

[3] Или команда **aixterm** в системе AIX. Мудро, да?

Эта схема может привести читателя в замешательство, поскольку *SSH-клиент* работает на том же компьютере, что и *X-сервер*, и соединяется с *SSH-сервером*, который работает на том же компьютере, где работают приложения *X-клиента*. Ситуация усугубляется еще и тем, что виртуальный дисплей, который формирует SSH для вашего X-сервера, является локальным по отношению к дистанционной системе. На рис. 25.2 показано, как осуществляется передача данных посредством SSH-соединения в системе X.

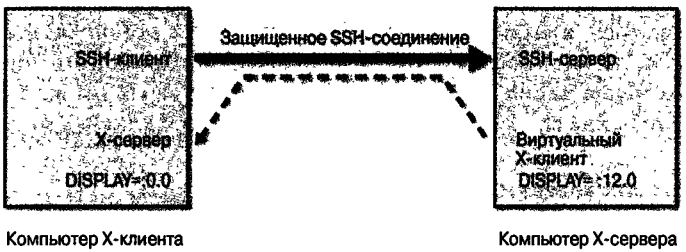



*Рис. 25.2. Использование протокола SSH в системе X*

Вашу переменную окружения **DISPLAY** и информацию об аутентификации можно задать автоматически посредством команды **ssh**. Номер дисплея начинается с **:10.0** и увеличивается на единицу для каждого SSH-соединения, при котором передаются данные в системе X.

В качестве примера можно предложить следующую последовательность.

```
x-server$ ssh -v -X x-client.mydomain.com
SSH-2.0-OpenSSH_5.1
debug1: Reading configuration data /home/boggs/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to x-client.mydomain.com [192.168.15.9] port 22.
debug1: Connection established.
Enter passphrase for key '/home/boggs/.ssh/id_rsa':
debug1: read PEM private key done: type RSA
debug1: Authentication succeeded (publickey).
debug1: Entering interactive session.
debug1: Requesting X11 forwarding with authentication spoofing.
debug1: Requesting authentication agent forwarding.
x-client$
```

В последних двух строках видно, что клиент получает запрос на перенаправление данных для X11-приложений. Перенаправление в системе X может быть разрешено как на SSH-сервере, так и на SSH-клиенте; кроме того, у клиента должен быть подходящий cookie-набор для сервера. Если у вас не получается организовать перенаправление, попробуйте использовать флаги **-X** и **-v**, как было показано выше (для OpenSSH), чтобы явным образом разрешить перенаправление в системе X и запросить подробные выходные данные[4]. Также можно просмотреть глобальные конфигурационные файлы SSH в каталоге **/etc/ssh** и убедиться в том, что перенаправление в системе X11 не было отме-

[4] Обратите внимание на то, что команда *ssh* также имеет флаг *-Y*, который "выражает доверие" соединениям со всеми клиентами. Эта возможность поможет решить некоторые проблемы перенаправления данных, но использовать ее все же нужно с предельной осторожностью.

нено администратором. Сразу после входа в систему вы можете проверить ваш дисплей и магические cookie-наборы.

```
x-client$ echo $DISPLAY
localhost:12.0
x-client$ xauth list
x-client/unix:12  MIT-MAGIC-COOKIE-1  a54b67121eb94c8a807f3ab0a67a51f2
```

Обратите внимание на то, что переменная **DISPLAY** указывает на виртуальный дисплей на SSH-сервере. Остальным SSH-соединениям (производимым как вами, так и другими пользователями) присваиваются разные виртуальные номера дисплеев. Если переменная **DISPLAY** и cookie-набор будут заданы правильно, вы сможете запустить приложение клиента.

```
x-client$ xeyes
  debug1: client_input_channel_open: ctype x11 rchan 4 win 65536 max 16384
  debug1: client_request_x11: request from 127.0.0.1 35411
  debug1: channel 1: new [x11]
  debug1: confirm x11
  debug1: channel 1: FORCE input drain
```

Если с помощью команды **ssh -v** разрешить отображение отладочной информации, вы увидите, что **ssh** получила запрос на X-соединение и покорно перенаправила его X-серверу. Сам процесс перенаправления может быть немного медленным, однако приложение в конечном счете должно появиться на вашем экране.

# 25.3. Конфигурирование X-сервера

Сервер X.Org, или **Xorg**, сложно поддается конфигурированию на конкретном аппаратном обеспечении. Но после определенных работ по модернизации сервера Xorg многие системы смогли успешно его запускать вообще без файла конфигурации. При этом сохранилась возможность вручную адаптировать сервер Xorg к широкому разнообразию оборудования для графики, устройств ввода, режимов видео, разрешений и глубины цвета, которую поддерживают устройства.

Очень хорошо, если ваша система нормально работает без файла конфигурации сервера Xorg! Это возможно, например, при использовании модуля KMS, который описан ниже в этой главе. В противном случае у вас есть два варианта. Первый вариант состоит в ручном конфигурировании файла **xorg.conf** (см. описание ниже). Причем в некоторых ситуациях этот вариант, по правде говоря, может оказаться вашим единственным выходом из положения. Второй вариант заключается в использовании утилиты **xrandr** для конфигурирования вашего сервера (подробности ниже).

Конфигурационный файл сервера Xorg обычно находится в каталоге **/etc/X11/xorg.conf**, однако X-сервер будет просматривать целые кипы каталогов, пытаясь найти его. Полный список есть на man-странице, однако нужно иметь в виду, что некоторые пути, по которым Xorg ведет поиск, содержат имя компьютера и глобальную переменную, что облегчает хранение конфигурационных файлов для множества систем в одном месте.


Система AIX не требует файла конфигурации **xorg.conf** и пытается автоматически распознавать все типы AIX-оборудования. Вы можете передать X-серверу свои пожелания по конфигурации как аргументы.

Некоторые программы могут помочь вам в конфигурировании системы X (например, **xorgconfig**), однако вам желательно самим разобраться со структурой конфигура-

ционного файла на тот случай, если вы захотите самостоятельно просматривать или редактировать эти файлы. Первичную информацию по этому вопросу можно найти прямо на X-сервере, выполнив команду **Xorg -probeonly** и просмотрев выходные данные для видеоплаты и прочие тестовые значения. Если выполнить команду **Xorg -configure**, X-сервер создаст исходный конфигурационный файл, который будет основан на тестовых значениях. Для начала этой информации вам вполне хватит.

Файл **xorg.conf** состоит из нескольких разделов, каждый из которых начинается с ключевого слова **Section** и заканчивается ключевым словом **EndSection**. Наиболее распространенные типы разделов перечислены в табл. 25.1.

**Таблица 25.1. Разделы файла xorg.conf**

| Раздел | Описание |
|---|---|
| **ServerFlags** | Перечисляет главные конфигурационные параметры X-сервера |
| **Module** | Определяет динамически загружаемые расширения для ускоренной графики, визуализаторов шрифтов и т.п. |
| **Device** | Конфигурирует информацию о видеоплате, драйвере и оборудовании |
| **Monitor** | Описывает физические параметры монитора, включая синхронизацию и разрешения дисплеев |
| **Screen** | Связывает монитор с видеоплатой (**Device**) и определяет разрешения и глубину цвета, доступные в данной конфигурации |
| **InputDevice** | Определяет устройства ввода, такие как клавиатуры и мыши |
| **ServerLayout** | Связывает устройства ввода с экранами и позиционирует экраны относительно друг друга |

Как правило, создавать конфигурационный файл проще всего "снизу вверх", начиная с определения разделов для устройств ввода и вывода и комбинируя их в различных компоновках. При таком иерархическом подходе простой конфигурационный файл можно использовать для многих X-серверов, каждый с разным оборудованием. Этот подход будет удобен и для одной системы, в которой установлено несколько видеоплат и используется несколько мониторов.

На рис. 25.3 показано, как некоторые из этих разделов располагаются в иерархии конфигурации X.Org. Физический дисплей **Monitor** и видеоплата **Device** формируют экран **Screen**. Набор экранов **Screen** плюс устройства ввода **InputDevices** формируют компоновку **ServerLayout**. В конфигурационном файле может быть определено множество компоновок сервера, однако только одна из них будет активной в данном экземпляре процесса **Xorg**.

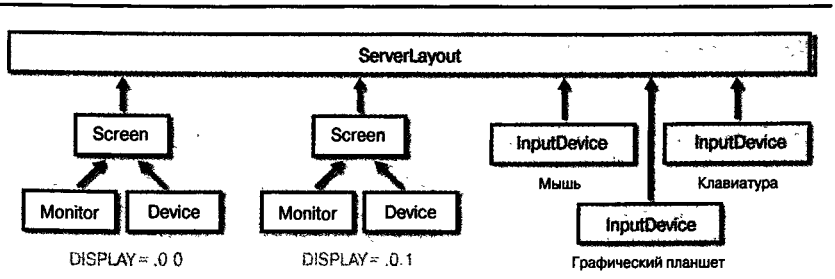



*Рис. 25.3. Взаимоотношение между разделами конфигурации* **xorg.conf**

Некоторые разделы, составляющие файл **xorg.conf**, являются относительно фиксированными. Стандартные разделы часто могут браться прямо из существующего конфигурационного файла или из его образца. Другие (например, **Device**, **Monitor**, **Screen**, **InputDevice** и **ServerLayout**) зависят от того, как настроено оборудование компьютера. В следующих подразделах мы рассмотрим наиболее интересные из этих разделов.

## Раздел Device

В этом разделе описывается определенная видеоплата. Вы должны указать строку, которая идентифицирует плату и драйвер, подходящий для данного устройства. Драйвер загружается только в том случае, если на это устройство имеется ссылка в соответствующем разделе **Screen**. Типичная раздел **Device** может выглядеть примерно так.

```
Section "Device"
   Identifier          "Videocard0"
   Driver              "radeon"
   опция               значение
   ...
EndSection
```

На странице справочного руководства этого драйвера (в данном случае это **radeon**) перечисляются и описываются оборудование, которым он управляет, и опции, которые он поддерживает. Если во время работы возникают какие-то странные явления, можно отключить аппаратное ускорение (если эта функция поддерживается), сократив доступ к видеопамяти, или модифицировать параметры интерфейса. Однако перед тем как что-либо изменять, желательно проконсультироваться в Сети или у специалиста.

## Раздел Monitor

В разделе **Monitor** описываются дисплеи, подключенные к вашему компьютеру. В ней могут быть определены детализированные значения синхронизации. Информация о синхронизации необходима для оборудования старого образца; тем не менее она может быть полезной для многих современных мониторов. Спецификации дисплеев обычно можно получить на веб-сайтах производителей, однако самым лучшим источником, естественно, является руководство, прилагаемое к монитору. В любом случае вам нужно будет знать как минимум частоты горизонтальной и вертикальной развертки для вашей модели.

Типичный раздел Monitor выглядит примерно так.

```
Section "Monitor"
   Identifier          "ViewSonic"
   Option              "DPMS"
   HorizSync           30-65
   VertRefresh         50-120
EndSection
```

Как и во всех разделах, строка **Identifier** присваивает имя, с помощью которого вы будете ссылаться на данный монитор. Здесь мы включили функцию DPMS (Display Power Management Signaling — сигналы управления энергопотреблением дисплеев), благодаря которой X-сервер будет выключать монитор, когда мы будем делать перерыв на чашечку кофе или просто отлучаться на некоторое время.

Строки **HorizSync** и **VertRefresh** должны содержать значения, подходящие для вашего CRT-монитора (с электронно-лучевой трубкой). Они могут быть определены

в виде диапазона частот (как это сделано в приведенном выше примере) или в виде набора дискретных значений, разделяемых запятыми. Драйвер теоретически может протестировать поддерживаемые режимы, однако если параметры будут определены, он не будет использовать неподдерживаемые частоты.

## Раздел Screen

В разделе **Screen** устройство (видеоплата) связывается с монитором при определенной глубине цвета, а также задаются параметры разрешения дисплея. Ниже показан пример, в котором используются упомянутые видеоплата и монитор.

```
Section "Screen"
  Identifier "Screen0"
  Device "Videocard0"
  Monitor "ViewSonic"
  DefaultDepth 24
  Subsection "Display"
    Depth 8
    Modes "640x400"
  EndSubsection
  Subsection "Display"
    Depth 16
    Modes "640x400" "640x480" "800x600" "1024x768"
  EndSubsection
  Subsection "Display"
    Depth 24
    Modes "1280x1024" "1024x768" "800x600" "640x400" "640x480"
  EndSubsection
EndSection
```

Как можно было предположить, в строке **Identifier** указывается имя экрана и перечисляются идентификаторы ранее определенной видеоплаты и монитора. Это первый представленный нами раздел, в котором имеются подразделы. Для каждой глубины цвета определяется одн подраздел, а стандартное значение глубины задается в поле **DefaultDepth**.

Данный экземпляр X-сервера может работать только при одной глубине цвета. В самом начале сервер определяет, какие разрешения поддерживаются для данной глубины цвета. Возможные разрешения зависят, как правило, от видеоплаты. В разделе "Специальные комбинации клавиш в системе X" этой главы описано, в каком порядке производится выбор определенных здесь разрешений.

Любая современная видеоплата должна уметь работать с вашим монитором с полным его разрешением при 24- или 32-битовом цвете. Если вам понадобится запустить старые программы, для которых сервер должен работать с 8-битовым цветом, запустите другой X-сервер в отдельной виртуальной консоли. Чтобы отменить действие опции **DefaultDepth**, в командной строке **Xorg** используйте флаг **-depth 8**.

## Раздел InputDevice

В разделе **InputDevice** описывается устройство ввода (например, клавиатура или мышь). Каждому устройству соответствует собственный раздел **InputDevice** и, как и в других разделах, имя в поле **Identifier**. Если вы используете один конфигурационный файл для нескольких компьютеров с разным оборудованием, можете определить

все устройства ввода; использоваться будут только те из них, которые указаны в разделе **ServerLayout**. Ниже показано типичное определение клавиатуры.

```
Section "InputDevice"
   Identifier   "Generic Keyboard"
   Driver       "Keyboard"
   Option       "AutoRepeat" "500 30"
   Option       "XkbModel" "pc104"
   Option       "XkbLayout" "us"
EndSection
```

Среди прочего, в определении клавиатуры можно задать такие опции, которые определят настройку клавиш <Ctrl> и <Caps Lock>. В данном примере опция **AutoRepeat** определяет, как долго нужно удерживать клавишу нажатой, чтобы это привело к повтору нажатия символа, и как быстро будет повторяться ввод символа.

Конфигурация мыши определяется в другом разделе InputDevice.

```
Section "InputDevice"
   Identifier   "Generic Mouse"
   Driver       "mouse"
   Option       "CorePointer"
   Option       "Device"  "/dev/input/mice"
   Option       "Protocol" "IMPS/2"
   Option       "Emulate3Buttons" "off"
   Option       "ZAxisMapping" "4 5"
EndSection
```

Опция **CorePointer** обозначает эту мышь как основное указывающее устройство в системе. Файл устройства (**Device**), связанный с мышью, определен в поле **Option**. Файлы мультиплексоров мыши перечислены в табл. 25.2.

**Таблица 25.2. Файлы мультиплексоров мыши**

| ОС | Файл устройства |
|---|---|
| Linux | **/dev/input/mice** |
| Solaris | **/dev/mouse** |
| HP-UX | **/dev/deviceFileSystem/mouseMux** |
| AIX | **/dev/mouse0** |

Протокол связи зависит от конкретного производителя мыши, ее функциональных характеристик и интерфейса. Если ему присвоить значение **auto**, сервер попытается определить протокол самостоятельно. Если колесико мыши не работает, попробуйте воспользоваться протоколом IMPS/2. Если мышь содержит много кнопок (больше трех), используйте протокол ExplorerPS/2. Некоторые пользователи систем Solaris сообщают об успешном использовании протокола VUID.

Опция **Emulate3Button** позволяет двухкнопочным мышам имитировать работу трехкнопочных, определяя щелчок двумя кнопками как щелчок средней кнопкой мыши. Опция **ZAxisMapping** иногда нужна для поддержки прокрутки колесика или работы джойстика; эта поддержка осуществляется путем сопоставления кнопок. Большинство современных моделей мыши имеют как минимум три кнопки, колесико прокрутки, встроенный MP3-плеер, щетку для массажа ступней и холодильник для пива[5].

[5] Xorg поддерживает не все опции. Некоторые продаются отдельно.

## Раздел ServerLayout

Этот раздел является узлом верхнего уровня в иерархии конфигурации. Каждая аппаратная конфигурация, на которой будет работать сервер, должна иметь собственный экземпляр раздела **ServerLayout**. Компоновка, используемая определенным X-сервером, обычно определяется в командной строке сервера.

Этот раздел связывает вместе все остальные разделы, представляя X-дисплей. Она начинается с необходимого поля **Identifier**, в котором указывается данная конкретная компоновка. После этого набор экранов связывается с компоновкой[6]. Если к отдельным видеоплатам подключено несколько мониторов, то каждый экран будет определяться вместе с необязательными направлениями, чтобы показать их физическое месторасположение. В данном примере один экран (**Screen 1**) находится слева, а другой (**Screen 2**) — справа.

Ниже показан пример всего раздела **ServerLayout**.

```
Section "ServerLayout"
  Identifier    "Simple Layout"
  Screen       "Screen 1" LeftOf "Screen 2"
  Screen       "Screen 2" RightOf "Screen 1"
  InputDevice   "Generic Mouse" "CorePointer"
  InputDevice   "Generic Keyboard" "CoreKeyboard"
  Option       "BlankTime" "10" # Погасить экран через 10 минут
  Option       "StandbyTime" "20" # Отключить экран через 20 минут (DPMS)
  Option       "SuspendTime" "60" # Полное засыпание через 60 минут (DPMS)
  Option       "OffTime" "120"# Отключить DPMS-монитор через 2 часа
EndSection
```

Некоторые видеоплаты могут управлять одновременно несколькими мониторами. В данном случае в разделе **ServerLayout** определен только один экран **Screen**. За списком экранов следует набор устройств ввода, которые необходимо связать с этой компоновкой. Опции **CorePointer** и **CoreKeyboard** перемещены в раздел **InputDevice**, чтобы показать, что эти устройства должны быть активными в данной конфигурации. Эти опции могут быть заданы прямо в соответствующих разделах **InputDevice**, однако проще всего задавать их в разделе **ServerLayout**.

В последних нескольких строках конфигурируется несколько опций, специфических для компоновки. В приведенном выше примере все они связаны со спецификацией DPMS (Display Power Management System — управление режимом энергосбережения монитора), благодаря которой мониторы, рассчитанные на работу в режиме Energy Star (режим пониженного энергопотребления мониторов), "знают", когда нужно отключать питание. Мониторы должны также иметь собственные DMPS-опции, разрешенные в соответствующих разделах **Monitor**.

## Утилита xrandr: конфигуратор X-сервера

Расширение X-сервера RandR (X Resize and Rotate Extension) позволяет клиентам динамически изменять разрешение, частоту развертки, ориентацию (использовать "портретный" режим) и отображение экранов их X-серверов без перезапуска X-сервера. Утилита командной строки **xrandr** представляет собой интерфейс с расширением RandR.

[6] Помните, что экраны идентифицируют комбинацию "монитор/видеоплата" при определенной глубине цвета.

Безусловно, все мы (исключительно из любви к искусству программирования) с удовольствием потратили бы несколько дней, тщательно "вылизывая" каждую строку файла **xorg.conf**, на поддержку новейшей системы SUPERINATOR 3000 с ее четырьмя роскошными дисплеями. Но во многих случаях это (т.е. работу по конфигурированию) может сделать за нас утилита **xrandr** (тем самым освободив для нас время для встречи с друзьями). Выполненная без аргументов, утилита **xrandr** показывает доступные дисплеи и их возможные варианты разрешения.

```
$ xrandr
VGA-0 connected 1024x768+0+0 (normal left inverted right x a...) 0mm x 0mm
   1024x768    61.0    60.0    59.9    59.9
   800x600     60.3    61.0    59.9    56.2    59.8
   640x480     59.9    61.0    59.4    59.5
DVI-0 connected 1024x768+0+0 (normal left inverted right x a...) 0mm x 0mm
   1024x768    60.0    60.0
   800x600     60.3    59.9
   640x480     59.9    59.4
```

Вы можете задать разрешение для каждого дисплея, а также указать расположение конкретного монитора относительно других[7]. Рассмотрим пример.

```
$ xrandr --auto --output VGA-0 --mode 800x600 --right-of DVI-0
```

Аргумент **--auto** включает все мониторы. Аргументы **--output** и **--mode** устанавливают для монитора с видеографической матрицей (VGA) разрешение 800×600, а аргумент **--right-of** определяет, что VGA-монитор должен физически располагаться справа от DVI-монитора. (Последняя опция необходима для упорядочения составляющих рабочего стола.) Для просмотра множества доступных опций выполните команду **xrandr --help**.

Если вы хотите выполнять утилиту **xrandr** автоматически при запуске X-сервера, поместите ее в файл **~/.xprofile**.

## Установка режима ядра

Для того чтобы сделать представление системы более гладким и устранить шум мерцания, ответственность за установку начального режима графического отображения в настоящее время перекладывается на ядро Linux посредством модуля KMS (kernel mode setting). Как и в версии ядра 2.6.30-10.12, KMS по умолчанию инициализирует видеокарту в начале процесса загрузки ядра.

Вы можете разрешить или запретить KMS с помощью настроек в конфигурационных файлах видеодрайвера каталога **/etc/modprobe.d**. Например, если у вас есть видеокарта ATI Radeon, вы сможете отключить KMS, добавив следующую строку в файл **/etc/modprobe.d/radeon.conf**.

```
options radeon modeset=0
```

Модуль KMS пока не поддерживает все типы видеокарт. Если вам посчастливилось быть обладателем поддерживаемой карты, то лучше всего переименовать файл **xorg.conf** так, чтобы X-сервер пытался стартовать без него и обращался к KMS-конфигурации.

[7] До начала использования утилиты `xrandr` выполните команду `Xorg -configure`, чтобы перевести файл `xorg.conf` в известное исходное состояние.

# 25.4. Устранение неполадок и отладка X-сервера

Конфигурация X-сервера прошла долгий путь за последние десять лет, однако она по-прежнему остается сложной, и вам будет непросто добиться того, чтобы все работало именно так, как вы того хотите. Вам придется экспериментировать с частотами мониторов, опциями драйверов, оригинальными драйверами или расширениями для трехмерной визуализации. Как правило, сбои в работе мониторов возникают именно тогда, когда вам крайне необходимо просмотреть информацию об отладке на экране. К счастью, сервер X.Org может дать вам любую необходимую информацию (и даже сверх того), которая поможет вам выявить причину возникновения проблемы.

## Специальные комбинации клавиш в системе X

Так как X-сервер принимает на себя управление вашей клавиатурой, дисплеем и мышью, вы можете себе представить, что он может оставить вам небольшое количество ресурсов и выключит питание системы, если что-то не будет работать. Однако прежде чем дело дойдет до этого, нужно попытаться выполнить некоторые действия.

Если нажать и удерживать клавиши <Ctrl> и <Alt>, а затем нажать функциональную клавишу (<F1>–<F6>), X-сервер переключит вас на один из текстовых виртуальных терминалов. В нем вы сможете войти в систему и выполнить отладку. Для того чтобы вернуться к X-серверу, работающему на терминале 7, нужно нажать <Alt+F7>[8] . Если вы работаете в сети, можете также попробовать войти в систему на другом компьютере, чтобы уничтожить X-сервер, прежде чем нажимать кнопку сброса.

Для поддержки виртуальной консоли в системе Solaris разрешите использование системы управления службами (Service Management Facility — SMF) **`svc:/system/vtdaemon:default`** и службы **`console-login:vt[2-6]`**.

Если монитор не синхронизируется с видеосигналом, поступающим с видеоплаты, попытайтесь изменить разрешение экрана. Доступные разрешения указаны в строке **`Modes`** раздела **`Screen`** конфигурационного файла. Строка **`Modes`**, являющаяся активной, зависит от глубины цвета; более подробно об этом см. раздел 25.3. Стандартные настройки X-сервера выбираются с учетом первого разрешения, указанного в активной строке **`Modes`**, однако вы можете переходить от одного разрешения к другому, удерживая клавиши <Alt> и <Ctrl> и нажимая клавиши со знаком "плюс" или "минус" на цифровой клавиатуре.

При нажатии комбинации клавиш <Ctrl+Alt+пробел> происходит немедленное уничтожение X-сервера. Если вы запустили сервер из консоли, окажетесь в ней после прекращения работы сервера. Если сервер был запущен диспетчером дисплеев, то диспетчер запустит новый сервер и предложит снова ввести имя пользователя и пароль. Вам нужно будет уничтожить диспетчер дисплеев (**`xdm`**, **`gdm`** и т.п.) из текстовой консоли, чтобы он не порождал новые X-серверы.

## Когда с X-сервером творится что-то неладное

После того как вновь получите возможность управлять компьютером, вы сможете найти причину возникновения проблемы. Проще всего начать с анализа выходных дан-

[8] При нажатии комбинации <Alt+функциональная клавиша> следует удерживать нажатой клавишу <Ctrl>, чтобы X-сервер мог переключить виртуальные терминалы; для текстовой консоли нажимать клавишу <Ctrl> не нужно.

ных X-сервера. Эти данные периодически появляются на первом виртуальном терминале (<Ctrl+Alt+F1>), на который поступают все выходные данные программы запуска. Как правило, выходные данные X-сервера направляются в регистрационный файл, например в файл **/var/log/Xorg.0.log** (**/var/X11/Xserver/logs/Xf86.0.log** в системе HP-UX).

Как будет показано ниже, каждой строке предшествует символ, определяющий ее категорию. Вы можете использовать эти символы для обозначения ошибок (**EE**) и предупреждений (**WW**), а также для того, чтобы определить, как сервер будет находить каждую порцию информации: посредством стандартных настроек (**==**), в конфигурационном файле (******), путем автоматического обнаружения (**--**) или в командной строке X-сервера (**++**).

Давайте проанализируем следующий фрагмент кода из системы Ubuntu.

```
X.Org X Server 1.6.0
Release Date: 2009-2-25
X Protocol Version 11, Revision 0
Build Operating System: Linux 2.6.24-23-server i686 Ubuntu
Current Operating System: Linux nutrient 2.6.28-11-generic #42-Ubuntu SMP
   Fri Apr 17 01:57:59 UTC 2009 i686
Build Date: 09 April 2009  02:10:02AM
xorg-server 2:1.6.0-0ubuntu14 (buildd@rothera.buildd)
   Before reporting problems, check http://wiki.x.org
   to make sure that you have the latest version.
Markers: (--) probed, (**) from config file, (==) default setting,
   (++) from command line, (!!) notice, (II) informational,
   (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Sun May 10 22:11:47 2009
(==) Using config file: "/etc/X11/xorg.conf"
(==) ServerLayout "MainLayout"
(**) |-->Screen "Screen 0" (0)
(**) |   |-->Monitor "Monitor 0"
(**) |   |-->Device "Console"
(**) |-->Input Device "Mouse0"
(**) |-->Input Device "Keyboard0"
...
```

В первых строках указывается версия X-сервера и номер протокола X11, которые он реализует. В последующих строках сообщается о том, что сервер использует стандартные значения для местонахождения регистрационного и конфигурационного файлов, а также компоновки активного сервера. Дисплей и устройства ввода в конфигурационном файле отображаются схематически.

Одной из часто встречаемых проблем, которые можно увидеть в журнале, является сложность с определенными разрешениями экрана, о чем обычно свидетельствует сообщение об ошибке "Unable to validate any modes; failing back to the default mode" (Невозможно проверить какой-либо из режимов; производится возврат к стандартному режиму). Если не определить список частот для вашего монитора, X-сервер попытается использовать их с помощью EDID-данных (Extended Display Identification Data — расширенные данные идентификации дисплея). Если ваш монитор не поддерживает EDID или если ваш монитор выключен, когда запускается система X, вам нужно указать диапазон частот для системы X в разделе **Monitor** конфигурационного файла.

Ошибка округления в результатах, полученных после тестирования EDID, может привести к тому, что некоторые разрешения будут недоступными, даже если они должны

поддерживаться вашей видеоплатой и монитором. Об этом свидетельствуют такие записи в журнале, как "No valid modes for 1280×1024; removing" (Нет подходящих режимов для разрешения 1280×1024; удаление). Решить эту проблему можно, приказав X-серверу игнорировать информацию EDID и использовать частоты, определенные вами в следующих строках раздела **Device**.

```
Option   "IgnoreEDID" "true"
Option   "UseEdidFreqs" "false"
```

В качестве другого примера предположим, что вы забыли правильно определить настройки мыши в разделе InputDevice. В этом случае выходные данные должны содержать сообщение об ошибке следующего рода.

```
(==) Using config file: "/etc/X11/xorg.conf"
Data incomplete in file /etc/X11/xorg.conf
  Undefined InputDevice "Mouse0" referenced by ServerLayout "MainLayout".
(EE) Problem parsing the config file
(EE) Error parsing the config file
Fatal server error:
no screens found
```

После того как система X заработает и вы войдете в нее, сможете запустить команду **xdpyinfo**, чтобы получить дополнительную информацию о конфигурации X-сервера[9]. После выполнения команды **xdpyinfo** вы получите имя дисплея и версию X-сервера. Кроме того, будет сообщено также о доступной глубине цвета, загруженных расширениях и установленных экранах, их размерах и конфигурации.

Выходные данные команды **xdpyinfo** можно проверять синтаксически с помощью сценария (такого, как **~/.xsession**), чтобы узнать размер активного экрана и правильно задать параметры рабочего стола. В процессе отладки команду **xdpyinfo** полезно выполнять для того, чтобы определить, что X-сервер работает и следит за запросами в сети, что у него правильно сконфигурирован экран и разрешение и что он работает с требуемой глубиной цвета. Если все это подтверждается, можно запускать X-приложения.

## 25.5. Краткое замечание о настольных средах

Гибкость модели "клиент/сервер" в системе X Window с годами привела к появлению наборов библиотек графических интерфейсных элементов, диспетчеров окон, обозревателей файлов, утилит панели инструментов и полезных программ. Первые полнофункциональные среды, OpenLook и Motif, были элегантными, но слишком узкоспециализированными, а условия лицензирования, распространяемые на библиотеки разработки и диспетчер окон, сделали их недоступными для широкой публики.

По мере того как приложения становились более совершенными и требовали более серьезной поддержки от базовой оконной системы, стало ясно, что без перехода к более передовой платфоме не обойтись. Как результат, в современных средах рабочего стола для Linux мы имеем двух крупных игроков: GNOME и KDE. Несмотря на то что некоторые пользователи имеют свое понимание того, какая из них является самым "истинным путем", обе среды являются полноценными диспетчерами рабочего стола. В сущ-

[9] Входить в систему X от имени суперпользователя не рекомендуется, поскольку эта операция может создать множество стандартных файлов запуска в домашнем каталоге суперпользователя, которым обычно является каталог **/** или **/root**. Кроме того, это небезопасно. Наоборот, входить в систему нужно от имени обычного пользователя и использовать утилиту **sudo**. В системах Ubuntu эта схема принята по умолчанию.

ности, работая в одном "королевстве", вы можете использовать приложения из другого "королевства". Вы должны лишь быть готовы к тому, что приложение будет выглядеть и вести себя по-другому.

Проект **freedesktop.org** посвящен созданию среды, которая позволит приложениям быть совместимыми с любой настольной средой.

## KDE

Среда KDE (расшифровывается как "K Desktop Environment — настольная среда K) написана на языке C++ и основана на библиотеке инструментов Qt. Ее предпочитают те пользователи, которым нравятся привлекательные элементы и эффекты интерфейса (прозрачные окна, тени и анимированные указатели мыши). Выглядит она привлекательно, однако может замедлить работу малопроизводительных персональных компьютеров. Работать в этой среде удобно тем пользователям, которые большую часть времени тратят на щелчки на рабочем столе, а не на запуск приложений.

Среду KDE часто предпочитают пользователи, перешедшие из Windows или Mac, и причина тому — все та же привлекательность графики. Она придется по вкусу также тем пользователям, кому нравится самостоятельно настраивать все параметры среды. Для остальных пользователей среда KDE вряд ли подойдет — они могут воспользоваться средой GNOME.

Приложения, написанные для среды KDE, практически всегда содержат букву "K" в своем названии (например, Konqueror — обозреватель Веб или файлов, Konsole — имитатор терминала, Kword — текстовый процессор). Стандартный диспетчер окон, KWin, поддерживает стандарт Window Manager Specification проекта **freedesktop.org**, конфигурируемые оболочки (skin) для изменения внешнего вида, а также многое другое. Комплект приложений KOffice позволяет обрабатывать тексты, электронные таблицы и проводить презентации. KDE предлагает обширный набор инструментов для разработки, включая интегрированную среду разработки (IDE).

## GNOME

Среда GNOME написана на языке C и основана на библиотеке графических интерфейсных элементов GTK+. Название *GNOME* первоначально являлось акронимом GNU Network Object Model Environment (Среда сетевых объектных моделей GNU), однако в наши дни это не имеет никакого значения — сейчас существует просто имя GNOME.

С последним добавлением поддержки Compiz (`compiz.org`) — композитного менеджера окон для X Window System — среда GNOME приобрела множество симпатичных функций, которых в ней ранее недоставало. Среда GNOME менее броская, чем KDE, предлагает меньше возможностей для конфигурирования и, вообще, является менее согласованной. Однако она гораздо понятнее, быстрее и проще, чем KDE. Среда GNOME используется во многих дистрибутивах в качестве стандартной настольной среды. Подобно KDE, GNOME имеет богатый набор приложений. Ее приложения обычно идентифицируются буквой "G" в названии приложения. Исключением является стандартный диспетчер окон, называемый Metacity, который предлагает базовые функции работы с окнами и оболочки, позволяющие изменять внешний вид. В соответствии с моделью GNOME, Metacity обязана быть "шустрой и проворной".

Если вам понадобятся какие-то дополнительные особенности вроде интеллектуального размещения окон, нужна будет поддержка внешних приложений, таких как **brightside** или **devilspie**. (В области "шика-блеска", KDE все еще занимает неплохие позиции.)

К офисным приложениям относятся AbiWord (для обработки текстов), Gnumeric (для работы с электронными таблицами), а также один из наиболее впечатляющих проектов — GIMP, позволяющий производить обработку изображений. Еще есть диспетчер файлов Nautilus. Как и KDE, для разработчиков приложений GNOME предлагает обширную инфраструктуру. Подводя итоги, можно с уверенностью сказать, что GNOME предлагает функциональную архитектуру для разработки приложений в простой в использовании среде рабочего стола.

### Что лучше: GNOME или KDE?

Попробуйте-ка задать такой вопрос на одном из открытых форумов, как тут же начнется "большой базар". Вопрос о предпочтениях практически всегда затрагивает личные чувства и становится причиной настоящих "крестовых походов", поэтому вы можете не согласиться с тем, что написано в следующих двух абзацах.

Конечно же, лучше всего попробовать в деле обе среды и решить самостоятельно, какая из них подходит вам больше всего. Следует иметь в виду, что ваши приятели, ваши пользователи и ваш администратор могут иметь совершенно разные предпочтения относительно среды рабочего стола, и в этом нет ничего плохого.

Не забывайте, что ваш выбор, сделанный в пользу той или иной среды, не означает, что вы сможете работать только с некоторыми приложениями. Не важно, какую среду вы выберете, — вы сможете выбирать приложения из всего имеющегося программного обеспечения, доступного для каждого из этих (и других) проектов с открытым исходным кодом.

## 25.6. Рекомендуемая литература

- На домашней странице X.Org (**www.x.org**) содержится информация о свежих выпусках, а также ссылки на проект Wiki X.Org, списки рассылки и ссылки для загрузки.
- На man-страницах Xserver и Xorg (или просто X в системе AIX) рассмотрены общие опции X-сервера и специфические опции командной строки **Xorg**. Они содержат также обзор принципов работы X-сервера.
- На man-странице **xorg.conf** приводится конфигурационный файл с подробным описанием его разделов и перечислены драйверы видеоплат в разделе **REFERENCES** (Справка). На этой странице уточните, какой драйвер управляет вашей видеоплатой, а затем ознакомьтесь с man-страницей этого драйвера, чтобы узнать обо всех его опциях.

## 25.7. Упражнения

25.1. Используя SSH, запустите программу X по сети. Используйте команду **ssh -v**, чтобы удостовериться, что перенаправление настроено должным образом. Какое значение будет иметь переменная окружения **DISPLAY** после того, как вы войдете в систему? Выведите на экран список cookie-наборов, выполнив команду **xauth**, и убедитесь в том, что для данного дисплея включена проверка достоверности с использованием cookie-наборов.

25.2. Напишите команду интерпретатора команд или сценарий для проверки синтаксиса выходных данных **xdpyinfo** и напечатайте текущее разрешение экрана в формате X×Y (например, 1024×768).

25.3. Проверьте регистрационный файл Xorg (**/var/logXorg.o.log**) и на его основе постарайтесь ответить на следующие вопросы:

а) какая видеоплата используется и какой драйвер управляет ее работой?

б) какой объем памяти установлен на видеоплате?

в) использовались ли данные EDID для тестирования настроек монитора? Как это можно узнать?

г) какие режимы (разрешения) поддерживаются?

д) включена ли функция DPMS?

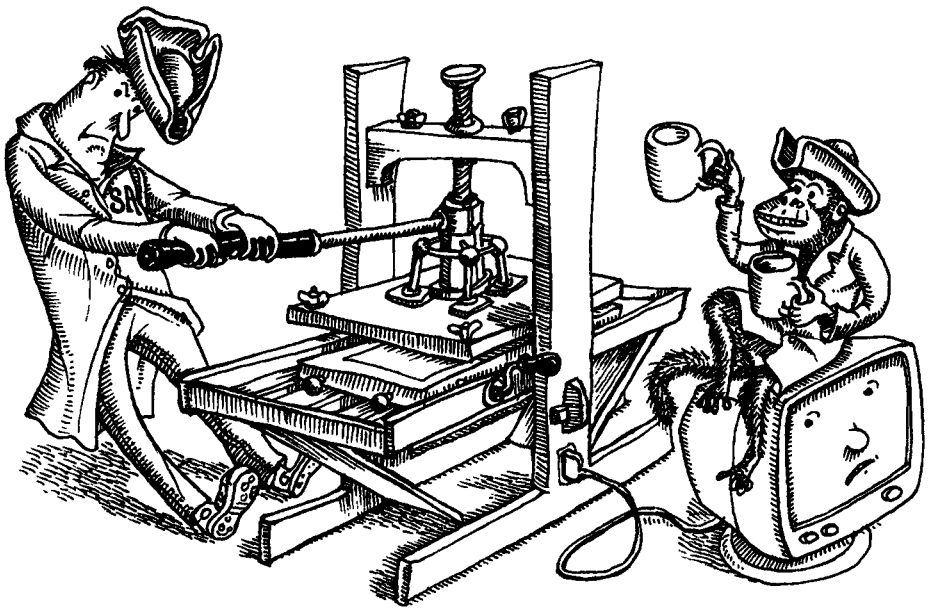
е) о каких физических размерах экрана известно серверу?

ж) какой файл устройства используется для мыши?

25.4. Какой флаг отключает нелокальные TCP-соединения с сервером? Объясните, чем полезна эта опция?

# Глава 26

# Печать



В операционной системе UNIX процесс печати довольно запутан. Давайте разберемся.

В операционной системе Linux печатать довольно удобно. Точно так же удобно печатать в системе Mac OS X. Эти операционные системы используют современную, сложную, сетевую и безопасную систему печати Common UNIX Printing System (CUPS). Эта система предоставляет современный графический пользовательский интерфейс, основанный на использовании браузера, а также команды оболочки, позволяющие печатать и управлять системой печати с помощью сценариев.

Подобно тому как новейшие транспортные системы для передачи почтовых сообщений сохраняют команду **sendmail**, позволяющую выполнять старые сценарии (старым администраторам!) в память о старых славных днях, так и система CUPS допускает выполнение команд **lp** и **lpr**, обеспечивая обратную совместимость с традиционными системами печати в UNIX. В результате все довольны.

Судя по названию, можно было бы подумать, что систему CUPS можно найти во всех версиях UNIX. Увы, это не так. Ни одна из рассмотренных нами платформ UNIX — Solaris, HP-UX и AIX — ее не использует. И что еще хуже, быстрый поиск в системе Google показывает, что попытки инсталлировать систему CUPS на этих платформах обычно терпят неудачу.

Вместо нее упомянутые платформы предлагают варианты устаревших систем печати System V и BSD. Что было хорошо для компьютера PDP-11, то должно быть хорошо и для вас! К сожалению, в сфере печати документов доминируют системы Microsoft Windows и Mac OS, поэтому разработчики систем UNIX не испытывают давления со стороны пользователей и не торопятся улучшать поддержку печати. В то время как раз-

работчики платформы UNIX внедряют систему CUPS (а вы осваиваете системы Linux и Max OS), пользователи вынуждены изучать старые системы печати.

Начнем главу с общего обсуждения систем печати и терминологии, принятой в этой сфере. Затем перейдем к описанию разных систем печати на платформах UNIX и Linux, а также их архитектур. После этого обсудим специфику конфигурирования и управления принтерами и закончим кратким руководством по отладке систем печати, обзором дополнительного программного обеспечения для печати, а также советами общего характера.

Впрочем, прежде чем начать, стоит напомнить: системные администраторы часто считают печать менее приоритетной задачей, чем пользователи. Администраторы читают документы в сети, а пользователи часто хотят иметь бумажную копию и хотят, чтобы система печати работала 100% времени.

# 26.1. Архитектура системы печати

Процесс печати состоит из нескольких компонентов.

Очередь печати (spooler), собирающая задания и устанавливающая их очередность. Аббревиатура "spool" образована от Simultaneous Peripheral Operation Online. Теперь эта аббревиатура стала термином.

Пользовательские утилиты (интерфейс командной строки и/или графический пользовательский интерфейс), которые отправляют задания в очередь печати и запросы системе об этих заданиях (приостанавливая или завершая их работу), удаляют задания или изменяют их очередность, а также конфигурируют другие части системы.

Внутренние интерфейсы (back end), которые обмениваются информацией с устройствами печати. (Обычно они не видны и скрыты под панелями.)

Сетевые протоколы, обеспечивающие связь с очередями печати и передачу заданий.

Для управления печатью полезно понимать, как распределены роли между частями системы. К сожалению, существует слишком много вариантов такого распределения.

## Основные системы печати

Каждая операционная система, описанная в книге, имеет подсистему печати, принадлежащую одному из трех семейств: System V, BSD или CUPS. Мы расскажем, как возникли эти имена. Это не только интересно с исторической точки зрения, но и полезно. Программное обеспечение для печати не всегда связано с линейкой операционных систем. Например, операционная система AIX, которая изначально была разработана на основе подсистемы System V.0 с расширениями V.2, использует подсистему печати BSD.

Существуют самостоятельные системы печати, которые можно инсталлировать отдельно (как, например, печально известная подсистема LPRng), но функции печати в этом случае часто оказываются перемешаны с другими частями операционной системы и заставить их работать бывает трудно. Инсталлирование собственных систем печати напоминает инсталлирование своей оболочки: система UNIX позволяет это сделать, и у вас могут быть веские причины для этого, но вы оказываетесь предоставлены сами себе. В этой книге мы рассматриваем только программное обеспечение, которое входит в дистрибутивные пакеты по умолчанию.

Для того чтобы увидеть, какое программное обеспечение для печати встроено в вашу операционную систему, достаточно посмотреть на очереди печати. Система CUPS имеет демон **cupsd**, BSD — **lpd**, а System V — **lpshed**, так что команда **which cupsd lpd lpshed** сообщит вам, что есть в вашем распоряжении.

## Очереди печати

Каждая система имеет очередь печати: часть программного обеспечения, которая получает задания на печать, хранит их, присваивает им приоритеты и последовательно посылает их на один или несколько принтеров. Иногда очередь называют демоном или сервером печати.

В некоторых принтерах (обычно сложных моделей) существуют собственные внутренние очереди печати. Если вы попросили вашего демона печати очистить очередь заданий на печать и проблема не исчезла, проверьте, нет ли заданий во внутренней очереди принтера. Для того чтобы очистить внутреннюю очередь заданий, принтер необходимо выключить и запустить снова.

Очереди печати **lpd** (BSD) и **lpshed** (SysV) — это автономные демоны, специально предназначенные для печати. Приложения, работающие в системе, либо обмениваются информацией с этими серверами, либо читают и записывают ее в файлы очереди печати или файлы конфигурации, размещенные в "хорошо известных" местах, например в каталоге **/var/spool** или **/etc**.

# 26.2. Система печати CUPS

Серверы CUPS — это веб-серверы, а клиенты CUPS — это веб-клиенты. Клиентами CUPS могут быть как команды (наподобие **lpr** и **lpq**), так и приложения с графическим пользовательским интерфейсом (наподобие **kprinter**). По сути, все они являются веб-приложениями, даже если просто передают информацию демону CUPS в локальной системе. Серверы CUPS могут также функционировать как клиенты других серверов CUPS

Сервер CUPS предоставляет полнофункциональный веб-интерфейс на порту 631. Для администраторов веб-браузер обычно является самым удобным способом для управления системой: достаточно просто перейти на адрес http://*printhost*:631. Если вам необходимо секретное соединение с демоном (и ваша система позволяет это), используйте порт https://*printhost*:433. Для управления системой сценарии могут использовать дискретные команды, и пользователи могут обращаться к ним с помощью интерфейсов GNOME или KDE. Эти способы эквивалентны.

В основе всех взаимодействий между серверами CUPS и клиентами лежит протокол HTTP, точнее, протокол печати в Интернете IPP (Internet Printing Protocol), его улучшенная версия. Клиенты посылают задания с помощью операции POST протокола HTTP/IPP и запрашивают их статус с помощью команды GET из того же протокола. Файлы конфигурации CUPS также выглядят подозрительно похожими на файлы конфигурации веб-сервера Apache.

## Интерфейсы для системы печати

Система CUPS является достаточно современной, так что большинство ее функций можно выполнить из графического пользовательского интерфейса, а администрирование часто осуществляется с помощью веб-браузера. Системный администратор (и, возможно, некоторые из консервативных пользователей) может также использовать команды оболочки. В системе CUPS есть аналоги многих команд оболочки из классических систем печати BSD и System V. К сожалению, система CUPS не эмулирует все, без исключения, свойства этих систем. Иногда она эмулирует старые интерфейсы *слишком* хорошо;

например, вместо того чтобы выдать пользователю краткую информативную справку, команды **lpr --help** и **lp -help** просто распечатывают сообщения об ошибках.

И все же многие старые сценарии, использующие эти команды, просто отлично работают в системе CUPS. Думайте об отсутствующих функциях как о возможностях: если вы хотите сохранить мир во всем мире и обеспечить оптимальность по Парето, пишите свою программу (а если вы используете старую систему, используйте существующий код).

Вот как можно было бы распечатать файлы **foo.pdf** и **/tmp/testprint.pdf**.

```
$ lpr foo.pdf /tmp/testprint.ps
```

Команда **lpr** передает копии файлов серверу CUPS **cupsd**, который сохраняет их в очереди на печать (print queue). Когда принтер готов, CUPS начинает обрабатывать каждый файл по очереди.

Во время печати система CUPS проверяет как документ, так и PPD-файл принтера (PostScript Printer Description — описание принтеров в PostScript), чтобы узнать, что необходимо сделать для того, чтобы документ был распечатан должным образом. (Как будет объяснено позже, PPD-файлы используются даже и для принтеров, не понимающих язык PostScript.)

Для того чтобы подготовить задание к печати на конкретном принтере, CUPS пропускает его через конвейер, состоящий из фильтров. Эти фильтры могут выполнять самые разные функции. Например, фильтр может изменять формат задания так, чтобы на каждой физической странице печаталось по два экземпляра уменьшенных в размере страниц, или преобразовывать задание из формата Postscript в PCL. Фильтр также может выполнять какую-нибудь специфическую для принтера операцию обработки, такую как инициализация принтера. Фильтр даже может преобразовывать изображения в растровый формат, превращая абстрактные инструкции, такие как "проведите линию поперек страницы", в двоичное изображение. Такая функция полезна для принтеров, которые не могут самостоятельно выполнять растеризацию или не понимают язык, на котором сформулировано задание.

Последним этапом в конвейере печати является внутренний интерфейс, который отправляет задание с узла на принтер через подходящий протокол, такой как Ethernet. Она также отправляет информацию о состоянии обратно на сервер CUPS. Просмотреть доступные внутренние интерфейсы можно при помощи такой команды.

```
$ locate backend | grep -i cups
```

Передав задание на печать, демон CUPS возвращается снова к обработке своих очередей и запросов от клиентов. Принтер приступает к работе, пытаясь распечатать переданное ему задание.

## Очередь на печать

Централизованное управление демона **cupsd** позволяет легко понять, как работают пользовательские команды. Например, команда **lpq** запрашивает с сервера CUPS информацию о состоянии задания и форматирует ее для отображения. Клиенты CUPS могут просить сервер откладывать задания, отменять их или изменять их приоритет. Они также могут переносить задания из одной очереди в другую.

Почти все изменения требуют указания номера задания, который сообщается в отчете, возвращаемом командой **lpq**. Например, чтобы удалить какое-нибудь задание, нужно просто выполнить такую команду: **lprm** *идентификатор_задания*.

Команда **lpstat -t** предоставляет хороший сводный отчет об общем состоянии сервера печати.

## Множество принтеров

Система CUPS создает для каждого принтера отдельную очередь. Клиенты командной строки принимают аргумент (обычно **-P** *принтер* или **-p** *принтер*) для указания очереди принтера. Также можно самостоятельно задать принтер, который должен использоваться по умолчанию, просто установив переменную окружения PRINTER

```
$ export PRINTER=имя_принтера
```

или указав системе CUPS использовать определенный параметр по умолчанию для данной учетной записи.

```
$ lpoptions -dимя_принтера
```

При выполнении от имени пользователя **root**, команда **lpoptions** устанавливает системные параметры по умолчанию, которые хранятся в каталоге **/etc/cups/lpoptions**. Эта команда позволяет каждому пользователю устанавливать свои параметры, которые хранятся в каталоге **~/lpoptions**. Команда **lpoptions -l** отображает список текущих опций.

## Экземпляры принтеров

Если имеется только один принтер и его нужно использовать по-разному — например, как для быстрой печати черновых вариантов, так и для качественной печати производственных документов, — система CUPS позволяет создавать для выполнения этих задач разные "экземпляры принтера".

Например, если уже есть принтер с именем Phaser_6120, команда

```
$ lpoptions -p Phaser_6120/2up -o number-up=2 -o job-sheets=standard
```

создаст экземпляр с именем Phaser_6120/2up, печатающий по две странице на листе и добавляющий титульные страницы. После создания нового экземпляра команда

```
$ lpr -P Phaser_6120/2up biglisting.ps
```

затем распечатает файл PostScript **biglisting.ps** как задание с двумя страницами на каждом листе и соответствующими титульными страницами.

## Сетевая печать

С точки зрения системы CUPS, сеть со множеством машин не очень отличается от изолированной машины. На каждом компьютере выполняется демон CUPS (**cupsd**), и все эти демоны взаимодействуют между собой.

Для того чтобы сконфигурировать демон CUPS так, чтобы он принимал задания на печать с удаленных систем, можно отредактировать файл **/etc/cups/cupsd.conf** (подробнее об этом будет рассказываться чуть позже в этой главе, в разделе "Настройка сервера сетевой печати"). Настроенные таким образом серверы CUPS по умолчанию каждые 30 секунд рассылают информацию об обслуживаемых ими принтерах. Благодаря этому имеющиеся в локальной сети компьютеры автоматически узнают о доступных им принтерах. Такую же конфигурацию можно создать, просто установив флажки в графическом пользовательском интерфейсе CUPS в вашем браузере.

Если кто-нибудь подключил новый принтер или включил в сеть ноутбук либо инсталлировал новую рабочую станцию, то с помощью команды **cupsd** можно найти и увидеть новые компоненты сети, щелкнув на кнопке Find New Printers вкладки Administration графического пользовательского интерфейса CUPS.

Сделать принтеры доступными для множества сетей или подсетей немного сложнее, потому что рассылаемые пакеты не пересекают границ подсетей. Наиболее популярным решением является выделить в каждой подсети подчиненный сервер, который будет запрашивать информацию с серверов в других подсетях и затем рассылать ее машинам в своей локальной подсети.

Предположим, необходимо, чтобы серверы `allie` (`192.168.1.5`) и `jj` (`192.168.2.14`), находящиеся в разных подсетях, были доступны пользователям в третьей подсети, `192.168.3`. Чтобы решить эту проблему, нужно просто выделить подчиненный сервер (например, `copeland`, `192.168.3.10`) и добавить в его файл **`cupsd.conf`** такие строки.

```
BrowsePoll allie
BrowsePoll jj
BrowseRelay 127.0.0.1 192.168.3.255
```

Первые две строки указывают демону **`cupsd`** подчиненного сервера запрашивать у находящихся на серверах `allie` и `jj` демонов **`cupsd`** информацию об обслуживаемых ими принтерах. Третья строка указывает серверу `copeland` рассылать получаемую информацию по своей собственной подсети.

Вам необходима более тонкая настройка? Требуется множество очередей для одного принтера и чтобы у каждой были свои параметры по умолчанию? Нужен один сервер, балансирующий нагрузку путем распределения заданий между несколькими принтерами? Необходимо иметь множество серверов, способных работать с равнозначными экземплярами одного и того же принтера? Необходимы клиенты **`lpd`** или Windows? Вариантов слишком много, чтобы их все можно было рассмотреть здесь, но CUPS имеет решение для всех этих ситуаций, и узнать о них подробнее можно в документации CUPS (см. раздел 26.13).

## Фильтры

Вместо того чтобы использовать специализированную утилиту печати для каждого принтера, система CUPS использует ряд фильтров, которые преобразовывают формат распечатываемого файла в формат, понятный для имеющегося принтера.

Схема фильтров в системе CUPS является довольно изящной. Когда система CUPS получает подлежащий распечатке файл, она определяет его тип MIME. Затем она проверяет файл PPD и выясняет, какие типы MIME может обрабатывать принтер. После этого с помощью файлов **`.convs`** она решает, какая цепочка фильтров может преобразовать один тип в другой и сколько это будет стоить. В заключение она выбирает цепочку и пропускает документ через фильтры. Последний фильтр в цепочке передает серверу пригодный для печати формат, а он, в свою очередь, передает данные на принтер либо через аппаратное устройство, либо через протокол, поддерживаемый принтером.

Для распознавания типа поступающих данных система CUPS использует правила, указанные в файле **`/etc/cups/mime.types`**. Например, правило

```
application/pdf      pdf string (0,%PDF)
```

означает следующее: "Если файл имеет расширение **`pdf`** или начинается со строки %PDF, тогда его типом MIME является **`application/pdf`**".

О том, как преобразовать один тип данных в другой, система CUPS узнает, просматривая правила в файле **`/etc/cups/mime.convs`**. Например, правило

```
application/pdf      application/postscript 33 pdftops
```

означает следующее: "Для того чтобы преобразовать файл **`application/pdf`** в файл **`application/postscript`**, нужно выполнить фильтр **`pdftops`**". Число 33 — это стои-

мость такой операции преобразования. Если выясняется, что одно и то же преобразование могут выполнить несколько цепочек фильтров, система CUPS выбирает цепочку с наименьшей общей стоимостью. (Стоимость определяется тем, кто создал файл, например производителем дистрибутивов. Мы не знаем, как. Если вы хотите уточнить этот механизм, потому что считаете, что можете сделать это лучше, значит, у вас слишком много свободного времени.)

Последним компонентом в конвейере CUPS является фильтр, который непосредственно взаимодействует с принтером. В PPD-файле принтера, не поддерживающего язык PostScript, можно увидеть строки типа

```
*cupsFilter: "application/vnd.cups-postscript 0 foomatic-rip"
```

или даже

```
*cupsFilter: "application/vnd.cups-postscript foomatic-rip"
```

Строка, заключенная в кавычки, имеет точно такой же формат, как и строка в файле **mime.convs**, но в ней указан только один тип MIME, а не два. Она сообщает о том, что фильтр **foomatic-rip** преобразовывает данные типа **application/vnd.cups-postscript** в родной формат данных принтера. Стоимость равна нулю (или вообще не указывается), потому что существует только один способ выполнить эту операцию, так что зачем притворяться, что есть еще какая-то стоимость? (PPD-файлы проекта Gutenprint для принтеров, не поддерживающих язык PostScript, немного отличаются.)

Узнать, какие фильтры доступны в системе, можно при помощи команды **locate pstops** (**pstops** — это популярный фильтр, который выполняет с PostScript-заданиями различные манипуляции, например добавляет PostScript-команды для указания количества копий. Остальные фильтры обязательно будут где-нибудь поблизости).

Для того чтобы просмотреть список доступных серверных служб (back ends), выполните команду **lpinfo -v**. Если в используемой системе нет серверной службы для необходимого сетевого протокола, возможно, она доступна в Интернете или на сайте производителя системы Linux.

## Управление сервером CUPS

Демон **cupsd** запускается во время загрузки и выполняется непрерывно. Все изученные нами разновидности системы Linux работают именно так по умолчанию.

Конфигурационный файл CUPS называется **cupsd.conf**; обычно его можно найти в каталоге **/etc/cups**. По формату он похож на конфигурационный файл Apache. Если вы умеете работать с одним из этих файлов, значит, сможете работать и с другим. Просматривать и редактировать файл **cupsd.conf** можно как с помощью текстового редактора, так и посредством графического пользовательского интерфейса CUPS. Стандартный конфигурационный файл хорошо прокомментирован. Эти комментарии и справочная страница **cupsd.conf** настолько подробные, что мы не будем повторять ту же самую информацию в книге.

Система CUPS считывает файл конфигурации только в момент запуска. Если вы внесли изменения в конфигурационный файл **cupsd.conf**, то должны запустить систему печати снова, чтобы они вступили в силу. Если изменения вносились с помощью графического пользовательского интерфейса через веб-браузер, то они вступят в силу автоматически. Для того чтобы перезапустить демон **cupsd** из командной строки, просто выполните команду **/etc/init.d/cups restart** или **/etc/init.d.cupsys restart**.

Конфигурационный файл CUPS можно редактировать вручную или с помощью графического пользовательского интерфейса. Если установлена настольная среда KDE, то

редактирование можно выполнить с помощью приложения KDE Print Manager, которое доступно через центр управления KDE. Когда мы его тестировали, он "жаловался" нам на то, что не понимает некоторые опции в создаваемых по умолчанию файлах **cups.conf**, на всех наших тестовых системах. Браузер графического пользовательского интерфейса безопаснее и, конечно, авторитетнее.

## Настройка сетевого сервера печати

Для того чтобы заставить систему CUPS принимать задания на печать из сети, нужно внести два изменения в файл **cupsd.conf**. Сначала нужно изменить

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
</Location>
```

на

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From адрес_сети
</Location>
```

В качестве параметра *адрес_сети* следует указать IP-адрес сети, из которой должны приниматься задания на печать (например, `192.168.0.0`). Далее нужно отыскать ключевое слово `BrowseAddress` и указать напротив него адрес рассылки в этой сети и порт CUPS.

```
BrowseAddress 192.168.0.255:631
```

Эти два действия заставят сервер принимать запросы с любой машины в сети и рассылать известную ему информацию об обслуживаемом им принтере всем имеющимся в сети демонам CUPS. Вот и все! После перезапуска демон **cupsd** станет сервером.

## Автоматическое конфигурирование принтера

На самом деле система CUPS может использоваться и без принтера (например, для преобразования файлов в формат PDF или формат факса), но в основном она все-таки используется для управления принтерами. В этом разделе речь пойдет о самих принтерах.

В некоторых случаях добавление принтера является обычным делом. Система CUPS старается автоматически обнаружить USB-принтеры при их подключении и выяснить, что с ними нужно делать.

Производители принтеров обычно предоставляют установочное программное обеспечение, которое автоматически (без участия пользователя) выполняет большую часть необходимых установочных операций на Windows и даже на Mac OS X (где тоже используется система CUPS). Однако нельзя рассчитывать на то, что производители позаботятся о предоставлении подобного установочного программного обеспечения для системы Linux.

Даже если установку приходится выполнять самостоятельно, процесс добавления принтера часто состоит всего лишь из подключения оборудования, установки соединения с веб-интерфейсом CUPS по адресу `localhost:631/admin` и предоставления ответов на несколько вопросов. Среды KDE и GNOME поставляются с собственными утилитами для конфигурирования принтера, которые могут использоваться вместо интерфейса CUPS.

Если кто-то другой добавляет принтер и об этом становится известно одному или более функционирующим в сети серверам CUPS, ваш сервер CUPS тоже уведомляется о появлении нового принтера. Ни добавлять этот принтер явно в локальный перечень устройств, ни копировать его PDD-файл на свою машину вам не нужно. Это магия.

## Конфигурирование сетевых принтеров

Сетевым принтерам необходима собственная конфигурация только для того, чтобы быть членами сети TCP/IP. В частности, им необходимо знать свой IP-адрес и сетевую маску. Такая информация обычно передается им одним из двух способов.

Почти все современные принтеры могут получать эту информацию по сети от сервера BOOTP или DHCP. Этот метод работает хорошо в средах со множеством гомогенных принтеров. Более подробную информацию о DHCP можно найти в разделе 14.7.

Второй способ заключается в установке статического IP-адреса при помощи консоли принтера, которая обычно состоит из ряда кнопок на передней панели и однострочной индикаторной панели. Все, что нужно сделать, — это "походить" по меню и отыскать раздел, где можно установить IP-адрес. (Если вдруг попадется команда, позволяющая распечатать меню, следует воспользоваться ею и сохранить печатную версию.)

Некоторые принтеры предоставляют доступ к виртуальной консоли через последовательный порт. Это замечательная идея, но в целом предполагает выполнение почти точно такого же количества операций, как и использование кнопок на передней панели принтера.

После завершения конфигурирования сетевые принтеры обычно имеют веб-консоль, доступную через браузер. Однако для того чтобы с ними можно было работать через эту консоль, у них должен быть IP-адрес и они уже должны функционировать в сети, когда вы к ним обратитесь, так что этот интерфейс оказывается недоступным как раз тогда, когда он вам очень нужен.

Когда принтер уже находится в сети и готов к использованию, нужно обязательно убедиться в том, что он защищен от несанкционированного доступа. О том, как это можно сделать, см. в разделе 26.12.

## Примеры конфигурирования принтеров

В качестве примеров добавим принтер с параллельным интерфейсом `groucho` и сетевой принтер `fezmo` из командной строки.

```
$ lpadmin -p groucho -E -v parallel:/dev/lp0 -m pxlcolor.ppd
$ lpadmin -p fezmo -E -v socket://192.168.0.12 -m laserjet.ppd
```

Как вы видите, интерфейс `groucho` предоставляется через порт **`/dev/lp0`**, а `femzo` — через IP-адрес `192.168.0.12`. Мы указываем каждое устройство в виде универсального идентификатора ресурса (URI) и выбираем PDD-файл из списка PDD-файлов, находящихся в каталоге **`/usr/share/cups/model`**.

Если локальный демон **`cupsd`** сконфигурирован как сетевой сервер, он сразу же сделает эти новые принтеры доступными для других клиентов в сети.

Поскольку сервер **`cupsd`** конфигурируется как сетевой, он немедленно делает новые принтеры доступными для других клиентов в сети. Перезапуск при этом не требуется.

CUPS позволяет использовать для принтеров самые разные URI-идентификаторы. Вот еще несколько примеров.

```
ipp://zoe.canary.com/ipp
lpd://riley.canary.com/ps
serial://dev/ttyS0?baud=9600+parity=even+bits=7
```

```
socket://gillian.canary.com:9100
usb://XEROX/Phaser%206120?serial=YGG210547
```

Одни URI-идентификаторы принимают параметры (например, `serial`), а другие — нет. Команда **`lpinfo -v`** отображает список устройств, которые система может видеть, и список типов URI-идентификаторов, которые понимает система CUPS.

## Создание класса принтеров

"Класс" — это группа принтеров, которые используют очередь совместно. Задания в очереди могут распечатываться на любом из этих принтеров, а именно на том, который становится доступен первым. Показанная ниже команда создает класс `haemer` и добавляет в него три принтера: `riley`, `gilly` и `zoe`.

```
$ lpadmin -p riley -c haemer
$ lpadmin -p gilly -c haemer
$ lpadmin -p zoe -c haemer
```

Обратите внимание на то, что никакого явного шага для создания класса нет; класс существует только при условии добавления в него принтеров. На самом деле интеллектуальные возможности системы CUPS этим не ограничиваются: если множеству принтеров в сети присваивается одинаковое имя, система CUPS воспринимает их как неявный класс и автоматически распределяет между ними связанную с заданиями нагрузку. Если не все принтеры расположены в одной комнате, это может оказаться неудобным.

## Отключение принтера

Если вы хотите удалить принтер или класс, это можно легко сделать при помощи команды **`lpadmin -x`**.

```
$ lpadmin -x fezmo
$ lpadmin -x haemer
```

Но как быть, если нужно только временно отключить принтер для прохождения технического осмотра, а не удалять его? В таком случае можно просто заблокировать очередь печати на входе или выходе. Если отключить "хвост" (т.е. выходную или принтерную сторону) очереди, пользователи по-прежнему смогут отправлять задания, но эти задания никогда не будут печататься. Если отключить "головную" (входную) часть очереди, те задания, которые уже находятся в очереди, будут распечатаны, а вот все попытки добавить в очередь новые задания будут отклоняться.

Команды **`cupsdisable`** и **`cupsenable`** контролируют выходную сторону очереди, а команды **`reject`** и **`accept`** — ее принимающую сторону. [1]

```
$ sudo disable groucho
$ sudo reject corbet
```

Какую же из них лучше использовать? Принимать задания на печать, которые точно не будут распечатаны в ближайшем будущем, глупо, поэтому для длительных периодов простоя лучше использовать команду **`reject`**. Для коротких перерывов, которые даже

[1] В старых версиях системы CUPS вместо команд **`cupsdisable`** и **`cupsenable`** используются команды **`disable`** и **`enable`**. К сожалению, команда **`enable`** также является встроенной командой оболочки **`bash`**, поэтому оболочка **`bash`** предполагает, что вы используете ее собственную команду **`enable`**, если не указать полный путь к команде. В этом случае выполняется версия команды **`enable`** из оболочки **`bash`**, которая включает и отключает встроенные команды этой оболочки, поэтому ее саму можно выключить, выполнив команду **`enable -n enable`**.

не должны быть заметны пользователям (например, когда нужно просто удалить застрявшую в принтере бумагу), лучше использовать команду **cupsdisable**.

Администраторы иногда просят дать им какую-нибудь подсказку, которая помогла бы им запомнить, какие команды контролируют каждый конец очереди. Предлагаем попробовать такой вариант: если система CUPS "отклоняет" (**reject**) задание, это означает, что его нельзя "предоставить". Еще один способ не путать команды — это запомнить, что приниматься (**accept**) и отклоняться (**reject**) могут только *задания на печать*, а отключаться (**disable**) и включаться (**enable**) — только *принтеры*.

Система CUPS иногда сама временно отключает принтер, с которым не может нормально работать (например, из-за того, что кто-то выдернул его кабель). Устранив проблему, следует снова активизировать очередь. Если забыть это сделать, команда **lpstat** напомнит. (Более подробную информацию по этой теме и альтернативному подходу можно найти по адресу `www.linuxprinting.org/beh.html`.)

## Другие связанные с конфигурированием задачи

Современные принтеры имеют очень много опций, которые можно конфигурировать. Система CUPS позволяет настраивать самые разные функциональные возможности с помощью ее веб-интерфейса и команд **lpadmin** и **lpoptions**. Как правило, команду **lpadmin** используют для выполнения задач на уровне системы, а команду **lpoptions** — для выполнения задач на уровне пользователя.

Команда **lpadmin** позволяет более четко задать ограничения доступа, чем команды **disable** и **reject**. Например, с ее помощью можно создавать квоты на печать и указывать, на каких именно принтерах разрешается выполнять печать тем или иным пользователям.

В табл. 26.1 перечислены все команды, которые поставляются с системой CUPS, и их источники, в которых они использовались первоначально.

**Таблица 26.1. Утилиты командной строки, поставляемые с системой CUPS, и системы, в которых они использовались первоначально**

| | Команда | Функция |
|---|---|---|
| CUPS | **cups-config**[a] | Выводит информацию об API-интерфейсе, компиляторе, каталоге и канале связи системы CUPS |
| | **cupsdconf**[a] | Представляет собой утилиту для конфигурирования |
| | **cupsdisable**[б] | Останавливает печать принтера или класса |
| | **cupsenable**[б] | Восстанавливает печать принтера или класса |
| | **lpinfo** | Показывает доступные устройства или драйверы |
| | **lpoptions** | Отображает или устанавливает опции и параметры по умолчанию принтера |
| | **lppasswd** | Добавляет, изменяет или удаляет пароли дайджеста |
| System V | **accept, reject** | Принимает или отклоняет поступающие в очередь задания |
| | **cancel** | Отменяет задания |
| | **lp** | Печатает файлы |
| | **lpadmin** | Конфигурирует принтеры и классы CUPS |
| | **lpmove** | Перемещает задание в новое место |
| | **lpstat** | Печатает информацию о состоянии CUPS |

*Окончание табл. 26.1*

| | Команда | Функция |
|---|---|---|
| BSD | lpc | Представляет собой общую программу для управления принтерами |
| | lpq | Отображает статус очереди принтера |
| | lpr | Печатает файлы |
| | lprm | Отменяет задания на печать |

[a] Не путайте эти похожие имена: **cupsdconf** — это утилита с графическим пользовательским интерфейсом (GUI), поставляемая с KDEPrint, а **cups-config** — это утилита с интерфейсом командной строки (CLI), поставляемая с CUPS.

[б] На самом деле это просто переименованные команды **disable** и **enable** из System V.

## 26.3. Печать в настольных системах

Мы уже упоминали о графическом пользовательском интерфейсе системы CUPS, который является альтернативой надстройкам для среды KDE. Этот графический интерфейс широко признан и отлично работает.

Рассмотрим теперь проблему переносимости. Возможно, вы уже сталкивались с тремя разными семействами систем печати, так зачем вносить дополнительную путаницу, заставляя администраторов разбираться с разными графическими пользовательскими интерфейсами? Если ваше руководство хочет печатать файлы из нового ноутбука Macintosh, то, возможно, вы не знаете, где надо щелкнуть, чтобы получить доступ к конфигурации новейшего пользовательского интерфейса, разработанного компанией Apple. Но если вы просмотрите порт localhost:631, то окажетесь на знакомой территории.

Впрочем, если все ваши пользователи используют конкретные настольные системы, вы можете использовать для их поддержки графический пользовательский интерфейс. В качестве примера рассмотрим KDEPrint, доминирующее приложение для системы печати KDE.

Программа KDEPrint включает средства для добавления принтеров, администрирования заданий на печать, перезапуска серверов печати и т.д. Как и все остальные инструменты системы KDE, она имеет внешний вид KDE, что делает их более понятными для пользователей KDE. (Не трудно заметить, что даже имена утилит KDE имеют определенный внешний вид. Однажды у нас кто-то спросил, является ли **ksh** приложением KDE.)

Программа KDEPrint не зависит от системы CUPS. Несмотря на то что она может выполнять все функции системы CUPS, ее можно настроить на работу с любой системой — от LPRng до универсальной внешней программы. Если по какой-то причине вы не используете систему CUPS (или, что еще хуже, вынуждены переключаться то на одну, то на другую систему печати), все равно можете использовать программу KDEPrint для управления процессом печати. Однако заранее предупреждаем, что CUPS мощнее других систем печати, так что если вы перейдете на какую-нибудь альтернативную систему печати, не исключено, что некоторые из функциональных возможностей KDEPrint вам больше не будут доступны.

Ниже перечислены основные компоненты приложения KDEPrint, о которых следует знать.

Утилита **kprinter** с графическим пользовательским интерфейсом, которая позволяет отправлять задания на печать.

Мастер Add Printer, который автоматически обнаруживает сетевые принтеры (JetDirect, IPP и SMB) и некоторые принтеры, подключаемые локально. Он также по-

зволяет добавлять и конфигурировать принтеры, которые не удается обнаружить автоматически.

Утилита Print Job Viewer, которая позволяет менять местами и отменять задания на печать, а также просматривать информацию об их состоянии.

Руководство *KDEPrint Handbook*, в котором содержится необходимая справочная информация по данной системе. Оно доступно через панель KDE Help Center, но там его иногда трудно найти. Легче всего вызвать какую-нибудь программу наподобие **kprinter** и щелкнуть на кнопке Help или же выполнить команду **konqueror help:/kdeprint**. Кроме того, документация по KDEPrint еще также доступна по адресу `printing.kde.org`.

Программа Print Manager, которая является главным средством управления графическим пользовательским интерфейсом для системы печати. Иногда ее тоже немного трудно найти. Можно попробовать поискать ее в главном меню рабочего стола, хотя ее расположение в дереве меню в каждом дистрибутиве выглядит по-разному. Еще один вариант — выполнить команду **kcmshell printmgr** или **konqueror print:/manager**.

Мастер Add Printer и программа Print Job Manager доступны либо через **kprinter**, либо KDE Print Manager (не говоря уже об URL-адресах `print:/manager` и `print:/printers` в браузере Konqueror).

Информация о пользователях KDEPrint хранится в каталоге **~/.kde**. Эти файлы имеют понятный для человека формат и их можно изменять через программу Print Manager. Если хотите, попробуйте отредактировать что-нибудь в них на свой страх и риск.

## Документы kprinter: printing

Утилита **kprinter** — это имеющая графический интерфейс версия программы **lpr**. Ее также можно использовать и из командной строки. Например, команда

```
$ kprinter --nodialog -5 -P lj4600 riley.ps gillian.pdf zoe.prn
```

эквивалентна следующей команде.

```
$ lpr -5 -P lj4600 riley.ps gillian.pdf zoe.prn
```

Ваши пользователи, скорее всего, захотят воспользоваться GUI-интерфейсом. Покажите им, как можно перетащить файлы из окна диспетчера файлов или с рабочего стола в диалоговое окно **kprinter** и затем распечатать их все сразу. Замените **lpr** на **kprinter** в появляющемся в их браузере окне печати, и у них будет окно печати с GUI-интерфейсом. Научите их устанавливать флажок Keep this dialog open after printing, и они даже не будут тратить время на повторный запуск этой программы каждый раз, когда им необходимо будет напечатать что-нибудь.

Обратите внимание на меню Print System currently in use, показывающее, что система KDEPrint сейчас не активна. Также обратите внимание на то, что когда принтер не доступен, **kprinter** предлагает выполнить печать в PDF или распечатать документ через факс. Дополнительные опции тоже заслуживают внимания: они позволяют ставить в очередь на печать свое резюме и указывать, что оно должно быть распечатано после того, как начальник уйдет домой.

## Konqueror и печать

Многие веб-браузеры распознают ряд специальных URI-адресов, которые выступают в роли шлюзов к уникальным функциональным возможностям. Вы, наверняка, пробовали вводить `about:config` и `about:mozilla` в окне Firefox. Точно так же семейство URI-адресов `print:` является в веб-браузере Konqueror шлюзом к KDEPrint.

При вводе URI-адреса `print:/` отображается перечень всех возможностей, при вводе `print:/jobs` — перечень заданий на печать, а при вводе `print:/manager` внутри Konqueror запускается программа Print Manager.

Даже если вы не работаете с системой печати CUPS непосредственно, все значительно упрощается, если помнить, что система CUPS является веб-браузером. Браузеры знают, как общаться с веб-серверами, поэтому их относительно легко оснастить функциями печати CUPS.

# 26.4. Система печати System V

Программное обеспечение System V самое старое и простое среди всех рассматриваемых нами систем печати. Оно настолько старое, что вообще не предполагает работу в сети. Большинство разработчиков, использующих его, вынуждены были внести в него много изменений. Как это обычно бывает, некоторые из этих изменений оказались полезными, а некоторые нет.

Среди изученных нами операционных систем программное обеспечение System V используют системы Solaris и HP-UX. В обоих случаях программы System V были существенно модифицированы. Ниже мы рассмотрим стандартную систему, попутно отмечая изменения, внесенные разработчиками.

## Обзор

Пользователь, желающий что-нибудь распечатать, должен использовать либо команду **lp**, либо команду, которая вызывает команду **lp** неявно. Команда **lp** заносит данные в буферный каталог (spool directory), связанный с их пунктом назначения. Демон **lpsched** определяет, когда и где следует распечатать данные, а затем выполняет интерфейсную программу, форматирующую данные и посылающую их на соответствующий принтер.

В табл. 26.2 перечислены команды системы печати System V.

**Таблица 26.2. Команды системы System V**

| | Команда | Каталог | Назначение |
|---|---|---|---|
| Общие | **accept** | **/usr/sbin** | Включает режим постановки заданий в очередь |
| | **cancel** | **/bin** | Удаляет задания на печать из очереди |
| | **disable** | **/bin** | Выключает печать заданий, находящихся в очереди |
| | **enable** | **/bin** | Включает печать заданий, находящихся в очереди |
| | **lp** | **/bin** | Ставит задания в очередь для печати |
| | **lpadmin** | **/usr/sbin** | Конфигурирует систему печати |
| | **lpmove** | **/usr/sbin** | Переносит задания из одной очереди в другую |
| | **lpsched** | **/usr/sbin** | Планирует и выполняет задания |
| | **lpshut** | **/usr/sbin** | Останавливает работу системы печати |
| | **lpstat** | **/usr/sbin** | Выводит отчет о состоянии системы печати |
| | **reject** | **/usr/sbin** | Выключает режим постановки заданий в очередь |
| | **lpfilter** | **/usr/sbin** | Управляет фильтрами печати |
| | **lpforms** | **/usr/sbin** | Управляет использованием предпечатных форм |

*Окончание табл. 26.2*

| Команда | Каталог | Назначение |
|---|---|---|
| **lpget** | **/bin** | Считывает параметры конфигурации |
| **lpset** | **/bin** | Модифицирует параметры конфигурации |
| **lpuser** | **/usr/sbin** | Управляет приоритетами заданий в очереди |
| **lpalt** | **/bin** | Модифицирует задания в очереди |
| **lpana** | **/usr/sbin** | Анализирует регистрационные записи о работе системы печати |
| **lpfence** | **/usr/sbin** | Устанавливает минимальный приоритет задания на принтере |
| **lpr** | **/bin** | Поддерживает функции печати системы BSD |

## Пункты назначения и классы

"Пункт назначения" печати имеет имя, которое может состоять максимум из 14 букв, цифр и знаков подчеркивания. Пункт назначения — это, как правило, принтер, хотя и не обязательно. Например, пунктом назначения может быть файл, в который различные пользователи хотят добавить текст. Поскольку система печати является системой массового обслуживания, команду **lp** можно использовать для предотвращения ситуаций, когда два человека одновременно пытаются добавить текст в один и тот же файл.

Каждый пункт назначения может относиться к одному или нескольким классам, но может и не принадлежать ни одному классу. Класс — это группа пунктов назначения, которые имеют одно и то же назначение. Например, если в организации два принтера стоят в одной комнате, то их можно объединить в один класс. Аналогично два принтера с одинаковыми параметрами (например, цветом, разрешением, дуплексом или скоростью) также можно отнести к одному классу. Демон **lpsched** направляет данные, предназначенные для определенного класса, на тот из двух принтеров, который свободен в данный момент. Имена классов подчиняются тем же правилам, что и имена пунктов назначения.

Как бы там ни было, термин "пункт назначения" часто используется как синоним слов "принтер" или "класс". Конкретный смысл этого термина зависит от контекста.

## Краткое описание команды lp

Это команда пользовательского уровня, которая помещает данные в очередь на печать. Команда **lp** копирует эти данные (которые могут поступать либо из именованных файлов, либо из стандартного ввода) в файл или в набор файлов в буферном каталоге. В системе HP-UX буферный каталог для пункта назначения обычно называется **/var/spool/lp/request/***пункт_назначения*, где *пункт_назначения* — это имя, под которым этот принтер или класс принтеров известны команде **lp**.

Буферные файлы называются *xxxn*, где *n* — идентификационный номер задания, присвоенный командой **lp**, а часть имени *xxx* зависит от конкретной системы. Это имя файла позволяет идентифицировать задание как пользователю, так и системе печати для внутренних целей. Далее мы будем использовать это имя в качестве идентификатора задания.

Запрос **lp -d** ставит входную информацию в очередь на печать в конкретном пункте назначения. Если опция **-d** не указана, команда **lp** использует в качестве имени пункта назначения содержимое переменной среды LPDEST. Если же эта переменная не установлена, команда **lp** ставит данные в очередь для вывода в пункте назначения, используемом по умолчанию, который администратор может установить с помощью команды **lpadmin -d**.

SOLARIS В системе Solaris если пункт назначения по умолчанию не задан командой **lpadmin -d**, то команда **lp** ищет файл **~/.printers**, файл **/etc/printers.conf** и, наконец, службу Federated Naming Service как пункт назначения, заданный по умолчанию.

## Команды lpsched и lpshut: начало и конец печати

Демон **lpsched** посылает файлы, помещенные в буферный каталог командой **lp**, на соответствующее устройство по мере его освобождения. Он заносит в файл информацию о каждом обработанном файле и возникших ошибках.

В операционной системе Solaris регистрационный файл по умолчанию имеет имя **/var/lp/logs/lpsched**. Операционная система HP-UX хранит регистрационный файл в каталоге **/usr/spool/lp/log**. Когда демон **lpsched** начинает работу (обычно во время загрузки), он переносит старый регистрационный журнал в каталог **oldlog** и начинает новый.

Журнальный файл выглядит примерно так.

```
***** LP LOG:  Jul    6 12:05 *****
prl-107        garth     prl    Jul    6    12:10
pr-112         scott     prl    Jul    6    12:22
pr-117         evi       pr2    Jul    6    12:22
prl-118        garth     prl    Jul    6    12:25
prl-119        garth     prl    Jul    6    13:38
pr-132         evi       prl    Jul    6    13:42
```

В первой колонке указывается идентификатор задания, во второй — имя пользователя, пославшего задание, в третьей колонке задается принтер, на который было послано задание, а в последней — время постановки задания в очередь.

В системе HP-UX, взятой для этого примера, имеется два принтера — `prl` и `pr2`, которые относятся к классу `pr`. Пользователь `garth` всегда указывал конкретный принтер `prl`, поэтому его задания постоянно посылались именно на него. Пользователи `scott` и `evi`, со своей стороны, указывали класс `pr`, поэтому их задания посылались на первый свободный принтер этого класса.

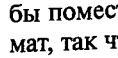
Для того чтобы по какой-то причине прервать выполнение демона **lpsched**, необходимо выполнить команду **lpshut** как суперпользователь или как пользователь **lp**. Когда демон **lpsched** не работает, никакие задания не печатаются, хотя можно продолжать постановку заданий в очередь с помощью команды **lp**. Задание, которое в момент останова демона печаталось, после его перезапуска будет напечатано сначала.

Команда **lpsched** создает файл **/usr/spool/lp/SCHEDLOCK**, который предназначен для индикации ее работы. Если попытаться запустить другую копию демона **lpsched**, вы получите сообщение, что такой файл уже есть, и запуск демона будет прерван. Если выполнение демона **lpsched** было прервано не с помощью команды **lpshut**, а другими средствами, то перед перезапуском демона файл **SCHEDLOCK** обязательно нужно удалить вручную.

## Команда lpadmin: конфигурирование среды печати

Команда **lpadmin** информирует систему печати о конфигурации принтеров. Она присваивает принтерам имена, создает классы и задания для принтера, используемого по умолчанию. На самом деле эта команда только создает текстовые файлы и модифицирует текстовые файлы в каталоге **/usr/spool/lp**.

Несмотря на то что эти файлы конфигурации можно читать, рядом с ними неплохо бы поместить табличку: "Руками не трогать!" Эти файлы имеют жестко заданный формат, так что их легко испортить.



Разработчики операционной системы Solaris попытались использовать файл описания принтера, принятый в системе BSD, чтобы облегчить конфигурирование системы. Однако на деле это привело к разбиению конфигурации на два файла: **/etc/printers.conf** и **/etc/lp**.

Операционная система Solaris требует, чтобы во время выполнения большинства административных команд работал демон **lpsched**. С другой стороны, в операционной системе HP-UX большинство команд демона **lpadmin** не выполняется, если работает демон **lpsched**. В этом случае демон **lpsched** должен быть остановлен командой **lpshut** и лишь после этого можно использовать демон **lpadmin**. (Возможно, эти разработчики противятся переходу на систему CUPS, потому считают, что она будет одинаковой во всех операционных системах и лишит мир богатого разнообразия?)

Перед тем как система начнет вывод заданий на принтер, необходимо сообщить ей о том, что этот принтер существует. Для того чтобы добавить новый принтер, необходимо выполнить следующую команду.

```
$ sudo lpadmin -pпринтер -vустройство { -eпринтер | -mмодель | -iинтерфейс }
    [ -cкласс ... ]   [{ -l | -h }]
```

Здесь *принтер* — это имя принтера (как внутреннее в системе организации очередей, так и на уровне пользовательских команд), а *устройство* — это файл устройства, с которым связан принтер. В качестве аргумента *устройство* обычно используется специальный файл в каталоге **/dev**, хотя в принципе можно использовать любой файл.

Флаги **-e**, **-m** и **-i** сообщают системе организации очередей, какую интерфейсную программу принтера она должна использовать. Интерфейсная программа отвечает за фактическое форматирование заданий, поступающих непосредственно на принтер. Интерфейсные программы в системе System V аналогичны фильтрам системы CUPS. Более подробную информацию о фильтрах можно найти в разделе 26.2.

Интерфейсную программу можно задать одним из трех способов.

- **-e***принтер*. В этом случае *принтер* — это имя существующего принтера. Этот метод задания интерфейсной программы полезен в том случае, если вы добавляете принтер, идентичный одному из уже имеющихся. Команда **lpadmin** создает копию интерфейсной программы с новым именем пункта назначения.
- **-m***модель*. В этом случае *модель* — это тип устройства, для которого в системе имеется стандартная интерфейсная программа. Информация о том, какие модели поддерживает система, содержится в каталоге **/usr/spool/lp/model**. В этом случае команда **lpadmin** создает копию файла **/usr/spool/lp/model/***модель* в каталоге **/usr/spool/lp/interface/***пункт_назначения*.
- **-i***интерфейс*. В этом случае интерфейс является полным именем пути для программы, которая будет использоваться как интерфейсный сценарий. Большинство версий команды **lpadmin** делает копию интерфейсной программы, поэтому если вы захотите изменить ее после запуска команды **lpadmin**, необходимо будет заменить копию, а не свой оригинал.

Система HP-UX позволяет определять программы, которые возвращают информацию о состоянии очереди и отменяют задания печати. Их можно задавать как интерфейсные сценарии, но при этом используются другие префиксы

параметров (префиксы **-ocm** и **-osm** задают соответственно сценарий отмены и сценарий выдачи статуса).

Кроме того, при вызове команды **lpadmin** можно задавать следующие параметры.

- **-p***принтер*. Этот параметр сообщает команде **lpadmin**, с какими принтерами вы хотите работать. Для того чтобы изменить принтер, необходимо комбинировать этот флаг с другими параметрами.
- **-c***класс*. Этот параметр задает имя *класса*, в который следует включить принтер. Для одного принтера можно задать сколько угодно классов. Если указан несуществующий класс, он будет создан. Длина имени класса не должна превышать 14 символов.
- **-x***принтер*. Этот параметр удаляет принтер из системы печати. Если *принтер* является представителем целого класса принтеров, удаляется весь класс. Ни принтер, ни класс не могут быть удалены, если они уже выполняют задание из очереди на печать. Если поставленные в очередь задания мешают удалить принтер, используйте команду **reject**, чтобы предотвратить загрузку в очередь печати новых заданий, а также команды **lpmove** и **cancel** для удаления существующих заданий. Если команда **lpadmin -x** по-прежнему не удаляет принтер, последуйте совету, приведенному в конце этого раздела.
- **-r***класс*. Этот параметр удаляет принтер из заданного *класса*, а не из системы печати. Если указанный принтер является единственным представителем класса, удаляется сам класс.

Команда **lp** не принимает запросы к новому принтеру, пока ей не поступит такое указание от команды **accept**.

Команды печати в системе System V вместо одного пункта назначения часто получают список аргументов, заключенных в кавычки и разделенных запятыми. Например, команда

```
$ sudo /usr/sbin/lpadmin -p"howler-lw,ralphie-lw" -ceng-printers
```

добавляет принтеры `howler-lw` и `ralphie-lw` в класс `eng-printers`. Флаги, которые используются при вызове программы **lpadmin**, перечислены в табл. 26.3.

**Таблица 26.3. Флаги команды lpadmin**

| Флаг | Функция |
|---|---|
| **-p***принтер* | Задает принтер, к которому относятся все последующие параметры |
| **-d***пункт_назначения* | Назначает пункт назначения, используемый по умолчанию |
| **-x***пункт_назначения* | Удаляет пункт назначения из системы печати |
| **-c***класс* | Добавляет принтер в класс |
| **-r***класс* | Удаляет принтер из класса |
| **-e***пункт_назначения* | Копирует интерфейсную программу для принтера |
| **-i***интерфейс* | Задает интерфейс в качестве интерфейсной программы для принтера |
| **-m***модель* | Назначает модель интерфейсной программой для принтера |
| **-h** | Определяет, подключен ли принтер физически |
| **-v***файл* | Определяет полный путь к файлу устройства, связанному с принтером |
| **-D***"описание"* | Задает строку описания принтера |
| **-L***"местоположение"* | Задает текстуальное описание местоположения принтера |

## Примеры использования команды lpadmin

Рассмотрим несколько примеров использования команды **lpadmin**.

```
$ sudo lpadmin -phowler-lw -v/dev/tty06 -mPostScript -cpr
```

Эта команда сообщает системе печати, что принтер с именем `howler-lw` назначен файлу устройства **/dev/tty06** и что он должен относиться к классу `pr`, а также о том, что следует использовать интерфейсную программу для принтеров PostScript. Команда **lpadmin** сама создает буферный каталог.

Команда

```
$ sudo lpadmin -dpr
```

задает используемый по умолчанию системный пункт назначения для класса (или принтера) `pr`.

Команда

```
$ sudo lpadmin -phowler-lw -l"Conference room"
```

задает строку описания принтера `howler-lw`.

Команда

```
$ sudo lpadmin -phowler-lw -rpr -cfast
```

удаляет принтер `howler-lw` из класса `pr` и добавляет его в класс `fast`.

Команда

```
$ sudo lpadmin -xhowler-lw
```

полностью удаляет принтер `howler-lw` из системы печати.

## Команда lpstat: получение информации о состоянии системы печати

Команда **lpstat** выдает информацию о состоянии системы печати. Если не указаны аргументы, эта команда сообщает статус всех заданий, которые посланы на печать задавшим ее пользователем. Если задан аргумент **-p**, то она выдает информацию о состоянии конкретного принтера. Например, команда

```
$ lpstat -phowler-lw

howler-lw is now printing pr-125. enabled since Jul 4 12:25
```

выводит информацию о состоянии принтера `howler-lw`. Статус демона **lpsched** можно определить с помощью команды **lpstat -r**.

```
$ lpstat -r

scheduler is running
```

В данном случае она сообщает, что все нормально. Флаги команды **lpstat** перечислены в табл. 26.4.

**Таблица 26.4. Флаги команды lpstat**

| Флаг | Функция |
|---|---|
| **-r** | Выводит информацию о состоянии демона lpsched |
| **-d** | Выводит информацию о пункте назначения, используемом по умолчанию |
| **-с***класс* | Выводит список членов класса |

*Окончание табл. 26.4*

| Флаг | Функция |
|---|---|
| **-о***аргумент* | Выводит информацию о состоянии запросов на вывод аргумента |
| **-u***пользователь* | Выводит информацию о состоянии заданий, посланных на печать пользователем |
| **-р***принтер* | Выводит информацию о состоянии принтера |
| **-v***принтер* | Выводит информацию об устройстве вывода, связанном с принтером |
| **-а***лункт_назначения* | Выводит информацию о режиме постановки заданий в очередь пункта назначения |
| **-s** | Выводит краткую информацию о состоянии системы печати |
| **-t** | Выводит полную информацию о состоянии системы печати |

## Команда cancel: удаление заданий печати

Команда **cancel** удаляет из очереди задания, которые находятся в ней или печатаются в данный момент. В этой команде можно указывать либо идентификатор задания (он определяется с помощью команды **lpstat**), либо имя принтера (тогда отменяется задание, которое выводится на печать в данный момент).

🕮 Более подробную информацию о выполнении программ с установленным битом setid можно найти в разделе 6.5.

Для команды **cancel** обычно установлены такие права доступа: владелец — псевдопользователь `lp`, группа — `bin`, а код прав доступа — `6775`, поэтому любой пользователь может выполнить ее и отменить явно фиктивное задание. Если задание отменяет тот, кто его не посылал, владельцу задания посылается сообщение по электронной почте. Если пользователи будут злоупотреблять этой возможностью, установите такой режим, чтобы команда **cancel** не устанавливала бит setuid.

## Команды accept и reject: управление очередью печати

Если принтер недоступен длительное время (например, из-за неисправности), буферизацию заданий, посылаемых на это устройство, следует отключить, чтобы задания пользователей, не знающих о ситуации, не переполнили очередь. Отключение буферизации производится с помощью команды **reject**. Например, следующая команда заставляет демон **lp** отменить запросы к принтеру `howler-lw`.

```
$ sudo reject -r"howler-lw will be down until Tuesday" howler-lw
```

Флаг **-r** является необязательным, однако это отличный способ сообщить пользователям, почему принтер отклоняет запросы. При попытке вывести файл на печать, команда **lp** выведет следующее сообщение.

```
$ lp -dhowler-lw myfile
lp: cannot accept requests for destination "howler-lw"
    -- howler-lw will be down until Tuesday
```

Команда **accept** *принтер* указывает демону **lp** начать прием запросов к *принтеру*. Команду **accept** необходимо выполнять один раз для каждого нового принтера, добавляемого командой **lpadmin**, потому что новые принтеры по умолчанию конфигурируются так, чтобы запросы к ним отклонялись. Для того чтобы обеспечить возможность буферизации для целого класса, в командах **accept** и **reject** вместо имени принтера можно указать имя класса.

## Команды enable и disable: управление печатью

Команда **disable** отдает демону **lpsched** указание прекратить посылку заданий на конкретный принтер. В отличие от команды **reject**, команда **disable** позволяет утилите **lp** продолжать ставить задания в очередь на этот принтер. Однако стоящие в очереди задания не будут выводиться на печать, пока принтер не будет повторно включен командой **enable**. Команда **disable** обычно не прерывает печать текущего задания; это можно сделать с помощью флага **-c**. Как и команда **reject**, команда **disable** поддерживает флаг **-r**, который позволяет объяснить причину отключения принтера. Например, команда

```
$ sudo disable -r"Being cleaned, back in 5 minutes" howler-lw
```

выключает печать на принтере `howler-lw`. Для возобновления печати необходимо выполнить следующую команду.

```
$ sudo enable howler-lw
```

## Команда lpmove: перемещение заданий

Иногда необходимо переместить задания, стоящие в очереди к одному принтеру или классу принтеров, на другой принтер. Это можно сделать с помощью команды **lpmove**, которая получает список идентификаторов заданий и имя нового принтера. Например, команда

```
$ sudo lpmove howler-lw-324 howler-lw-325 anchor-lj
```

перемещает задания с номерами 324 и 325 из очереди к принтеру `howler-lw` в очередь к принтеру `anchor-lj`. В качестве источника в команде **lpmove** можно задать принтер или класс принтеров. Например, команда

```
$ sudo lpmove howler-lw anchor-lj
```

перемещает все задания из очереди к принтеру `howler-lw` в очередь к принтеру `anchor-lj`. В этом случае возникает побочный эффект: к исходному принтеру применяется команда **reject**. В предыдущем примере демон **lp** перестанет принимать запросы для принтера `howler-lw`.

Операционная система HP-UX спроектирована так, чтобы команду **lpmove** нельзя было использовать, если выполняется демон **lpsched**. Вначале следует выполнить команду **lpshut**.

## Интерфейсные программы

Интерфейсная программа извлекает данные из файла, заданного демоном **lpsched**, форматирует их и посылает в свой стандартный поток вывода. Кроме того, интерфейсная программа отвечает за правильную установку режимов на устройстве вывода и за формирование заголовков и завершителей, если они необходимы. Интерфейсные программы, как правило, представляют собой сценарии оболочки, но могут быть и исполняемыми двоичными файлами.

Демон **lpsched** вызывает интерфейсные программы со следующими аргументами.

```
идентификатор_задания пользователь заголовок копии опции файл ...
```

Здесь использованы следующие аргументы.

- *идентификатор_задания* — назначается программой lp;

- *пользователь* — владелец задания;
- *заголовок* — необязательный заголовок (задается пользователем);
- *копии* — количество печатных экземпляров документа;
- *опции* — указываемые пользователем параметры;
- *файл* — полный путь к файлу, который следует распечатать.

Перечисленные аргументы должны указываться при каждом выполнении интерфейсной программы, но некоторые из них могут быть пустыми строками. Стандартным входным потоком для интерфейсной программы является файл **/dev/null**, а стандартные потоки вывода и ошибок направляются на устройство, заданное командой **lpadmin -v**.

В отличие от систем CUPS и BSD, в которых каждому формату файлов должна соответствовать собственная интерфейсная программа, в системе System V интерфейсные программы предназначены для обработки данных всех типов, которые поддерживает принтер. (Получив данные неизвестного типа, такие программы прекращают работу.) По этой причине интерфейсные программы обычно представляют собой сценарии оболочки, которые просто обрабатывают аргументы, а для выполнения реального форматирования вызывают другие программы.

По существу, интерфейсный сценарий в системе печати отвечает за весь этап вывода данных. Это облегчает настройку, но вместе с тем приводит к тому, что разные принтеры работают совершенно по-разному.

Интерфейсы необходимы, если вы собираетесь печатать что-либо помимо текста, подготовленного специально для печати на PostScript-принтере. В настоящее время почти все принтеры используют интерфейсы. Струйные принтеры обязательно используют интерфейсы для преобразования задания на печать в свой собственный формат.

Если интерфейсная программа закончила свою работу успешно, то ее код завершения равен 0; при ошибке он равен целому числу от 1 до 127. Если задание не выведено на печать, интерфейсный сценарий должен предпринять повторную попытку. Если же ошибка оказалась серьезной, интерфейсная программа должна отключить принтер с помощью команды **disable**. Если проблемы с печатью возникают постоянно, причину, скорее всего, следует искать именно в интерфейсном сценарии.

## Что делать, если система печати вышла из строя

Иногда попытки задать или отменить конфигурацию принтера приводят к тому, что система печати выходит из строя. Файлы конфигурации, хранящие информацию о принтере, имеют сложную структуру и очень уязвимы. Один неверный символ может привести принтер в нерабочее состояние.

Если каким-то образом вы создали принтер, который вывел систему печати из строя, то лучше всего полностью удалить его и начать все сначала. Иногда система запутывается настолько, что принтер не удается даже удалить.

В такой ситуации может спасти только метод грубой силы. В следующем примере мы попытаемся удалить принтер `hoser`. (Если имя `hoser` не уникально, этого делать нельзя.)

```
$ sudo lpshut
$ sudo lpadmin -xhoser
$ sudo find /usr/spool/lp -name hoser -exec rm -rf {} ; # удаляем задания
$ lpsched
$ lpstat -t
```

Первые две команды отключают систему буферизации и пытаются удалить принтер. Если система запуталась, команда **lpadmin -x** может не сработать. Команда **find** удаляет все интерфейсные программы и буферные каталоги, соответствующие данному принтеру. Демон **lpsched** перезапускает систему буферизации, а команда **lpstat** используется для проверки того, все ли ссылки на принтер `hoser` в системе печати уничтожены.

# 26.5. Печать BSD и AIX



Мы могли бы назвать этот раздел просто "Печать в системе AIX", поскольку AIX — единственная среди рассмотренных нами операционных систем, которая все еще поддерживает систему печати BSD. Однако мы сохранили традиционное имя этой системы, поскольку ее все же можно обнаружить в некоторых операционных системах, помимо AIX.

Система печати BSD была разработана специально для использования старых игольчатых принтеров, но хорошая конструкция этой системы позволяет легко приспосабливать ее для поддержки большинства современных принтеров. Сетевая часть системы печати BSD также хорошо масштабируется для использования в крупных неоднородных сетях и позволяет многим компьютерам совместно использовать принтеры. Программа буферизации печати **lpd**, относящаяся к системе BSD, получила широкое распространение и применяется во многих сетевых принтерах.

## Обзор архитектуры системы BSD

В системе BSD доступом к принтерам управляет демон **lpd**. Он принимает задания на печать от пользователей и других (удаленных) демонов **lpd**, обрабатывает их и посылает на свободный принтер. Для того чтобы выполнить эти действия, демон **lpd** считывает данные о конфигурации принтера из файла **/etc/printcap** — системной базы данных, содержащей информацию о принтерах.

Для отправки своих заданий демону **lpd** пользователи вызывают программу **lpr**. Эти два процесса взаимодействуют через сокет домена системы UNIX **/dev/printer**.

Для того чтобы определить, какому принтеру послать задание, демон **lpr** сначала анализирует командную строку. Если пользователь указал аргумент **-P***принтер*, то пунктом назначения становится *принтер*. В противном случае демон **lpr** проверяет, определена ли переменная окружения PRINTER. Если эта переменная определена, демон **lpr** использует ее значение. Во всех остальных случаях демон **lpr** передает задание на общесистемный принтер, заданный по умолчанию. Его имя должно начинаться с букв lp. Если таких символов нет, то используется первый принтер, указанный в файле **/etc/printcap**. Почти все команды, связанные с принтерами, включая команды **lpq** и **lprm**, понимают переменную окружения PRINTER и аргумент **-P**.

Как только программа **lpr** узнает, на какой принтер направляется текущее задание, она начинает искать его в базе данных о принтерах системы **/etc/printcap**. Прочитав файл **printcap**, программа **lpr** узнает, где хранится задание на печать для данного принтера. Этот буферный каталог обычно называется **/var/spool/lpd/***имя_принтера*.

Для каждого задания программа **lpr** создает в буферном каталоге два файла. Имя первого состоит из букв **cf** (control file) и числа, идентифицирующего задание. Этот файл содержит справочную информацию и информацию об обработке задания, например сведения о пользователе, который его послал. Числовая часть имени файла может состоять максимум из трех цифр, поэтому если в очереди более 999 заданий, система

печати дает сбой. Имя второго файла начинается с букв **df** (data file) и заканчивается тем же числом. Этот файл содержит данные, которые должны быть выведены на печать. Поместив этот файл в очередь печати, программа **lpr** уведомляет демона **lpd** о существовании задания.

Получив это уведомление, демон **lpd** обращается к файлу **printcap** и выясняет, является ли пункт назначения локальным или удаленным. Если принтер подключен локально, демон **lpd** проверяет наличие демона печати, обрабатывающего соответствующую очередь, и при необходимости создает его (копируя самого себя).

Если соответствующий принтер подключен к другому компьютеру, демон **lpd** устанавливает соединение с демоном **lpd** удаленного компьютера и пересылает туда файл данных и управляющий файл. Затем демон **lpd** удаляет локальные копии этих файлов.

Планирование заданий печати осуществляется по принципу "первым пришел—первым вышел" (FIFO), но системный администратор может при желании изменить порядок печати с помощью команды **lpc**. К сожалению, не существует способа постоянно указывать системе печати о том, чтобы она отдавала предпочтение заданиям, направленным на печать конкретным пользователем или компьютером.

Когда задание готово к печати, демон **lpd** создает ряд каналов UNIX между буферным файлом и печатающим устройством для передачи данных, подлежащих печати. Посредине этой цепочки демон **lpd** устанавливает фильтрующий процесс, в задачи которого входит просмотр и, возможно, редактирование содержимого потока данных, прежде чем они поступят на принтер.

Фильтрующие процессы могут выполнять различные преобразования данных или вообще ничего не делать. Их главное назначение — обеспечивать форматирование и поддержку любых связанных с устройствами протоколов, которые могут понадобиться для работы с конкретным принтером. Фильтр, заданный по умолчанию для конкретного принтера, указывается в файле **/etc/printcap**, но с помощью команды **lpr** для принтера можно назначить другой фильтр.

## Управление средой печати

Для ежедневного сопровождения системы печати достаточно знать всего три команды: **lpq**, **lprm** и **lpc**. Команда **lpq** отображает содержимое очереди заданий конкретного принтера. Команда **lprm** позволяет удалять одно или несколько заданий, при этом из системы печати удаляются соответствующие файлы данных и все ссылки на них. Обе команды доступны для пользователей и могут работать по сети, хотя только суперпользователь может удалить чье-то еще задание.

Команда **lpc** позволяет вносить изменения в среду печати, например отключать принтеры и переупорядочивать задания, стоящие в очереди. Некоторые ее функции доступны пользователям, но, в основном, это инструмент администратора. В табл. 26.5 перечислен ряд других команд и демонов системы BSD.

**Таблица 26.5. Команды системы BSD**

| Команда | Каталог | Назначение |
|---|---|---|
| lpc | /usr/sbin | Позволяет управлять принтером или очередью на печать |
| lpd | /usr/sbin | Планирует обработку заданий и направляет их на печать |
| lpq | /usr/bin | Отображает содержимое очереди на печать и статус заданий в ней |
| lpr | /usr/bin | Ставит задания в очередь на печать |

*Окончание табл. 26.5*

| Команда | Каталог | Назначение |
|---|---|---|
| lprm | /usr/bin | Отменяет задание |
| lptest | /usr/bin | Генерирует тестовую ASCII-страницу |

## Демон lpd: буферизация заданий на печать

Демон **lpd**, запущенный с флагом **-l**, регистрирует задания на печать в системе Syslog от имени средства `lpr`. При отсутствии этого флага демон регистрирует только системные ошибки.

Контроль доступа осуществляется на уровне отдельных компьютеров. Система BSD не позволяет контролировать конкретных удаленных пользователей. Только компьютерам, указанным в файле **/etc/hosts.equiv** или **/etc/hosts.lpd**, разрешено ставить задания в очередь. По соображениям безопасности, использовать файл **hosts.equiv** не рекомендуется; применяйте лучше файл **hosts.lpd**.

## Команда lpr: выдача заданий на печать

Это единственная команда в системе BSD, которая может помещать задания в очередь на печать. Все остальные программы печати файлов (например, **enscript** и браузер) вынуждены вызывать эту команду.

Флаг **-#***число* позволяет напечатать указанное количество копий документа, а флаг **-h** запрещает печать титульной страницы. Например, для того чтобы напечатать две копии файла **thesis** на принтере `howler-lw`, следует выполнить следующую команду.

```
$ lpr -Phowler-lw -#2 thesis
```

## Команда lpq: просмотр очереди печати

Рассматриваемая команда обычно вызывается с опцией **-P** для выбора принтера, но есть и опция **-l**, позволяющая получить более детальный отчет. Выходные данные команды выглядят примерно так.

```
$ lpq
anchor-lj is ready and printing
Rank    Owner    Job  Files              Total Size
active  garth    314  domain.2x1.ps      298778 bytes
1st     kingery  286  standard input     17691 bytes
2nd     evi      12   appendices         828 bytes
...
```

Выходные строки всегда расположены по порядку: активное задание указывается первым, а последнее — в самом низу. Если первое задание обозначено как `1st`, а не как `active`, значит, демон печати не запущен.

Во втором столбце приводится имя пользователя, который послал задание на печать. В третьем столбце указывается идентификатор задания. Его необходимо знать, если задание должно обрабатываться с помощью команд **lprm** или **lpc**. В четвертой колонке перечислены файлы, посылаемые на печать (они были заданы в строке вызова команды **lpr**). Если данные для печати поступили через канал (как, например, первое и пятое задания в нашем примере), в этой колонке будет стоять запись `standard input`. В пятой колонке указывается размер задания. Это значение соответствует размеру задания до его передачи программе-фильтру и не дает информации о том, сколько страниц займет задание и как долго оно будет печататься.

## Команда lprm: удаление заданий на печать

Самая распространенная форма команды **lprm** — **lprm** *номер*, где *номер* — это идентификатор задания согласно выходным данным команды **lpq**. Команда **lprm** *пользователь* удаляет все задания, принадлежащие указанному пользователю. Команда **lprm** без аргументов уничтожает активное задание. Команда **lprm** - (это дефис) удаляет все задания, посланные на печать текущим пользователем. Если это пользователь **root**, уничтожаются все задания, стоящие в очереди. Рядовой пользователь не может удалять чужие задания, в отличие от суперпользователя.

В противоположность ожидаемому, команда **lprm** делает ошибки молча, а при успешном завершении выдает результат. Если вы не видите результат вроде строк

```
dfA621xinet dequeued
cfA621xinet dequeued
```

после запуска команды **lprm**, значит, команда не была выполнена, т.е. она либо не смогла удалить задание, либо вы ее неправильно вызвали.

Система печати записывает узлы, откуда поступило задание, и пользователя, который его послал. Эта информация учитывается командой **lprm**. Таким образом, пользователь `garth@boulder` не эквивалентен пользователю `garth@sigi`, и ни один из них не имеет права удалять задания другого.

Попытка удалить с помощью команды **lprm** активное задание может вызвать проблемы на некоторых принтерах. Фильтрующий процесс, связанный с заданием, надлежащим образом не уведомляется о его уничтожении, вследствие чего вся система "зависает", а фильтрующий процесс продолжает блокировать порт принтера, не допуская к нему другие процессы.

Единственный способ исправить положение — определить идентификаторы фильтрующих процессов с помощью команды **ps** и вручную уничтожить их. Команда **lpc** в такой ситуации бесполезна. В принципе, при зависании демона **lpd** можно перезагрузить систему, но это радикальная мера. Прежде чем прибегнуть к ней, попробуйте уничтожить и перезапустить главный процесс **lpd**, а также вручную удалить задания из буферного каталога с помощью команды **rm**.

## Команда lpc: внесение административных изменений

Команда **lpc** выполняет следующие функции:

- включение и выключение режима постановки заданий в очередь на конкретный принтер;
- включение и выключение печати на конкретном принтере;
- удаление всех заданий из очереди принтера;
- перемещение задания в начало очереди;
- запуск, останов и перезапуск демона **lpd**;
- получение информации о состоянии принтера.

Пока система печати работает нормально, команда **lpc** тоже функционирует корректно. Но стоит "зависнуть" фильтру или появиться другой незначительной проблеме, как команда **lpc** полностью теряет контроль и начинает просто "врать": иногда она заявляет, что все исправлено, тогда как на самом деле ничего подобного не произошло. Приходится устранять проблемы вручную и даже отключать и включать питание, если печать выполняется очень плохо.

Команду **lpc** нельзя вызывать по сети; нужно зарегистрироваться на компьютере, к которому подключен принтер. Обычно команда используется в интерактивном режиме, хотя возможен и одноразовый ее вызов путем включения интерактивной директивы в командную строку. Команда **lpc** понимает описанные ниже директивы.

**help** [*директива*]

Если директива **help** вводится без аргументов, то выдается краткий перечень всех поддерживаемых директив. При наличии аргумента выдается однострочное описание указанной директивы.

**enable** *принтер*
**disable** *принтер*

Эти директивы включают и выключают буферизацию заданий для указанного принтера. Если постановка задания в очередь невозможна, пользователю сообщается об этом. Задания, уже находящиеся в очереди, не затрагиваются. Указанные операции реализуются путем установки или сброса бита, обозначающего право выполнения для группы, которой принадлежит файл **/var/spool/lpd/***принтер***/lock**.

**start** *принтер*
**stop** *принтер*

Эти директивы включают и выключают режим печати на указанном принтере. После выполнения директивы **stop** постановка заданий в очередь продолжается, однако вывод их на печать будет приостановлен до тех пор, пока режим печати вновь не будет включен. Эти операции реализуются путем установки или сброса бита, обозначающего право выполнения для владельца файла **/var/spool/lpd/***принтер***/lock**. Кроме того, описываемые директивы запускают и уничтожают соответствующие демоны печати. После получения директивы **stop** сначала завершается активное задание и только затем выключается режим печати.

**abort** *принтер*

Директива **abort** аналогична директиве **stop**, но при этом активное задание немедленно прерывается. После возобновления печати оно будет печататься заново.

**down** *принтер сообщение*
**up** *принтер*

Эти директивы влияют и на буферизацию, и на печать. Они используются в случае серьезного сбоя принтера, когда необходимо отключить его на длительное время. Параметр *сообщение* директивы **down** может иметь произвольную длину (в пределах одной строки) и не должен заключаться в кавычки. Указанное сообщение помещается в файл **/var/spool/lpd/***принтер***/status** и выдается всем пользователям, которые выполняют команду **lpq**. Как правило, сообщение содержит краткое разъяснение причины отключения принтера и информацию о том, когда он вновь заработает. Директива **up** выполняет противоположные действия.

**clean** *принтер*

Эта директива удаляет из очереди на принтер все задания, включая активное. Поскольку демон обработки очереди все равно будет содержать ссылки на файлы текущего задания, Linux не удалит их по-настоящему, и текущее задание будет завершено.

**topq** *принтер номер_задания*
**topq** *принтер имя_пользователя*

Первая директива **topq** перемещает в начало очереди указанное задание, а вторая — все задания, принадлежащие заданному пользователю.

**restart** *принтер*

Эта директива используется для перезапуска демона печати, который по каким-то причинам завершил работу. При отсутствии демона команда **lpq** сообщает: "No daemon present". На первый взгляд может показаться, что действие директивы **restart** аналогично действию связки **stop/start**, но это не так: если продолжает работать фильтр печати, то с помощью директивы **restart** перезапустить демон нельзя.

**status** *принтер*

Эта директива сообщает следующие сведения об указанном принтере: включен ли режим буферизации, разрешена ли печать на нем, сколько заданий стоит в очереди и каково состояние демона данного принтера. Если в очереди нет заданий, будет выдано примерно следующее.

```
lpc> status cer
cer:
    queuing is enabled
    printing is enabled
    no entries
    no daemon present
```

Факт отсутствия демона сам по себе не является причиной для беспокойства. Если очередь пуста, демоны печати прекращают работу и запускаются вновь главным процессом **lpd** только при постановке в очередь нового задания.

## Файл /etc/printcap

Этот файл является главной базой данных системы BSD. В ней содержится информация, необходимая для печати на локальных и удаленных принтерах. Задания на принтер можно передавать только в том случае, если он описан в этом файле.

Каждая запись файла **printcap** начинается со списка имен принтера, разделенных символами вертикальной черты (|). Затем следует ряд конфигурационных параметров, разделенных двоеточиями. Каждый параметр имеет вид *xx*, *xx=строка* или *xx#число*, где *xx* — двухсимвольное имя параметра, а *строка* и *число* — присваиваемые ему значения. Если никакого значения не присваивается, то переменная является булевой и ей соответствует значение "истина".

Допускаются пустые операторы: два двоеточия, стоящих рядом. Рекомендуется начинать и заканчивать каждую строку двоеточием, чтобы в будущем легче было вносить изменения. Комментарии в файле **/etc/printcap** должны начинаться со знака #. Запись может занимать несколько строк, при этом строки, имеющие продолжение, должны заканчиваться обратной косой чертой. По соглашению, строки продолжения немного сдвигаются вправо по отношению к первой позиции.

В качестве иллюстрации приведем небольшой пример, в котором определяется удаленный принтер, подключенный к компьютеру **anchor**.

```
anchor-lj|cer|1-56|LaserJet 5M in cer lab:\
    :lp=/var/spool/lpd/anchor-lj/.null:\
    :sd=/var/spool/lpd/anchor-lj:\
    :lf=/var/adm/lpd-errs:\
    :rw:mx#0:rm=anchor:rp=anchor-lj:
```

Из первой строки видно, что `cer`, `anchor-lj`, `1-56` и `LaserJet 5M in cer lab` — эквивалентные имена одного принтера. Принтерам можно присваивать сколько угодно имен, но обязательно следует указывать три формы основного имени:

- сокращенную (три-четыре символа, которые удобно вводить, например `cer`);
- полную (имя компьютера и тип принтера, например `anchor-lj`);
- описательную (прочая информация, например `LaserJet 5M in cer lab`).

Следующие три строки примера содержат конфигурационные параметры для имени устройства (`lp`), буферного каталога (`sd`) и журнала ошибок (`lf`). Последняя строка определяет режим чтения/записи при подключении к принтеру (`rw`), максимальный размер файла (`mx`; в данном случае не ограничен), имя удаленного компьютера (`rm`) и, наконец, имя удаленного принтера (`rp`).

Задания, поступившие в систему печати и не адресованные конкретному принтеру, направляются на первый принтер, среди псевдонимов которого есть **`lp`**. Псевдоним **`lp`** нельзя использовать в качестве основного имени принтера, поскольку замена стандартного принтера в этом случае будет затруднена. Если принтер с именем **`lp`** отсутствует, используется первый принтер, описанный в файле **`printcap`**.

## Переменные файла printcap

Своей гибкостью система BSD во многом обязана файлу **`printcap`**. Его структура хорошо описана на соответствующей **`man`**-странице, поэтому мы рассмотрим лишь самые полезные переменные этого файла (табл. 26.6).

**Таблица 26.6. Наиболее часто используемые переменные файла printcap**

| Имя | Тип | Назначение | Пример |
|---|---|---|---|
| `sd` | строка | Буферный каталог | `sd=/var/spool/lpd/howler-lw` |
| `lf` | строка | Журнал ошибок | `lf=/var/log/lpd-errors` |
| `lp` | строка | Имя устройства | `lp=/dev/lp0` |
| `af` | строка | Файл учета | `af=/var/adm/lpr.acct` |
| `rm` | строка | Имя удаленного компьютера | `rm=beast.xor.com` |
| `rp` | строка | Имя удаленного принтера | `rp=howler-lw` |
| `of` | строка | Выходной фильтр | `of=/usr/libexec/lpr/lpf` |
| `if` | строка | Входной фильтр | `if=/usr/sbin/stylascii` |
| `mx` | число | Максимальный размер файла | `mx#0` |
| `sh` | булев | Подавление печати титульных страниц | `sh` |

Все записи файла **`printcap`** должны включать, как минимум, спецификацию буферного каталога (`sd`), журнала ошибок (`lf`) и устройства печати (`lp`). Если используется современный принтер, нужно включить режим чтения/записи (`rw`), чтобы принтер мог возвращать сообщения об ошибках и статусные сообщения.

### *Переменная sd: буферный каталог*

У каждого принтера должен быть свой буферный каталог. Такие каталоги должны находиться в одном родительском каталоге (обычно это **`/var/spool/lpd`**) и называться в соответствии с полными именами обслуживаемых ими принтеров (в нашем примере это `anchor-lj`). Буферный каталог необходим даже в том случае, когда описываемый прин-

тер подключен к другому компьютеру. Задания находятся на локальном компьютере до тех пор, пока не будут переданы на печать.

При инсталляции нового принтера необходимо создать буферный каталог вручную и назначить ему режим доступа 775. Владельцем этого каталога должен быть пользователь **daemon**, и группа тоже должна называться **daemon**.

В буферном каталоге находятся два статусных файла: **lock** и **status**. Файл **status** содержит однострочное описание состояния принтера. Эту строку формирует демон **lpd**; она отображается командой **lpq**. Назначение файла **lock** заключается в том, чтобы избежать активизации нескольких экземпляров демона **lpd** для одной очереди. Кроме того, в нем хранится информация об активном задании. В процессе управления очередью печати и печатью на принтере команда **lpc** изменяет права доступа к файлу **lock**.

### Переменная lf: журнал ошибок

Сообщения, генерируемые фильтрами печати, помещаются в журнал ошибок. Один общий журнал может совместно использоваться всеми принтерами и располагаться где угодно. В каждой записи этого файла указывается имя соответствующего принтера. Журнальные файлы должны создаваться даже для удаленных принтеров, на случай если вдруг возникнет проблема со связью.

🕮 Подробнее о журнальных файлах рассказывалось в главе 11.

Помните о том, что демон **lpd** посылает сообщения об ошибках в систему Syslog. Некоторые фильтры направляют сообщения туда же, ничего не оставляя в журнальном файле. В случае возникновения проблем проверяйте оба источника информации.

### Переменная lp: имя устройства

Имя устройства должно назначаться только локальному принтеру. Обычно это имя файла в каталоге **/dev**, связанного с портом, к которому подключен принтер.

Демон **lpd** проверяет наличие блокировки файла, указанного в переменной `lp`, чтобы определить, используется ли принтер. Даже если доступ к принтеру осуществляется через сетевое соединение, переменная `lp` обязательно должна быть задана. Необходимо указать уникальный файл, который существует на локальном диске.

### Переменная rw: режим открытия устройства

Если принтер возвращает информацию о своем состоянии через файл устройства, должна быть определена булева переменная `rw`, запрашивающая открытие файла устройства в режиме чтения/записи. Этот режим используется для учета использования принтера и передачи статусных сообщений, поэтому некоторые фильтры требуют включения такого режима.

### Переменная af: файл учета

Можно учитывать объем распечатываемой пользователями информации, просто указав файл учета на компьютере, к которому принтер подключен физически. Записи вносятся в файл по завершении вывода задания на печать.

Для обобщения учетной информации используется команда **pac**. По соглашению, учетные файлы принтеров называются **/var/adm/***принтер*-**acct**. В них фиксируется количество страниц, напечатанных по каждому заданию (обычно не соответствующее действительности), имя компьютера, от которого было получено задание, а также имя владельца задания.

Записи для учета генерируются входным фильтром печати. Не стоит доверять счетчику страниц, если только фильтр не запрашивает показания счетчика до и после печати очередного задания.

### Переменная mx: ограничения на размеры файлов

Эта переменная задает предельный объем данных, посылаемых на печать за раз. Это значение имеет смысл только для построчно печатающих принтеров. Небольшие файлы формата PostScript или PCL могут выводить на печать сотни страниц.

В некоторых системах значение `mx`, устанавливаемое по умолчанию, отлично от нуля (0 означает отсутствие ограничений), и для того чтобы иметь возможность выводить на печать большие задания, необходимо явно указать `mx#0`. Отметим, что `mx` является числовой переменной, поэтому выражение `mx=0` ошибочно.

### Переменные rm и rp: информация об удаленном доступе

В большинстве случаев к принтеру необходимо обращаться не с одного, а с нескольких компьютеров. Даже если принтер является сетевым устройством, следует выбрать один компьютер и назначить его ответственным за связь с принтером. Все остальные компьютеры должны пересылать ему свои задания. Это позволяет демону **lpd** создавать единую очередь заданий, что исключает случаи конфликтов между компьютерами за право управления принтером. Кроме того, если печать не работает, администратор будет знать, где искать причину.

В файле **printcap** удаленного компьютера (не имеющего непосредственного соединения с принтером) будет присутствовать запись, в которой указывается, куда направлять задания. Переменная `rm` определяет компьютер, на который должны посылаться задания, а переменная `rp` задает имя принтера на этом компьютере.

Тот факт, что записи файла **printcap** для локальных и удаленных принтеров отличаются друг от друга, затрудняет совместное использование этого файла на разных компьютерах. Выход заключается в том, чтобы сделать локальное и сетевое имена принтера разными, например `howler-lw-local` и `howler-lw`. В такой конфигурации принтер `howler-lw` будет "удаленным" даже для того компьютера, к которому он непосредственно подключен, что вполне допустимо. Однако при работе с командой **lpc** придется указывать имя `howler-lw-local`.

### Переменные of и if: фильтры печати

Фильтры выполняют ряд функций. Фильтр печати, заданный по умолчанию (обычно это **/usr/lib/lpf**), обрабатывает различные управляющие последовательности и, если требуется, генерирует записи для учета. Не существует единого стандарта фильтров. Каждый поставщик стремится разработать свой набор уникальных фильтров, что усложняет задачу администраторам.

Даже если используется лазерный или струйный принтер или даже старинное наборное устройство или плоттер, вместе с программным обеспечением принтера обязательно поставляются необходимые фильтры.

Фильтры — это, как правило, просто сценарии оболочки, которые вызывают ряд программ преобразования данных. Фильтрующая программа должна принимать задание из стандартного входного потока, преобразовывать его в формат, поддерживаемый устройством, и записывать результат в стандартный выходной поток.

Если при вызове команды **lpr** пользователь не указал фильтр, будет использоваться либо входной (переменная `if`), либо выходной (переменная `of`) фильтр. Термины

"входной/выходной фильтр" не должны вводить в заблуждение, так как оба фильтра посылают данные на принтер.

Если в файле **/etc/printcap** присутствует переменная `if`, но нет переменной `of`, файл устройства будет открываться один раз для каждого задания, а программа-фильтр будет посылать одно задание на принтер и завершать работу.

Если определен только выходной фильтр, демон **lpd** однократно откроет файл устройства и вызовет программу-фильтр для обработки сразу всех заданий в очереди. Это полезно для тех устройств, соединение с которыми устанавливается очень долго, хотя такие устройства встречаются редко.

Если определены оба фильтра, то выходному посылается титульная страница (этот фильтр вызывается даже в том случае, когда печать титульных страниц отключена), а входной обрабатывает остальную часть задания. Такая комбинация слишком сложна для непосвященных, поэтому лучше ее избегать.

При написании новых фильтров ориентируйтесь на входные фильтры: их легче отлаживать. Входные фильтры вызываются с многочисленными аргументами. Наиболее интересные из них — имя пользователя, имя компьютера и имя учетного файла. Если требуется организовать учет работы принтера, то входной фильтр должен генерировать записи для учета и добавлять их в конец учетного файла. Если необходимо ограничить доступ к принтеру (например, запретить печать пользователю **guest** на всех компьютерах), то это тоже должен делать входной фильтр, поскольку демон **lpd** не имеет соответствующих встроенных средств.

В качестве иллюстрации рассмотрим очень простой пример входного фильтра. В данном случае это фильтр PostScript-принтера, подключенного через последовательный порт к локальному компьютеру.

```
#!/bin/csh -f
/usr/local/bin/textps $* | /usr/local/bin/psreverse
```

Поскольку принтер подключен последовательно, демон **lpd** открывает файл устройства в соответствующих режимах согласно указаниям файла **/etc/printcap**. Первой вызывается программа **textps**, которая анализирует входные данные и определяет их формат. Если это не PostScript (формат, поддерживаемый принтером), выполняется преобразование. Программа **textps** принимает все аргументы фильтра (`$*`) и на основе этой информации генерирует записи для учета. Вторая программа, **psreverse**, изменяет порядок следования страниц на противоположный.

### *Переменные файла printcap для устройств последовательного доступа*

Многие переменные и функции из файла **printcap** используются для работы с устаревшими принтерами, подключаемыми к последовательному порту. Если вы устанавливаете сетевой принтер, то найдите время и откройте руководство, найдите спецификации, касающиеся взаимодействия с принтером, и хорошенько напейтесь. Если же вы не пьете, то потратьте деньги, предусмотренные в смете для покупки алкоголя, на приобретение нового принтера.

### *Расширения файла printcap*

У системы BSD есть одна прекрасная особенность: если пользователь задает значения нестандартных переменных файла **printcap**, они игнорируются. Часто, когда конкретному принтеру нужно больше сведений о конфигурации, чем имеется в базовой системе, можно определить в файле **printcap** дополнительные переменные, которые будут использоваться фильтрами печати.

Например, выходному фильтру сетевого принтера может потребоваться сетевое имя этого устройства. В запись файла **printcap** для этого принтера можно добавить следующую установку.

```
:nn=laser.colorado.edu:\
```

Использование такого рода расширений позволяет хранить всю информацию о конфигурации принтера в одном, удобном для администратора месте. Если вы вдруг увидите в файле **printcap** переменные, не упомянутые на **man**-странице, поищите их значения в документации к фильтрам принтера.

В нашей сети описанным выше способом документируется физическое расположение каждого принтера. Все принтеры имеют установки следующего вида.

```
:lo=Room 423, Engineering building:\
```

Мы используем сценарии, которые отслеживают наличие бумаги и запас тонера в принтерах и при возникновении проблем посылают группе технического обслуживания сообщения типа "Добавьте бумагу в принтер, находящийся в комнате 423 здания машиностроительного факультета".

# 26.6. Долгая странная история

Из предыдущих разделов становится ясно, насколько разнятся три основные системы печати, почему разработчикам стоит отказываться от старых систем и переходить на систему CUPS и почему ее разработчики приняли мудрое решение сохранить команды, напоминающие команды старых систем.

Как же все это произошло? Приведем краткое изложение истории систем печати.

## История печати и появления систем печати

Много лет тому назад были распространены игольчатые принтеры. Лазерные принтеры встречались редко, да и стоили дорого. Для устройств вывода с высоким разрешением требовались специальные драйверы и программы для форматирования.

В настоящее время лазерные принтеры часто подключены к сети TCP/IP через интерфейс Ethernet, Wi-Fi или Bluetooth, а не к единственному компьютеру с помощью последовательного или параллельного порта. На рынке дешевой продукции лазерные принтеры уступили место струйным. Цветными принтерами, которые когда-то считались роскошью и обеспечивали качество печати, сравнимое с качеством цветной фотографии и изображений на дисплеях, сегодня никого не удивишь. Найти черно-белый принтер скоро будет также сложно, как черно-белый телевизор, если к тому времени телевизоры вообще сохранятся.

Специализированные принтеры, сканеры, копировальные машины и факсы постепенно вытесняются многофункциональными устройствами, которые способны выполнять всю их работу. Некоторые из них уже сейчас могут считывать файлы непосредственно из микросхемы памяти на цифровой видеокамере.

Прежние принтеры были примитивными, как и их механизмы буферизации. Предполагалось (по правилам), что компьютер, на котором вы работали, непосредственно подсоединен к принтеру. Конфигурирование принтера сводилось к ответу на вопрос: "Последовательный или параллельный интерфейс?" Это касалось и операционных систем, отличных от UNIX, хотя все они были специализированными: системы IBM знали, как управлять принтерами IBM, компьютеры Apple знали, как управлять принтерами

Apple, и т.д. Используемый компьютер часто должен был (по правилам) подключаться к принтеру напрямую. Настройка принтера заключалась в предоставлении ответов на вопросы типа "Принтер с последовательным интерфейсом или принтер с параллельным интерфейсом?"

Первым коммерческим приложением для системы UNIX, проданным компанией INTERACTIVE Systems Corporation, была система подготовки документов для юридической фирмы. Основными компонентами этой системы были текстовый редактор, языки разметки (**nroff/troff**) и программы для печати.

Поскольку технологии становились все сложнее и сложнее, предпринимались попытки создать универсальный стандарт, но ни одна из них не имела должного успеха. Используемые протоколы устаревали и работали все хуже и хуже.

Системы печати BSD и System V были разработаны довольно давно для игольчатых принтеров. Эти системы, переделанные и перегруженные в попытках догнать развивающиеся технологии, так никогда и не смогли обеспечить поддержку современных принтеров, и каждая новая функциональная возможность принтеров, например дуплексирование (двусторонняя печать), требовала специальных доработок.

Почему же существовали только две конкурирующие системы печати и почему так важны различия между ними? Встаньте посредине группы пользователей и воскликните: "Все, кто использует команду **vi** вместо **emacs**, — идиоты!", а затем приходите и задавайте свои вопросы.

Когда стали доступны сетевые принтеры, количество проблем только возросло. Первые сетевые системы печати были слишком уникальными и использовали ряд протоколов для обеспечения связи между принтером и очередью печати, между клиентом и очередью печати и для осуществления обмена сетевым трафиком.

Принтеры JetDirect компании HP часто принимали необработанные данные на порт 9100, как это делали принтеры старых разработчиков, принявших правила компании HP. Принтеры с внутренними демонами **lpd** (реализациями протокола BSD) принимали задания на порту 515.

Стиснув зубы, рабочая группа по решению проблем печати (Printing Working Group) из института IEFT создала протокол IPP (Internet Printing Protocol — протокол печати через Интернет) поверх протокола HTTP. Он не только оформлял взаимодействия в виде простых запросов GET и POST, но и позволял системам печати пользоваться преимуществами стандартных, широко используемых технологий для аутентификации, контроля доступа и шифрования.

Майкл Свит (Michael Sweet) и Эндрю Сенфт (Andrew Senft) из компании ESP (Easy Software Products) перенесли IPP в UNIX в виде реализации CUPS. Компания Apple адаптировала систему печати CUPS для операционной системы Mac OS X (и в 2007 году купила ее исходный код), после чего система CUPS стала самой полноценной реализацией протокола IPP на планете. Система CUPS является свободно распространяемым проектом с открытым кодом, в котором устранены проблемы, связанные со старыми системами.

## Разнообразие принтеров

Кроме разнообразия систем печати, администраторы столкнулись с разнообразием самих принтеров.

Поскольку принтеры могут подключаться к компьютерам, пользователи отождествляют их с периферийными устройствами, такими как мышь и мониторы. Однако они

представляют собой более сложные устройства. На самом деле они больше напоминают смартфоны или маршрутизаторы, но с движущимися частями.

Одно время самым мощным компьютером, который когда-либо создавала компания Apple, был принтер Apple LaserWriter. В настоящее время ваш настольный компьютер, вероятно, мощнее принтера, но принтер остается компьютером. Он имеет центральный процессор, память, операционную систему, а иногда и диск.

Сетевой принтер имеет свой сетевой стек и IP-адрес. Если у вас есть сетевой принтер, введите его адрес (или имя DNS) в ваш веб-браузер. Скорее всего, принтер обслуживает несколько веб-страниц, с помощью которых вы можете управлять его аппаратными компонентами: принтер запускает свой собственный веб-сервер.

Поскольку системные администраторы озабочены вопросами безопасности, вы, возможно, уже подумали: "Значит ли это, что принтер можно взломать или подвергнуть атаке на основе отказа в обслуживании?" Увы, да. Вопросы безопасности рассматриваются в разделе 26.12.

Какая операционная система работает на вашем принтере? Как?! Вы не знаете? Ничего удивительного. Вы, вероятно, не можете найти нужную информацию, даже хорошенько покопавшись в документации. Операционные системы принтеров зависят от разработчиков, а иногда даже от модели. Средние и дорогие принтеры могут даже запускать операционные системы, производные от UNIX или Linux.

Ваш принтер способен выполнять множество протоколов и принимать задания на разных языках, описывающие страницы и документы. Он может даже понимать и распечатывать такие форматы, как GIF, JPG и TIFF.

Ваш принтер может быть черно-белым или цветным. Он может печатать страницы с разрешением от 150 до 2400 точек на дюйм (dpi — dots per inch) и даже страницы с асимметричным разрешением, например 1200×600, т.е. 1200 dpi в одном направлении и 600 dpi в другом.

Если вы управляете крупной сетью, то вам, вероятно, придется обслуживать несколько моделей принтеров, созданных разными производителями, каждый из которых имеет разные возможности. Это значит, что программное обеспечение для принтеров на ваших компьютерах должно справляться с разнообразным (и часто неизвестным) аппаратным обеспечением с помощью набора протоколов.

## 26.7. Основные программы печати

Печать — это не только буферизация и задания на печать. Даже на системе Ubuntu (которая использует систему печати CUPS) команда

```
$ man -k . | egrep -i 'ghostscript|cups|print(er|ing| *(job|queue|filter))'
```

отображает более сотни страниц из руководства, посвященных проблемам печати, — и это при поиске "на скорую руку". (Кстати, изучая связанные с печатью команды, обратите внимание на то, что не все команды имеют отношение к печати. Например, **apcupsd** — это демон, который взаимодействует с устройством Universal Power Supplies, разработанным компанией APC, и даже команда **print** не имеет никакого отношения к печати.) О некоторых из этих команд и утилит стоит знать.

В системах BSD и System V недостает многих функций, связанных с преобразованиями форматов, необходимыми для управления современными принтерами. По этой причине многие разработчики, использующие эти системы, имеют хотя бы один набор инструментов, реализованных на основе их систем печати, для выполнения этих функций.

Иногда эти инструменты включаются в операционную систему, но чаще они продаются как надстройки. Широко используются также свободно распространяемые пакеты, разработанные сторонними производителями.

Утилита **pr** — это один из самых старых инструментов для печати. Она изменяет формат текстовых файлов так, чтобы он соответствовал формату печатной страницы. В частности, она разбивает содержащиеся в них данные на включающие по 66 строк страницы, добавляет верхние и нижние колонтитулы и, при необходимости, делает между строками двойной интервал. (Почему 66? Потому что именно столько строк помещалось на странице, которая распечатывалась старыми построчными принтерами.)

Команда **enscript**, разработанная компанией Adobe, выполняет похожие преобразования с несколькими дополнительными "наворотами"; она тоже выводит данные в формате PostScript. Команда **enscript**, разработанная в рамках проекта GNU, представляет собой распространяемую с открытым исходным кодом версию этой команды и имеет обратную совместимость с аналогичной командой Adobe; однако команда **enscript** предлагает ряд новых функциональных возможностей, в частности, чувствительное к языку выделение, поддержка для различных форматов (размеров бумаги), загрузка шрифтов и определяемые пользователем заголовки.

Одним из главных преимуществ команды **enscript** была реализация возможности печати сразу двух страниц на одном листе (2-up printing). Те, кто до сих пор пользуется командой **enscript** из-за этой возможности, могут попробовать применить в системе CUPS команду **lpr -o number-up=2**.

Одной из самых сложных считается утилита Ghostscript, которую первоначально создал Л. Питер Дойч (L. Peter Deutsch) для того, чтобы иметь возможность печатать документы PostScript на недорогих принтерах PCL. Сегодня программа Ghostscript интерпретирует не только формат PostScript, но и формат PDF. Система CUPS использует ее в качестве фильтра, но программа Ghostscript также может создавать изображения страниц для экрана как самостоятельно, так и при помощи внешних программ наподобие **gv**, GNOME Ghostview (**ggv**), или KDE KGhostView.

Абсолютно все дистрибутивы Linux поставляются с бесплатной версией программы Ghostscript, подробнее о которой можно узнать на сайте `ghostscipt.com`. Коммерческая версия Ghostscript с поддержкой доступна на сайте компании Artifex Software.

## 26.8. Языки принтеров

Задание на печать — это на самом деле компьютерная программа, написанная на специальном языке программирования. Они называются языками описания страниц или просто PDL-языками (PDL — Page Description Language).

Страницы, кодируемые на языке PDL, могут быть гораздо меньше по размеру (иногда больше) и передаваться гораздо быстрее аналогичных необработанных изображений. Кроме того, PDL-описания также могут не зависеть ни от используемых устройств, ни от их разрешающей способности.

Наиболее известными PDL-языками на сегодняшний день являются PostScript, PCL5, PCL6 (также называемый "PCL/XL" или "pxl") и PDF. Многие принтеры способны принимать входные данные на более чем одном языке. Каждый из этих языков кратко описывается в следующих разделах.

Принтеры должны интерпретировать задания на этих языках и преобразовывать их в какой-нибудь понятный для используемого устройства обработки изображений формат. Поэтому принтеры содержат языковые интерпретаторы. Точно так же как и у языка C

или Java, у этих языков имеется множество версий, и каждая из них работает по-своему. Почти все принтеры PostScript понимают PostScript Level 3, но если вы отправите программу Level 3 на принтер, который понимает только Level 2, принтер, скорее всего, не сможет ее выполнить.

Процесс преобразования PDL-описания (или, например, графических файлов) в растровые изображения страниц называется *обработкой растровых изображений*, а программа, которая выполняет его, — *процессором растровых изображений* (Raster Image Processor) или просто *RIP*.

Задания на печать можно копировать на компьютер и просматривать на экране. Централизованные интерпретаторы, которые позволяют это делать, такие как Ghostscript, описываются в разделе 26.11. В теории можно даже делать следующее: копировать задания на компьютер и затем отправлять готовые (и значительно большие по размеру) растровые изображения на печать какому-нибудь "не очень разумному" устройству печати. На самом деле именно так и работают многие принтеры GDI (Windows), да и в системе Linux этот подход тоже поддерживается до некоторой степени.

## PostScript

Это самый распространенный из всех встречающихся на системах Linux языков семейства PDL. Первоначально PostScript был разработан компанией Adobe Systems, и многие принтеры до сих пор используют интерпретатор, требующий наличия лицензии от Adobe. Почти все программы компоновки страниц умеют генерировать данные на языке PostScript, а некоторые вообще работают только с такими данными.

PostScript — этот полнофункциональный язык программирования. Почти все написанные на нем программы могут просматриваться с помощью текстового редактора или утилиты **less**. Эти программы отличаются обилием круглых и фигурных скобок, а также символов косой черты и часто начинаются символами `%!PS`. Хотя эти начальные символы не являются обязательными в самом языке PostScript, PostScript-интерпретаторы и другие программы для печати часто ищут их, пытаясь распознать и классифицировать задания на печать.

## PCL

Одной из альтернатив языка PostScript является разработанный компанией Hewlett-Packard язык PCL (Printer Control Language — язык управления принтерами). Его понимают не только принтеры производства Hewlett-Packard, но и принтеры многих других производителей; некоторые принтеры вообще умеют работать только с ним. В отличие от PostScript, который поддерживает все операции Тьюринга и является универсальным языком программирования, PCL просто сообщает принтерам, как им следует печатать страницы. Написанные на PCL задания на печать имеют бинарный, не доступный для понимания человека формат и обычно намного короче аналогичных заданий, написанных на языке PostScript. Приложения Linux редко генерируют выходные данные на языке PCL напрямую, но фильтры могут преобразовывать PostScript в PCL.

В отличие от языка PostScript, версии языка PCL разнятся лишь немного. Отличия небольшие, но достаточно значительные для того, чтобы вызвать раздражение. Задания, которые печатаются правильно на LaserJet 5si, могут печататься немного не так на LaserJet 5500 и наоборот. И это касается не только этой пары моделей; каждый PCL-принтер имеет свой диалект языка PCL со своими специальными командами, которые извлекают пользу из функциональных возможностей именно этого принтера.

Например, если указать компьютеру, что используется принтер LaserJet 4500, когда на самом деле используется принтер LaserJet 4550, он может сгенерировать какие-нибудь PCL-команды, которые принтер LaserJet 4550 проигнорирует или неправильно интерпретирует. Также, если имеется какое-нибудь сохраненное, написанное на языке PCL задание на печать (скажем, чистый бланк на оформление покупки), а принтер, для которого оно было сгенерировано, заменяется на какой-нибудь более новый, не исключено, что это задание придется генерировать заново.

Еще хуже то, что компания HP (Hewlett-Packard) определила два совершенно не связанных между собой семейства языков — PCL5 (PLC5C означает "цветной", PLC5E — "черно-белый") и PCL6 (второе название — "PCL/XL"). Сегодня новые принтеры HP обычно имеют языковые интерпретаторы для обоих.

## PDF

Разработанный компанией Adobe формат PDF (Portable Document Format — формат переносимых документов) генерируется программой Adobe Acrobat, а многие другие настольные средства для издательского дела, такие как OpenOffice, позволяют экспортировать документы в этом формате.

PDF-документы не зависят от платформы. Формат PDF очень часто используется для обмена электронными документами с возможностью последующего просмотра и печати. Окончательный вариант текста данной книги, например, передавался на принтер именно в виде PDF-файла.

PDF — это язык описания документов, а не просто язык описания страниц. Он позволяет описывать не только отдельные страницы, но и общую структуру документа: к каким главам относятся те или иные страницы, какие текстовые столбцы переходят в другие текстовые столбцы и т.д. Он также предлагает ряд различных мультимедийных возможностей для использования на экране.

Некоторые принтеры интерпретируют PDF напрямую. Если ваш этого не делает, существует масса программ-трансляторов и программ для просмотра PDF-файлов (среди них Ghostview, **xpdf**, **kpdf**, Evince и Acrobat Reader), которые позволяют преобразовать документы в какой-нибудь более универсальный формат (такой, как PostScript например). Ваша система печати может даже скрывать требование на выполнение преобразования от вас и автоматически преобразовывать PDF-документы перед их отправкой на принтер.

## XPS

Стоит также упомянуть язык Paper Specification, разработанный компанией Microsoft. Он также называется XPS или OpenXPS. Этот язык не получил широкого распространения даже в системах Windows. Системы UNIX и Linux слабо поддерживают этот язык, хотя компания Artifex уже создала интерпретатор языка XPS. Дистрибутивные пакеты системы Linux, несомненно, вскоре включат поддержку языка XPS, если он станет популярным.

Зачем это может быть нужно пользователям? Представьте, что вы — начальник отдела маркетинга, просматривая веб-страницы на своем телефоне, вдруг обнаруживаете среди них одну, очень подходящую к теме вашей презентации, которую вы как раз собираетесь провести. Вы подходите к ближайшему поддерживающему Bluetooth принтеру и отправляете ему со своего телефона URL-адрес этой страницы. Дальше принтер делает все сам: загружает страницу из Интернета, визуализирует ее и печатает копии. Вы берете копии из лотка и спокойно направляетесь на свою презентацию.

## PJL

PJL (Printer Job Language — язык заданий принтеров; разработан компанией Hewlett-Packard) — это не PDL-язык, а метаязык, который описывает задания принтеров. Мы рассказываем о нем здесь потому, что он будет встречаться в описаниях принтеров.

Итак, PJL — это язык управления заданиями, который позволяет указывать, какой PDL-язык должен использоваться для задания, это задание типа дуплекс или симплекс, какой размер бумаги следует использовать и т.д. PJL-команды идут в начале задания, а PJL-операторы все начинаются с символов @PJL.

```
@PJL SET COPIES=3
@PJL COMMENT FOO BAR MUMBLE
@PJL SET DUPLEX=ON
@PJL SET PAGEPROTECT=OFF
@PJL ENTER LANGUAGE=PCL
```

PJL обычно нормально распознается (или преднамеренно игнорируется) принтерами стороннего производства (не Hewlett-Packard), но в случае проблем при распечатке такого документа на стороннем принтере попробуйте удалить PJL при помощи текстового редактора и снова отправить задание на печать.

## Драйверы принтеров и как они обрабатывают PDL-языки

А что если принтер поддерживает только некоторые из нужных языков? Что делать, если из Интернета загружен файл PostScript, а имеющийся принтер распознает только PCL5E? Как распечатать PDF-файл, если принтер не интерпретирует PDF напрямую?

Один из вариантов — это преобразовать файл вручную. Дистрибутивы Linux поставляются со множеством утилит преобразования; практически всегда есть какой-нибудь способ превратить имеющийся документ в документ, который может быть распечатан на принтере. Браузеры, например, позволяют преобразовывать страницы HTML (или XHTML) в PostScript. OpenOffice позволяет преобразовывать файлы MS Word в формат PDF. Ghostscript позволяет преобразовывать PDF-файлы в формат PostScript, а уже из формата PostScript — в почти любой другой формат, включая PCL.

Более простой вариант — позволить системе печати сделать все самой. Многие системы имеют встроенные знания о том, какие преобразования должны выполняться, и могут выполнять их автоматически.

Если необходимо определить, какой PDL-язык используется в файле, и сделать это по имени файла (например, **food.pdf**) невозможно, воспользуйтесь командой **file** (если только файл не начинается с ряда PJL-инструкций, потому что в таком случае команда **file** вернет просто такое сообщение: "HP Printer Job Language data", т.е. "данные на языке заданий принтеров").

Сохраните несколько заданий на печать в файлах, вместо того чтобы отправлять их на принтер, и увидите, как выглядит программа на одном из этих языков. Изучив файлы каждого из этих типов минуту или две в своем текстовом редакторе, вы поймете, насколько они разные. (Не выводите их прямо на экран при помощи команды **cat**, потому что только PostScript отображается в формате ASCII. Случайные бинарные данные только запутывают интерпретаторов терминалов.)

PostScript

```
%!PS-Adobe-3.0
%%BoundingBox: 0 0 612 792
%%Pages: 1
```

```
% ...
% Draw a line around the polygons...
pop pop pop dup 0 setgray 0 0 moveto dup 0 lineto 0.707106781 mul dup
   lineto closepath stroke
```

PDF

```
%PDF-1.3
%A.A.AA"
81 0 obj
<<
/Linearized 1
/O 83
/H [ 915 494 ]
/T 125075
>>
endobj
 xref
81 24
0000000016 00000 n
A€<8f>
^P^@A¤A'<9e>
endstream
endobj
```

PCL5

```
^[E^[&l1o0o1t016D^[&l1X^[*r0F^[*v0n1O^[*p4300X^[%1BDT~,1TR0TD1SP1FT10,50
CF3,1LB.~;^[%1A^[*c100G^[*v2T^[&a0P^[*p0X^[*p0Y^[(10U^[(s1p12vsb4148T^[&l0
E^[*p0Y^[*ct7920Y^[(10U^[(s1p12vsb4101T^[&a0P^[&l0o66f0E^[9^[&a0P^[*p0X^[*
p0Y^[*p474Y^[*p141X^[(10U^[(10U^[(s1p12vsb4101T^[*p402Y^[*p186X^[*v0O^[*c9
00a4b100g2P^[*v1O^[*p250Y^[*v0O^[*c900a4b100g2P^[*v1O^[*v0O^[*c4a156b100g2
P^[*v1O^[*p251Y^[*p187X^[*v0O^[*c899a154b10g2P^[*v1O^[*p346Y^[*p256X
```

PCL/XL

```
A'X^BX^BA.<89>A^@A.<86>A^CA.<8f>AA^@A.<88>A^AA.<82>HA^@A.(A^@A.%A
A.cA^AP^@TimesNewRmnBdA.A.A…UUA©BA.A¦Au^BA.A.o<85>A"A›^CA^BA.Lk
A^@^@A.A.dA€A:^@
```

# 26.9. Файлы PPD

При вызове демона **lpr** для распечатки файла **book.ps** на цветном принтере Pollux, он может спросить, бумага какого размера должна использоваться для печати. Но подождите, откуда система CUPS знает, что ей нужно указать своему клиенту, демону **lpr**, что принтер Pollux может выполнять печать на бумаге размера A4? Откуда система CUPS знает, что он понимает язык PostScript, и что ей делать, если это не так? Где система CUPS находит информацию о том, что Pollux является цветным принтером?

Если вы используете систему CUPS, то вся эта информация содержится в файле PPD (PostScript Printer Description — файл описания принтера на языке PostScript), который описывает атрибуты и возможности принтера, поддерживающего язык PostScript. Демон CUPS считывает PPD-файлы обслуживаемых им принтеров и передает информацию о них клиентам и, если необходимо, фильтрам.

Файлы PPD изначально разрабатывались для мира Mac, но были быстро переняты системой Windows. Каждый новый принтер поставляется с файлом PPD от производителя. Драйверы принтеров Mac и Windows используют файл PPD для выяснения того,

как отправлять на принтер задания на языке PostScript. Например, нет никакого смысла просить поддерживающий только одностороннюю печать черно-белый принтер, продаваемый в Америке, распечатать двухсторонний цветной документ на европейской бумаге формата B4.

Каждый принтер, поддерживающий язык PostScript, имеет свой собственный файл PPD, хотя его нелегко найти. Проверьте инсталляционный диск и веб-сайт разработчика.

Файлы PPD — это просто текстовые файлы. Откройте один из них в текстовом редакторе и посмотрите, какая информация в нем содержится. В сети файл PPD может быть даже удаленным, а пользователи системы CUPS могут получить информацию на соответствующем сервере CUPS.

Система CUPS также использует файлы PPD для описания принтеров, у которых нет интерпретатора PostScript. Весь трюк заключается в одном дополнительном поле.

```
$ grep cupsFilter /usr/share/cups/model/pxlmono.ppd
*cupsFilter:    "application/vnd.cups-postscript 100 pstopxl"
*cupsFilter:    "application/vnd.cups-pdf 0 pstopxl"
```

Для того чтобы увидеть, чем именно отличаются два типа принтеров, можно сравнить два связанных друг с другом файла PPD (например, файл **pxlmono.ppd** и файл **pxlcolor.ppd**) при помощи команды **diff**.

Если производитель принтера не предоставляет файл PPD — возможно, потому, что у принтера нет интерпретатора PostScript и производителя не волнует ничего, кроме системы Windows, — следует зайти на сайт `linuxprinting.org` и поискать информацию в базе данных Foomatic. Не исключено, что принтер поддерживается проектом Gutenprint (`gutenprint.sourceforge.net`). Если доступны несколько или все перечисленные источники файлов PPD и пользователи хотят наилучшего возможного качества, нужно попробовать источник с пометкой "footmatic+gutenprint".

Если отыскать PPD-файл нигде не удается, тогда примите следующие советы.

- Вероятно, следовало проверить принтер по базе данных на `linuxprinting.org`, прежде чем приобретать его. Даже если вы получили принтер по цене металлолома, "свободно распространяемый" не всегда означает "бесплатный".
- Вполне вероятно, что существует какой-нибудь файл PPD, который позволит выполнять печать, хотя и не будет пользоваться преимуществами всех функциональных возможностей принтера. Например, нам удалось использовать типичные драйверы HP на принтерах других компаний.

Несмотря на то что система CUPS зависит от файлов PDS, старые системы печати не используют их. Для систем BSD и System V вы можете либо сами настроить механизм PostScript, либо смириться с тем, что получаете по умолчанию.

## 26.10. Форматы бумаги

Пользователи хотят видеть информацию на бумаге. Бумага имеет размер и форму. Для того чтобы пользователи были довольны, следует знать основные факты о форматах бумаги.

В США и Канаде самый распространенный формат бумаги Letter размером 8,5×11 дюймов. Некоторые дистрибутивы системы Linux (такие, как пакеты Knopix и SUSE) выпускаются в Европе, где никто даже не знает, что такое дюймы, в Англии знают, что такое дюймы, но не пользуются ими для измерения бумаги. В этих странах и в Японии самый распространенный формат бумаги A4, и все принтеры поставляются с лотками

размера A4. Поэтому утилиты печати в некоторых дистрибутивах генерируют изображения размером со страницу формата A4 по умолчанию.

Лист формата A4 является удобным, потому что он иррационален (с математической точки зрения). Соотношение длины к ширине у листа формата A4 равняется $\sqrt{2}$. Если сложить лист A4 пополам горизонтально, получится два листа с одинаковым соотношением длины к ширине. Такой формат листа уже называется A5. Если сложить пополам лист формата A5, получится два листа формата A6. Если пойти в обратном направлении, лист формата A3 будет по площади в два раза больше листа формата A4, но точно такой же формы, и т.д.

Другими словами, можно изготавливать листы формата A0, которые по площади равняются одному квадратному метру, и при помощи бумагорезальной машины создавать из них листы необходимых размеров. Единственным распространенным в США форматом бумаги, с которым можно проворачивать подобное, является формат Ledger (11x17 дюймов, также известный как "таблоид"): если сложить лист такого формата пополам, получится два листа формата Letter.

Также еще существуют ISO-форматы серий B и C, которые сохраняют пропорцию $1:\sqrt{2}$, но имеют другую базовую площадь. Формат B0 составляет 1 метр в высоту, а C0 имеет площадь 2 м$^2$. Инженеры сразу же увидят, что стороны листа B*n* представляют собой среднее геометрическое сторон A*n-1* и A*n*, а стороны листа C*n* — среднее геометрическое сторон A*n* и B*n*.

Что это все означает? Формат B*n* выглядит точно так же, как A*n*, но он больше, а формат C*n* представляет собой нечто среднее между ними двумя. Отчет на листе формата A4 прекрасно поместится в папку формата C4. Если сложить пополам письмо формата A4, оно превратится в письмо формата A5, которое легко войдет в конверт формата C5. А если его сложить еще раз, оно также легко войдет и в конверт формата C6.

Немного усложняет дело то, что в Японии имеется своя серия форматов B, которые немного отличаются. Хоть они и имеют те же пропорции, что и ISO-листы, японский формат B4 представляет собой среднее арифметическое форматов A3 и A4, что делает его немного больше ISO-формата B4. Японских форматов серии C не существует.

Система ISO позволяет копировать две страницы формата B5 на одну страницу формата B4, а также печатать сколько угодно уменьшенных изображений страниц на одном листе. На европейских копировальных машинах часто имеются кнопки, позволяющие уменьшать или увеличивать изображение на множитель $\sqrt{2}$.

Если в системе установлена команда **paperconf**, можно использовать ее и сделать так, чтобы размеры разных известных форматов бумаги отображались в дюймах, сантиметрах или точках (каждая из которых составляет 1/72 дюйма). В табл. 26.7 перечислены некоторые наиболее распространенные случаи использования основных форматов бумаги (для американцев).

**Таблица 26.7. Основные ISO-форматы бумаги**

| Форматы | Назначение |
|---|---|
| A0, A1 | Объявления |
| A3, B4 | Газеты |
| A4 | Универсальные документы |
| A5 | Блокноты (примерно 5x8 дюймов) |
| B5, B6 | Книги, открытки, немецкая туалетная бумага |
| A7 | Учетные карточки "3x5" |

*Окончание табл. 26.7*

| Форматы | Назначение |
|---|---|
| B7 | Паспорта (даже в США паспорта имеют формат B7) |
| A8 | Визитные карточки |
| B8 | Игральные карты |

К сожалению, бумага формата A4 немного тоньше и длиннее (8,3 × 11,7 дюймов) бумаги американского формата Letter. Печать документа формата A4 на бумаге формата Letter обычно заканчивается обрезанием таких его важных частей, как верхние колонтитулы, нижние колонтитулы и номера страниц. И наоборот, печать документа формата Letter на бумаге формата A4 приводит к усечению его сторон (хотя это и менее важно).

Некоторые программные пакеты имеют собственные параметры по умолчанию для формата бумаги. Например, утилита **enscript** по проекту GNU обслуживается в Финляндии человеком по имени Маркку Росси (Markku Rossi) и по умолчанию использует формат A4. Те, кто проживают в Америке и пользуются дистрибутивом, в котором утилита **enscript** не была скомпилирована с другим параметром по умолчанию, могут сделать только одно — взять исходный код и переконфигурировать его. Однако обычно проще поступить следующим образом: указать нужный формат бумаги в командной строке или конфигурационном файле GUI. Если документы печатаются с обрезанными концами или сторонами, причиной, скорее всего, является несовместимость форматов бумаги.

Иногда можно настроить используемый по умолчанию формат бумаги для многих связанных с печатью задач при помощи команды **paperconfig**, переменной среды PAPERSIZE или содержимого файла **/etc/papersize**. (Обратите внимание: **paperconfig != paperconf**.)

Следует признать, что информация выводится не только на бумагу. Если в хлебный отдел крупного супермаркета принести цветную фотографию, то, возможно, кулинары смогут сделать для вас торт, на поверхности которого будет ваше изображение. Такие рисунки печатаются на специализированных растровых принтерах, использующих съедобные чернила. Крупные прямоугольные торты называются "тортами на противне" ("sheet cake") и тоже имеют стандартные размеры. К сожалению, мы вынуждены закончить наш разговор о форматах бумаги. Мы не можем говорить обо всем на свете ...

## 26.11. Практические советы

Принтеры могут стать причиной беспокойства. Для того чтобы смягчить это беспокойство, мы решили сформулировать несколько советов. Если все остальные устройства отказали, радуйтесь тому, что вы, по крайней мере, не обязаны до сих пор возиться с игольчатыми принтерами, подсоединенными через последовательный порт RS-232. (Если это так, конечно.)

### Выбор принтера

Если вы используете систему CUPS, то прежде чем покупать принтер или брать "бесплатный" принтер, который кто-то выкидывает, следует заглянуть в базу данных Foomatic на сайте `linuxprinting.org` (которая поддерживается и финансируется организацией Linux Foundation) и проверить, насколько хорошо он поддерживается в системе Linux. В этой базе данных все принтеры поделены на четыре категории, начиная от категории Paperweight (Пресс-папье) и заканчивая категорией Perfectly (Отлично); нам нужна категория категорию Perfectly.

Все предпочитают принтеры со встроенным интерпретатором языка PostScript. Эти принтеры обычно имеют простую конфигурацию.

Принтеры, не понимающие язык PostScript, тоже поддерживаются, но не так хорошо. Для того чтобы выполнять печать на этих принтерах, потребуется специальное программное обеспечение, умеющее преобразовывать задания на печать в предпочитаемый принтером формат PDL или другой формат данных. Не исключено, что такое программное обеспечение удастся найти в пакете CUPS или в одном из других упоминавшихся ранее в этой главе источников. Впрочем, система CUPS поддерживает все эти принтеры очень хорошо.

Если вы не используете систему CUPS и работаете с принтером, поддерживающим язык PostScript, то вы, вероятно, в хорошей форме. Если вы работаете с принтером, не понимающим язык PostScript, попробуйте использовать программу Ghostscript, чтобы преобразовать документы на языках Postscript и PDF в нечто понятное для вашего принтера.

## GDI-принтеры

Система Windows по-прежнему имеет два преимущества, одним из которых является ее поддержка очень старых и дешевых принтеров. Дешевые принтеры, используемые на системах Windows, называются *GDI-принтерами* или *WinPrinter*. Такие принтеры обладают очень незначительным количеством встроенных логических возможностей и не имеют никаких интерпретаторов для языка PDL. Они ожидают, что растеризация будет выполняться на обслуживающем их компьютере.

Часть информации, необходимой для взаимодействия с GDI-принтерами, скрыта в специальном, запатентованном коде Windows. Такая секретность затрудняет разработку поддержки для таких устройств в системе Linux, но сообщество сторонников программного обеспечения с открытым кодом продемонстрировало удивительную способность к воспроизведению структурной схемы и алгоритма работы. Система CUPS поддерживает многие принтеры WinPrinter.

Вторым преимуществом системы Windows является ее поддержка новейших принтеров. Точно так же как новые видео- и аудиокарты, новые принтеры сначала выпускаются с драйверами для системы Windows, которые полностью поддерживают все задокументированные и не задокументированные функциональные возможности данной модели. Драйверы для системы Linux обычно выходят с некоторым опозданием. В случае покупки нового модного принтера с расширенными возможностями, не исключено, что некоторое время им можно будет пользоваться только из Windows.

Старые системы UNIX, которые обычно не поддерживают систему CUPS, имеют с этими принтерами еще больше проблем. Если вы хотите использовать принтер WinPrinter в старой системе UNIX, лучше купите недорогие дистрибутивы систем Mac, Linux или Windows. Вы всегда можете совместно использовать принтер в сети.

## Двусторонняя печать

Дуплексор — это компонент аппаратного обеспечения, который позволяет принтеру выполнять печать на обеих сторонах страницы. У некоторых принтеров он входит в комплектацию, а у некоторых идет как дополнение. Это удобно, потому что экономит и бумагу, и место.

Если у вас нет принтера с возможностью двусторонней печати или желания его купить, можно поступить следующим образом: распечатать на листах бумаги сначала только нечетные страницы, а затем перевернуть листы и распечатать только четные страницы.

Однако сначала следует поэкспериментировать с двусторонним документом, чтобы узнать, как нужно правильно переворачивать листы бумаги, и только потом давать соответствующие инструкции принтеру.

Многие программы печати могут помочь с этим процессом; например, в программе Ghostview (**gv**) есть пиктограммы, позволяющие отмечать подлежащие печати страницы (четные или нечетные), и опция, позволяющая распечатывать только отмеченные страницы. Версии команд **lp** и **lpr** в системе CUPS позволяют делать то же самое при помощи опций **-o page-set=odd** и **-o page-set=even**. В случае частого использования эти опции можно сохранить в "экземпляре принтера" (см. раздел 26.2).

Некоторые, особенно недорогие лазерные принтеры разрабатывались без возможности двусторонней печати. Их производители часто предупреждают о непоправимом ущербе, которым может закончиться для принтера печать на обеих сторонах страницы. Мы никогда не слыхали о том, что такой ущерб действительно имел место, но вряд ли производители стали бы предупреждать нас о нем просто так. Не так ли?

## Другие аксессуары для принтера

Помимо дуплексоров, многие принтеры позволяют добавлять память, дополнительные лотки для бумаги, жесткие диски и другие аксессуары. Эти дополнительные компоненты могут давать возможность распечатывать задания, печать которых в противном случае была бы невозможна, или просто печатать задания более эффективно. При наличии проблем с отправкой заданий на печать следует просмотреть журналы ошибок и выяснить, не поможет ли справиться с этой проблемой увеличение памяти. О журналах ошибок подробнее рассказывается в разделе 26.12.

## Принтеры с последовательным и параллельным интерфейсом

Если принтер подключается прямо к компьютеру через кабель, значит, он использует для соединения либо последовательный, либо параллельный порт.

Несмотря на то что стандарт параллельных интерфейсов немного устарел, он все-таки подразумевает использование портов, конфигурирование которых требует минимальных усилий. Те, кто приобрел принтер с параллельным интерфейсом, скорее всего, смогут установить его без проблем — конечно, если у них на компьютере предусмотрен параллельный порт.

Последовательным портом на оборудовании Mac может служить порт FireWire, но в мире Linux в роли такого порта обычно выступает порт USB. Узнать больше об этом можно, заглянув в базу данных о поддерживаемых USB-устройствах на сайте `linuxprinting.org`.

Вероятность приобретения давно устаревшего принтера, подключаемого через последовательный порт RS-232, очень низка. Но если такое все-таки случилось, чтобы сконфигурировать его, скорее всего, придется приложить массу дополнительных усилий. Для того чтобы очередь печати могла нормально работать с принтером, ей необходимо знать скорость двоичной передачи (baud rate) и другие требующиеся для взаимодействия через последовательный порт параметры. Указать все эти параметры можно только в идентификаторе устройства URI. (Этот процесс подробно описан в руководстве CUPS *Software Administrators Manual*, размещенном в Интернете.) Пожалуй, легче будет купить другой принтер, чем вычислять точно все параметры, требующиеся для того, чтобы заставить работать данный принтер с последовательным интерфейсом.

## Сетевые принтеры

Многие принтеры имеют полнофункциональные сетевые интерфейсы, которые позволяют им находиться прямо в сети и принимать задания через одну и более сетей или протоколов печати. На принтеры, подключенные к сети, данные могут отправляться гораздо быстрее, чем на принтеры, подключенные к последовательному или параллельному порту.

Лазерные принтеры обычно используются как сетевые. Струйные принтеры реже используются в таком качестве. Если вы хотите узнать, есть ли в вашем распоряжении сетевой принтер, загляните в порт Ethernet или посмотрите на беспроводную антенну на задней панели.

## Другие советы

Некоторые проблемы администрирования не связаны с устройством системы печати. Большей частью они возникают из-за того, что принтеры представляют собой ненадежные механические устройства, которые порождают затраты при каждом их использовании.

### *Используйте титульные и завершающие страницы только в случае необходимости*

Система печати может предварять каждое задание страницей с названием задания и сведениями о том, кто направил его на печать (header page), а также дополнять задание завершающей страницей (trailer page). Такие страницы могут оказаться полезными при организации совместной печати на одном принтере, хотя, в основном, это пустая трата времени и бумаги.

Если титульные и завершающие страницы не нужны, запретите их печать в системе BSD, установив булеву переменную `sh`. В системе System V не используйте сценарий, генерирующий такие страницы.

В системе CUPS можно вообще отключить печать титульных и завершающих страниц с помощью графического пользовательского интерфейса или утилиты **`lpadmin`**, а затем включить ее для отдельных заданий.

```
$ lpr -o -job-sheets=confidential gilly.ps
```

Система CUPS позволяет включать печать титульных страниц для отдельных пользователей с помощью утилиты **`lpoptions`**. Можно также создать экземпляр принтера, который будет добавлять титульные страницы в задание (см. раздел 26.2). Система CUPS также позволяет создавать индивидуальную заголовочную страницу, копируя ее из каталога **`/usr/share/cups/banners`** и модифицируя. Сохраните новую страницу среди остальных, присвоив ей новое имя.

### *"Распушивайте" листы*

Принтеры загружают бумагу по одному листу из лотка за один раз. Иногда листы склеиваются, и принтер втягивает два и даже больше листов. Вероятность этой неприятности можно снизить, просто "распушив" стопку бумаги перед тем, как ее загрузить. Держа стопку бумаги за один конец, согните ее и проведите большим пальцем поперек среза, как будто вы тасуете игральные карты. Это просто, бесплатно и полезно.

Некоторые струйные принтеры реагируют на то, как листы положены в лоток. В их настройках должна быть указана предпочтительная ориентация.

### Обеспечьте утилизацию отходов

Все виды компьютерной бумаги пригодны к переработке. Коробки, в которых поставляется бумага, могут быть использованы в качестве контейнеров для ненужных листов. Пометьте коробки, чтобы в них не сбрасывали посторонние предметы (скрепки, скобы скоросшивателей, газеты).

### Выполняйте предварительный просмотр

Часто после распечатки документа обнаруживается незначительная ошибка в форматировании — и приходится все перепечатывать. Ненужного расхода бумаги можно избежать, если инсталлировать специальные программы, позволяющие увидеть готовый к печати документ на экране монитора.

Просто иметь средства предварительного просмотра недостаточно. Ваши пользователи должны уметь их применять. Впрочем, иногда они не хотят этому учиться. Для того чтобы стимулировать их к этому, введите учет, чтобы выяснить, сколько раз перепечатывался один и тот же документ. Это сразу позволит выявить пользователя, который пренебрегает предварительным просмотром.

Средства для предварительного просмотра встроены во многие современные редакторы WYSIWYG, браузеры и системы печати. Для документов другого типа следует использовать иные возможности. Например, для просмотра документов в форматах PostScript и PDF можно использовать программу Ghostview (**gv**). Для системы **roff** направляйте вывод в программу Ghostview, а для языка TeX применяйте утилиты **xdvi**, **kdvi** или Evince.

### Покупайте дешевые принтеры

Современный уровень технологий печати довольно высокий, поэтому на высокопроизводительные устройства с надежной механикой не приходится тратить много средств.

Не гоняйтесь за дорогими принтерами "для рабочих групп", если в этом нет острой необходимости. Разницы в качестве печати никакой, а по производительности и надежности "персональный" принтер среднего уровня во многих случаях ничуть не уступает принтеру старшей модели, не говоря уже о том, что он на несколько килограммов легче. Принтера со скоростью печати 10 страниц в минуту более чем достаточно для пяти штатных клерков или редакторов, так что для группы из 25 сотрудников выгоднее купить пять принтеров за 150 долл., чем один — за 750 долл.

Даже если вы предпочитаете продукцию ведущих компаний, помните, что ни один производитель не является совершенным. У нас был отличный опыт эксплуатации принтеров HP. Это надежные устройства, и компания HP очень агрессивно поддерживает их работу в системах Linux и CUPS. И тем не менее некоторые принтеры HP оказались просто катастрофой. Прежде чем совершать покупку, проконсультируйтесь в Интернете. При этом следует главным преимуществом считать дешевизну: ошибку стоимостью 150 долл. пережить легче, чем ошибку стоимостью 750 долл.

### Держите под рукой запасной картридж

Бледные или белые области на странице, напечатанной на лазерном принтере, означают, что в картридже заканчивается тонер. Купите запасной картридж заранее. В трудную минуту можно просто вынуть картридж из принтера и аккуратно потрясти, чтобы перераспределить оставшийся тонер. Этого часто оказывается достаточно, чтобы напечатать еще несколько сотен страниц.

Полосы и пятна, вероятно, свидетельствуют о том, что вы должны почистить принтер. Проверьте, есть ли у принтера "режим очистки". Если нет или это не помогло, прочитайте инструкции производителя. Большинство картриджей имеет барабан, поэтому попытайтесь поменять картриджи, чтобы узнать, в чем проблема: в принтере или картридже? Если и после этого полосы не исчезнут, отправляйтесь в сервисный центр.

Производители принтеров пытаются предотвратить повторное использование восстановленных картриджей и картриджей со вторичного рынка. Для этого они предпринимают разные меры. Многие устройства используют "именные" расходные материалы, которые распознаются принтером электронным или физическим способом. Даже если два принтера выглядят одинаковыми, например Xerox Phaser 6120 и Konica-Minolta Magicolor 2450, это не значит, что один и тот же картридж можно использовать на обоих принтерах.

Иногда приходится разбирать принтер, чтобы приспособить картридж одного производителя к принтеру другого производителя. Обычно это сопровождается "разведением грязи". Если вы просыпали тонер, втяните его пылесосом как можно быстрее и вымойте это место холодной водой. Вопреки расхожему мнению, тонер лазерного принтера не токсичен, но он состоит из очень мелких частиц и его не следует вдыхать.

Заменив картридж, сохраните коробку и чехол от нового картриджа, чтобы отчитаться. Затем найдите компанию, которая заберет у вас старый картридж.

Использование "именных" расходных материалов стимулировало быстрый рост компаний ("открывай и насыпай"), заново наполняющих старые картриджи за долю цены нового картриджа. Компании, восстанавливающие картриджи, также используют принцип "открывай и насыпай", поэтому вы можете восстановить старый картридж и одновременно заменить его.

Мнения о качестве и сроках службы восстановленных картриджей сильно разнятся. Одна из компаний, известных нам, не желает восстанавливать цветные картриджи или продавать их аналоги, потому что, по их мнению, экономия в этом случае будет меньше, чем стоимость обслуживания принтеров, использующих восстановленные картриджи.

### *Помните о стоимости печати одной страницы*

Самые дешевые принтеры продаются по цене их производства. Производители получают прибыль за счет непропорционально дорогих расходных материалов. На момент написания книги поиск "на скорую руку" выявил, что компания Amazon продает лазерный принтер за 80 долл., а картридж к нему — за 65 долл. Вы можете купить дешевый струйный принтер в магазине Wall-Mart за 50 долл., но вскоре вы будете вынуждены купить набор запасных картриджей, который стоит больше принтера.[2]

Эмпирическое правило гласит: струйный принтер является дешевым, пока вы на нем не печатаете. Лазерный принтер имеет более высокую начальную цену, но его расходные материалы дешевле и служат дольше. Полноцветная страница, напечатанная на струйном принтере, может стоить в 20–50 раз дороже, чем аналогичная страница, напечатанная на лазерном принтере. Кроме того, для струйного принтера требуется специальная бумага, и он работает медленнее, чем лазерный. Чернильные картриджи быстро опустошаются и часто выходят из строя. Чернила часто расплываются, поэтому не используйте струйный принтер для печати кулинарной книги, которая будет использоваться на кухне. С другой стороны, на струйном принтере теперь можно распечатывать фотографии,

[2] Впрочем, помните, что многие недорогие принтеры поставляются с "родным" картриджем, который стоит меньше, чем запасной картридж.

качество которых не уступает специальной фотолаборатории. Лазерные цветные фотографии? Довольно хороши, но сравнивать их нельзя.

Все принтеры имеют уязвимые механические детали. Дешевые принтеры ломаются быстрее.

Иначе говоря, все эти моменты являются предметом компромиссов. Если вы приобретаете принтер для персональных нужд и будете использовать его не очень интенсивно, например распечатывать веб-страницу один-два раза в день или несколько катушек с фотопленкой раз в месяц, то дешевый универсальный струйный принтер — прекрасный выбор.

Следующий раз, когда вы отправитесь покупать принтер, оцените, как долго, насколько интенсивно и для каких целей вы будете его использовать. Вычислите долгосрочную стоимость печати одной страницы для каждого принтера, который вы хотели бы купить. Спросите местную компанию, заправляющую картриджи, восстанавливает ли она картриджи для данного принтера и по какой цене.

### *Организуйте учет использования принтера*

Учет использования принтера даст вам четкое представление о затраченных ресурсах. В сетях среднего и большого размера рассматривайте как вариант организацию учета ресурсов принтера, даже если вы не собираетесь взимать плату за печать. На скорость печати заданий это почти никак не влияет, зато позволяет в любой момент выяснить, кто и как часто пользуется принтером. Информация об источниках заданий на печать очень полезна при выборе мест для установки новых принтеров.

### *Защищайте принтеры*

Большинство современных сетевых принтеров допускает дистанционное управление в том или ином виде. Даже если вы не используете систему CUPS и пароль IPP, возможность такого управления очень удобна для системных администраторов, так как они могут выполнять конфигурирование с помощью веб-браузера через протокол HTTP или, возможно, SNMP. С помощью удаленного интерфейса администратор может задать по сети IP-адрес принтера, стандартный шлюз, сервер Syslog, групповое имя (пароль) SNMP, параметры протокола и, что самое важное, пароль доступа.

По умолчанию большинство принтеров, допускающих дистанционное администрирование, не защищено, и пароль для доступа к ним следует задавать в процессе инсталляции (или строку имени и пароля в протоколе SNMP). Прочтите руководство по инсталляции, предоставленное производителем принтера, чтобы узнать, как это сделать в вашем конкретном случае.

Графические средства администрирования, такие как интерфейс браузера CUPS, все чаще скрывают от пользователя варианты разработчиков. Ожидается, что эта тенденция будет продолжаться.

## 26.12. Советы по выявлению проблем

Принтеры сочетают все слабости механического устройства со странностями внешней операционной системы. Они (и программное обеспечение, которое ими управляет) обожают создавать проблемы для вас и ваших пользователей. Рассмотрим несколько советов по борьбе с враждебными принтерами.

## Повторный запуск демона печати

Никогда не следует забывать перезапускать демоны после внесения изменений в конфигурационный файл.

Перезапустить демона **cupsd** можно любым способом, который операционная система предусматривает для повторного запуска демонов, — выполнить команду **/etc/init.d/cups restart** или аналогичную команду. Теоретически, можно также отправить демону **cupsd** сигнал HUP, но похоже, что на системах SUSE это приводит к уничтожению демона.

Кроме того, демона **cupsd** можно перезапустить с помощью графического интерфейса системы CUPS или посредством другого графического интерфейса, например приложения KDE Print Manager.

В других системах существуют свои специальные методы для перезагрузки системы печати; часто они зависят от разработчиков. Например, в системе AIX используется следующая последовательность команд.

```
$ sudo stopsrc -s lpd
$ sudo startsrc -s lpd
```

Вы так и думали, не так ли?

## Регистрационные журналы

Система CUPS поддерживает три регистрационных журнала: журнал страниц, журнал доступа и журнал ошибок. Журнал страниц представляет собой просто список напечатанных страниц. Журнал доступа и журнал ошибок похожи на аналогичные журналы в веб-сервере Apache. И в этом нет ничего удивительного, потому что сервер CUPS — это веб-сервер.

Уровень регистрации и путь к журнальным файлам задается в файле **cupsd.conf**. Обычно журнальные файлы сохраняются в каталоге **/var/log**.

Ниже приведен фрагмент из журнального файла, охватывающий одно задание на печать.

```
I [26/Jul/2006:18:59:08 -0600] Adding start banner page "none" to job 24.
I [26/Jul/2006:18:59:08 -0600] Adding end banner page "none" to job 24.
I [26/Jul/2006:18:59:08 -0600] Job 24 queued on 'Phaser_6120' by 'jsh'.
I [26/Jul/2006:18:59:08 -0600] Started filter usr/libexec/cups/filter/pstops
     (PID 19985) for job 24.
I [26/Jul/2006:18:59:08 -0600] Started backend /usr/libexec/cups/backend/usb
     (PID 19986) for job 24.
```

## Проблемы с прямой печатью

Удостовериться в наличии физического соединения с локальным принтером можно, напрямую выполнив внутреннюю программу принтера. Например, вот что мы получим, если выполним внутреннюю программу принтера, который подключается через USB-кабель.

```
$ /usr/lib/cups/backend/usb
direct usb "Unknown" "USB Printer (usb)"
direct usb://XEROX/Phaser%206120?serial=YGG210547 "XEROX Phaser 6120"
    "Phaser 6120"
```

Если USB-кабель принтера Phaser 6120 отключится, строка для этого принтера исчезнет из вывода внутренней программы.

```
$ /usr/lib/cups/backend/usb

direct usb "Unknown" "USB Printer (usb)"
```

## Проблемы с печатью в сети

Прежде чем начинать искать проблему печати в сети, проверьте, выполняется ли печать с компьютера, к которому непосредственно подключен принтер. Ваша "проблема печати в сети" может оказаться просто "проблемой печати". Также проверьте, все ли в порядке с сетью.

Далее попробуйте установить соединение с обслуживающим данный принтер демоном **cupsd** при помощи браузера (*имя_узла*:631) или команды **telnet** (**telnet** *имя_узла* **631**).

Сетевой демон **lpd** распечатывает задания, которые он доставил в TCP-порт 515. Если вы не хотите печатать задания от посторонних лиц, ваш брандмауэр должен блокировать весь трафик, поступающий на этот порт из Интернета. Для проверки соединения с удаленным сервером **lpd** примените команду **telnet** к порту 515 на этом сервере. Если можно установить соединение, то, по крайней мере, проверьте работоспособность сети и факт запуска демона **lpd** на сервере.

В случае возникновения проблем при настройке соединения с сетевым принтером, помните о том, что на клиентском компьютере у вас должна быть очередь для заданий, способ для решения того, куда отправлять задание, и метод отправки задания на компьютер, который отвечает за обслуживание очереди на печать. На сервере печати у вас должно быть место для постановки задания в очередь, соответствующие права доступа для разрешения печати задания и способ для вывода данных на устройство.

Для того чтобы выяснить причину этих проблем, возможно, придется заглянуть в несколько мест.

- Системные журнальные файлы на клиентском компьютере (в случае проблем с распознаванием имен и правами доступа).
- Системные журнальные файлы на сервере печати (в случае проблем с правами доступа).
- Журнальные файлы на клиентском компьютере (если отсутствуют какие-то фильтры или каталоги или если имеются неизвестные принтеры и т.д.).
- Журнальные файлы демона на сервере печати (в случае появления сообщений о неправильных именах устройств, неправильных форматах и т.д.).
- Журнальные файлы принтера на компьютере, получившем задание (в случае появления ошибок при передаче задания), как указано в переменной `lf` в файле **/etc/printcap/** в системе BSD.
- Журнальные файлы принтера на компьютере, отправившем задание (в случае появления сообщений о неправильной предварительной обработке или ошибочной постановке задания в очередь).

Проверьте по системной документации, где находятся журнальные файлы. Путь к журнальным файлам системы указан в файле **/etc/syslog.conf**, а путь к журнальным файлам CUPS — в файле **/etc/cups/cupsd.conf**.

## Проблемы с распространением

Любая программа содержит ошибки.[3] В операционной системе Ubuntu, например, система CUPS обновляется каждый месяц. Одни проблемы бывают хуже других, а другие имеют последствия для безопасности.

Например, на более старых версиях Red Hat Enterprise Linux система CUPS постоянно "взламывается". Наиболее правильным решением в таких случаях является обновление операционной системы, но если нет возможности установить более новую версию, можно попробовать установить текущую версию системы CUPS.

# 26.13. Рекомендуемая литература

Каждый дистрибутивный пакет и графический пользовательский интерфейс сопровождаются специальной документацией, посвященной системе печати. В состав пакета KDE входят справочные страницы с описанием команд KDEPrint и руководство *KDEPrint Handbook*. Дополнительную информацию можно найти на сайте `printing.kde.org`. Все эти источники содержат полезные ссылки на другую документацию. (Даже если у вас нет KDE, в документации по KDE вы сможете найти неплохую общую информацию о CUPS.)

Система CUPS поставляется с обширной документацией в формате HTML. Для того чтобы получить к ней доступ, необходимо соединиться с сервером CUPS и щелкнуть на справочной ссылке. Разумеется это не поможет, если у вас нет связи с этим сервером. Впрочем, на вашем компьютере эта документация инсталлируется в каталог **`/usr/share/doc/cups`** в форматах HTML и PDF. Если ее там нет, обратитесь к менеджеру, поставившему дистрибутивный пакет, или на сайт `cups.org`.

В форуме `cups.org` можно задавать вопросы, но сначала следует попробовать разобраться во всем самому и только потом ставить вопросы, причем делать это вежливо.

Если вы работаете в системе Linux, обратитесь на сайт `linuxprinting.org`. Он представляет собой обширную коллекцию различных источников информации по печати в системе Linux и является прекрасным местом для поиска ответов на вопросы. На этом сайте также выложено замечательное учебное пособие по CUPS, в котором имеется раздел, посвященный выявлению и устранению проблем.

Хорошие обзоры системы CUPS размещены на сайтах Википедии и SUSE. Обзор системы CUPS на сайте SUSE находится по адресу: `en.opensuse.org/SDB:CUPS_in_a_Nutshell`.

Если вы хотите иметь напечатанное справочное пособие по системе CUPS, то мы рекомендуем авторитетное руководство по системе CUPS, можно сказать, из первых рук.

Sweet Michael R. *CUPS: Common UNIX Printing System*. Indianaposis, Indiana. Sams Publishing, 2001.

# 26.14. Упражнения

26.1. Найдите кого-нибудь, кто не разбирается в компьютерах, и научите его распечатывать документ PDF в вашей системе. Не испытали ли вы сложности на каком-нибудь этапе? Как облегчить этот процесс для остальных пользователей?

[3] И любую программу можно сократить. Следовательно, продолжая эту мысль, можно сказать, что любую программу можно сократить до одной строки, которая не работает.

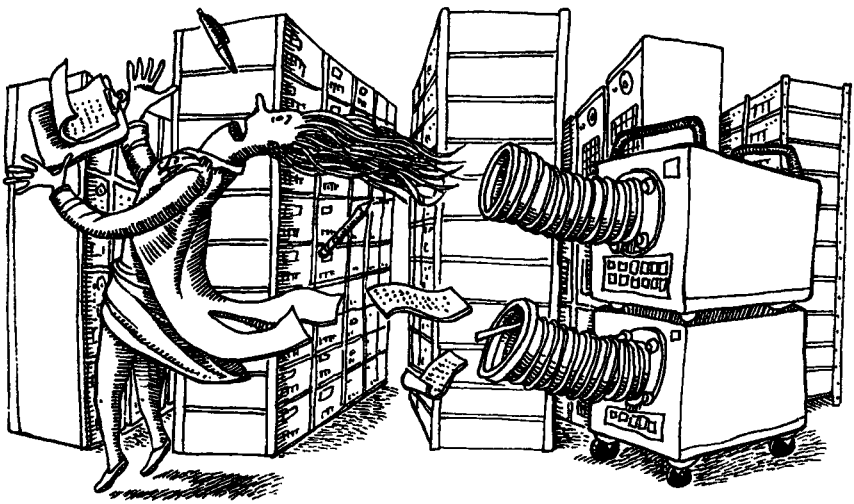
26.2. Используя веб-браузер, зайдите на ваш сетевой принтер. Если вы работаете с системой CUPS, посетите ее веб-сервер с помощью того же самого браузера. Что мешает вам внести изменения в настройки принтеров на этом сервере?

26.3. Зайдите в реальный или виртуальный магазин и выберите три цветных лазерных принтера, которые можно купить дешевле 400 долл. Если вы должны купить принтер завтра, то какой бы вы выбрали и почему? Обратитесь за информацией к базе данных на сайте `linuxprinting.org`.

26.4. Представьте, что вам поручили разработать системное программное обеспечение, которое должно запускаться внутри лазерного принтера, предназначенного для обеспечения работы корпоративных рабочих групп. Какой дистрибутивный пакет вы выберете? Какое дополнительное программное обеспечение вам потребуется? Как учесть потребности клиентов систем Windows и Mac Os X? (Подсказка: проверьте дистрибутивные пакеты для системы Linux, предназначенные для "встроенных систем".)

# Глава 27

# Центры обработки данных



Надежность службы зависит от надежности центра обработки данных, который ее поддерживает. Для специалистов, имеющих практический опыт, это очевидно. Однако высшему руководству центры обработки данных кажутся чем-то далеким и почти иллюзорным.

С появлением настольных рабочих станций и уходом с арены больших железных ящиков, набитых электроникой, показалось, что дни центров обработки данных сочтены. На самом деле в настоящее время потребность в центрах обработки данных еще выше, чем прежде. Они состоят из чрезвычайно важных серверов (часто работающих под управлением операционных систем UNIX или Linux), поставляющих информацию всем желающим получать оперативные данные и интернет-приложения.

Некоторые аспекты центров обработки данных — например, их физическая организация, энергоснабжение и охлаждение — традиционно разрабатывались и поддерживались "техниками". Однако бурное развитие информационных технологий и возрастающая нетерпимость к простоям вынудили специалистов по информационным технологиям и техников стать партнерами при планировании и эксплуатации центров обработки данных. Как системный администратор вы играете важную роль "эксперта по предметной области" в глазах техников.[1]

Центр обработки данных состоит из следующих компонентов.

- Безопасное место.
- Стойки с компьютерами, сетевым оборудованием и дисками.
- Система энергоснабжения, достаточная для работы инсталлированных устройств.
- Система охлаждения для поддержки требуемого температурного режима работы устройств.
- Сетевые соединения внутри центра и с внешними рабочими местами (корпора-

# 27.1. Уровни надежности центров обработки данных

На правильное функционирование центра обработки данных влияют следующие аспекты его организации.

- Устройства бесперебойного электроснабжения (uninterruptible power supplies — UPSs). Устройства UPS обеспечивают электроснабжение при отказе обычного долговременного источника электроэнергии (например, коммерческой электрической сети). В зависимости от размера и емкости, они могут обеспечить от нескольких минут до нескольких часов работы. Сами по себе устройства UPS не могут поддерживать работу сети при долговременном отключении электроснабжения.
- Автономное генерирование электроэнергии. Если коммерческая электрическая сеть не работает, можно подключить автономные генераторы, которые могут заменить долговременные источники электроэнергии. Генераторы обычно заправляются дизельным топливом, сжиженным газом или природным газом и могут поддерживать работу сети, пока не закончится топливо. Целесообразно иметь запас топлива на 72 часа автономной работы и покупать его у разных поставщиков.
- Для работы с автономными генераторами также необходимы устройства UPS, чтобы обеспечить кратковременный период (обычно не превышающий 60 секунд), необходимый для запуска генераторов и перехода от электрической сети на питание от генератора.
- Дополнительные источники электроэнергии. В некоторых организациях существует возможность использовать несколько источников электроснабжения от электрической сети (например, от разных электростанций).
- Механические системы. Эти системы называются HVAC[2], но в контексте центра обработки данных к ним относится только система терморегулирования — тепло для компьютеров не нужно! Для обеспечения работы основной и резервной системы терморегуляции существует множество технологий.

Исследованием вопросов, связанных с управлением центрами обработки данных, занимается промышленная группа Uptime Institute. Ее сотрудники разработали четырехуровневую систему классификации надежности центров обработки данных, которая представлена в табл. 27.1. В этой таблице обозначение *N* означает, что в распоряжении центра есть достаточно ресурсов (устройств UPS, генераторов), чтобы обеспечить нормальную работу. Обозначение *N*+1 означает, что у центра есть одна запасная единица оборудования; 2*N* — что у каждого устройства есть дублирующее оборудование.

**Таблица 27.1. Система классификации, предложенная промышленной группой Uptime Institute**

| Уровень | Генератор | UPS | Источник энергии | HVAC | Коэффициент доступности |
|---|---|---|---|---|---|
| 1 | Нет | N | Один | N | 99.671% |
| 2 | N | N+1[a] | Один | N+1 | 99.741% |
| 3 | N+1 | N+1[a] | Два с возможностью переключения | N+1 | 99.982% |
| 4 | 2N | 2N | Два работающих одновременно | 2N | 99.995% |

[a] С запасными компонентами.

[2] Heating, ventilation and air conditioning, — отопление, вентиляция и система терморегулирования. — *Примеч. ред.*

Центры, относящиеся к высшему уровню надежности, должны быть разделены на отсеки, так чтобы сбой систем электроснабжения или терморегуляции одной группы систем не приводил к отказу другой.

Коэффициент доступности, равный 99,671%, на первый взгляд может выглядеть привлекательным, но это значит, что в течение года система будет примерно 29 часов простаивать. Коэффициент использования, равный 99,995%, означает, что система будет простаивать 26 минут в год.


*Рис. 27.1. С любезного разрешения сайта* xkcd.com

Разумеется, никакое количество резервных источников электроснабжения или устройств терморегуляции не спасет систему, если она плохо управляется или неправильно спроектирована. Центр обработки данных — это основной структурный элемент, но его недостаточно для обеспечения работы всей организации с точки зрения конечного пользователя.

Стандарты работоспособности систем, разработанные группой Uptime Institute, можно найти на ее веб-сайте `uptimeinstitute.org`.

## 27.2. Терморегуляция

Как и люди, компьютеры лучше и дольше работают в благоприятной среде. Поддержка безопасной температуры — основное условие их благополучия.

🕮 Вопросы энергосбережения при эксплуатации центров обработки данных обсуждаются в главе 28.

Американская ассоциация инженеров по отоплению, охлаждению и кондиционированию воздуха (American Society of Heating, Refrigerating and Air-conditioning Engineers — ASHRAE) традиционно рекомендует поддерживать температуру воздуха в центрах обработки данных (измеряемую на воздухозаборниках серверов) от 20 до 25°C. Поддерживая попытки организаций сократить потребление электроэнергии, ассоциация ASHRAE выпустила в 2008 году новые рекомендации, согласно которым температуру воздуха следует поддерживать в диапазоне от 18 до 27°C.

Поддержка требуемой температуры начинается с точной оценки нагрузки на систему терморегуляции. Традиционные модели терморегуляции в центрах обработки данных, взятые из учебников (даже из книг, написанных в 1990-х годах), могут на порядки отличаться от сегодняшних реалий, связанных с функционированием высокоплотных шасси на лезвийных серверах. По этой причине мы рекомендуем перепроверять оценки

нагрузки на систему терморегуляции, полученные вашими инженерами по отоплению, охлаждению и кондиционированию воздуха.

Вы обязаны помочь инженерам HVAC провести вычисления нагрузки на систему терморегуляции, которую оказывают крыша, стены и окна вашего здания (не забудьте о солнечном свете). Инженеры HVAC обычно имеют большой опыт работы с этими компонентами и обязаны дать вам точную оценку. Вам, со своей стороны, следует проверить внутреннюю тепловую нагрузку на ваш центр обработки данных.

Вы должны проверить тепловую нагрузку, которую оказывают следующие компоненты.

- Крыша, стены и окна (от инженеров HVAC).
- Электронное оборудование.
- Осветительная арматура.
- Операторы (люди).

### Электронное оборудование

Для того чтобы определить тепловую нагрузку, которую оказывают ваши серверы (и другое электронное оборудование), следует определить потребление электроэнергии серверами. Получить эту информацию лучше всего с помощью прямых измерений. В этом вам может помочь знакомый электрик. Впрочем, вы можете купить недорогой электроизмерительный прибор и выполнить эту работу самостоятельно.[3] На большинстве оборудования указывается максимальный расход мощности, измеренный в ваттах, но реальное потребление обычно значительно меньше максимума.

Расход мощности можно преобразовать в стандартную единицу количества тепла BTUH (British Thermal Unit per hour — количество британских тепловых единиц в час), умножив его на коэффициент 3,413 BTUH/Вт. Например, если вы хотите построить центр обработки данных, в котором будет 25 центров, потребляющих по 450 Вт каждый, то вычисления будут выглядеть следующим образом.

25 серверов × 450 Вт/сервер × 3,412 BTUH/Вт = 38 385 BTUH.

### Осветительная арматура

Как и в случае электронного оборудования, оценить тепловую нагрузку от осветительного оборудования можно по расходу мощности. Как правило, в офисах в качестве осветительной арматуры используются 40-ваттные люминесцентные лампы. Если в вашем новом центре обработки данных шесть таких ламп, то вычисления будут выглядеть следующим образом.

6 ламп × 160 Вт/лампа × 3,412 BTUH/Вт = 3 276 BTUH.

### Операторы

Иногда людям необходимо войти в центр обработки данных, чтобы выполнить какие-то действия. Предположим, что каждый входящий человек создает тепловую нагрузку величиной 300 BTUH. Допустим, что одновременно в центр обработки данных могут войти четыре человека.

4 человека × 300 BTUH/чел = 1 200 BTUH.

[3] Самым популярным электроизмерительным прибором является Kill a Watt компании P3 стоимостью около 20 долл.

### *Общая тепловая нагрузка*

Вычислив тепловую нагрузку для каждого компонента, просуммируйте их и определите общую тепловую нагрузку. В нашем примере мы предполагали, что инженер HVAC оценил тепловую нагрузку от крыши, стен и окно равной 20000 BTUH.

20000 BTUH для крыши, стен и окон
38385 BTUH для серверов и другого электронного оборудования
3276 BTUH для осветительной арматуры
1200 BTUH для операторов

62861 BTUH всего

Производительность систем терморегуляции обычно измеряется в тоннах. Для того чтобы преобразовать единицы BTUH в тонны, необходимо поделить результат на 12000 BTUH/т. Кроме того, следует допустить, по крайней мере, 50%-ный коэффициент ухудшения, чтобы учесть ошибки и будущий рост системы.

62,681 BTUH × 1т / 12000 × 1,5 = 7,87 тонн охлаждения

Проверьте, насколько ваши оценки расходятся с оценками инженеров HVAC.

## Теплые и холодные отсеки

Вы можете резко уменьшить трудности, связанные с охлаждением центра обработки данных, проанализировав схему его устройства. Самой эффективной стратегией является разделение центра на теплые и холодные отсеки.

Устройства, приподнятые над полом и охлаждаемые традиционными средствами CRAC (кондиционеры воздуха в компьютерных комнатах), часто установлены так, что холодный воздух проходит над полом, поднимается через отверстия в перфорированном покрытии, охлаждает оборудование, а затем, уже в нагретом состоянии, поднимается к потолку и всасывается обратными воздуховодами. Обычно стойки и перфорированные плитки распределяются по центру обработки данных "случайно", и в результате обеспечивается относительно равномерное распределение температуры. Вследствие этого среда становится комфортабельной для человека, но не оптимальной для компьютеров.

Лучше было бы чередовать теплые и холодные отсеки, разделяя их стойками. В холодных отсеках есть перфорированные плитки, а в теплых — нет. Сойки следует расположить так, чтобы оборудование втягивало воздух из холодного отсека, а отдавало — в теплый. Таким образом, стороны выпуска воздуха двух смежных стоек должны располагаться бок о бок. Эта схема изображена на рис. 27.2.

Эта схема оптимизирует поток воздуха так, чтобы воздуховоды всегда втягивали холодный, а не теплый воздух, отработанный на охлаждении соседнего сервера. При правильном воплощении стратегия чередующихся стоек позволяет создать заметно холодные и заметно теплые отсеки. Успех охлаждения можно оценить с помощью инфракрасного термометра, который является незаменимым инструментом современного системного администратора. Это устройство стоит 100 долл. (например, Fluke 62) и постоянно измеряет температуру любого предмета, на который вы его нацелите на расстоянии до 20 метров. Только не доставайте его в баре!

Если вы *обязаны* проводить кабель под полом (см. раздел 27.4), проводите кабели питания под холодными отсеками, а сетевые — под теплыми.

В помещениях без фальшпола могут использоваться междурядные охлаждающие устройства, например APC (`www.apcc.com`). Эти маленькие устройства помещаются между стойками.

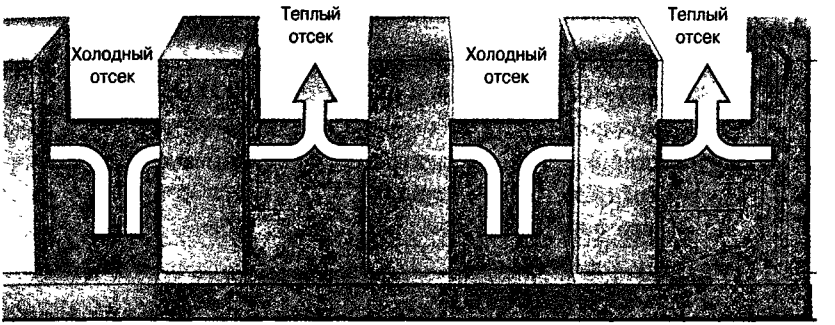



*Рис. 27.2. Холодные и теплые отсеки с фальшполом*

Принцип работы этой схемы показан на рис. 27.3.

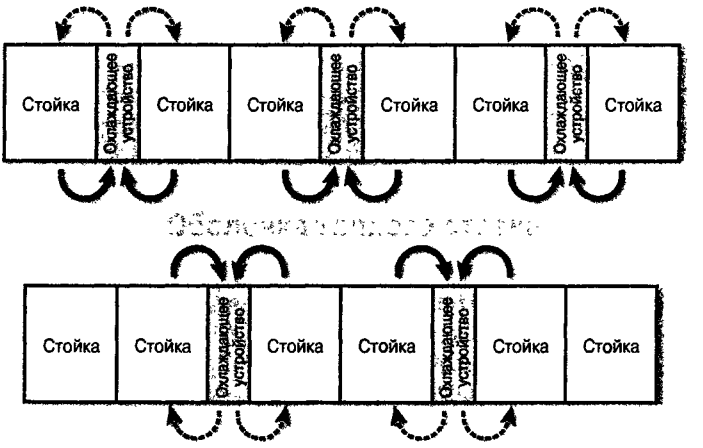



*Рис. 27.3. Холодные и теплые отсеки с междурядными охлаждающими устройствами*

Устройства CRAC и междурядные устройства охлаждения должны иметь возможность выводить тепло за пределы центра обработки данных. Это требование обычно удовлетворяется с помощью циркуляции охлаждающей жидкости (например, охлажденной воды, хладагента Puron/R410A или R22), которая выносит тепло во внешнюю среду. Мы не показали на рис. 27.2 и 27.3 циркуляцию охлаждающей жидкости, чтобы не загромождать рисунки, но в большинстве случаев они необходимы. Некоторые комментарии по поводу использования внешнего воздуха для охлаждения в качестве альтернативы механическому охлаждению читатель найдет в главе 28.

## Влажность

В соответствии с рекомендациями 2008 ASHRAE, влажность воздуха в центре обработки данных должна быть в пределах 30–55%. Если влажность воздуха слишком низкая, возникают проблемы из-за статического электричества. Если влажность слишком высокая, конденсация может создать электрические цепи, которые могут вызвать корот-

кое замыкание и окисление. В зависимости от географического расположения, может возникнуть необходимость увлажнения или осушения оборудования, чтобы поддерживать требуемый уровень влажности.

## Мониторинг окружающей среды

Если вы обеспечиваете работу чрезвычайно важного вычислительного оборудования, необходимо следить за температурой (и другими факторами окружающей среды, такими как шум и электропитание) в центре обработки данных, когда вы там отсутствуете. Вы будете очень огорчены, придя на работу в понедельник и обнаружив расплавленный пластик на полу вашего центра обработки данных.

К счастью, существуют автоматические мониторы, которые могут следить за состоянием окружающей среды в ваше отсутствие. Мы сами используем и рекомендуем вам продукцию семейства Sensaphone (`sensaphone.com`). Эти недорогие устройства для мониторинга следят за такими показателями, как температура, шум и мощность, и звонят вам по телефону или отправляют сообщение на веб-сайт, если возникают проблемы. Фирма Sensaphone находится в Астоне, штат Песильвания (Aston, Pennsylvania). Ее телефон: (610) 558-2700.

# 27.3. Электропитание

Компьютерное оборудование требует ровного и стабильного электропитания. В контексте центров обработки данных это значит, что в нем должен работать, по крайней мере, стабилизатор, фильтрующий всплески напряжения и обеспечивающий требуемый уровень напряжения и фазы.

🕮 Процедуры выключения описаны в разделе 3.7.

Серверы и сетевая инфраструктура должны оснащаться бесперебойными источниками электроснабжения. Качественные устройства UPS имеют интерфейсы RS-232, Ethernet или USB, позволяющие подключать их к компьютеру, который они обеспечивают электричеством, или к централизованной системе слежения, способной обеспечить быстрое реагирование. Такие соединения позволяют пересылать на компьютеры или операторам сообщения об отказе системы электроснабжения и требования выполнить аккуратное выключение компьютеров, пока не разрядились батареи.

Устройства UPS имеют разные размеры и емкости, но даже самое большое из них не может служить долговременным источником электроэнергии. Если переключение на резервные источники электроэнергии занимает больше времени, чем могут работать устройства UPS, необходимо включить локальный генератор.

Существует широкий выбор резервных генераторов, мощность которых колеблется от 5 до 2500 кВт. Золотым стандартом является семейство генераторов компании Cummins Onan (`onan.com`). Большинство организаций выбирает дизельное топливо. Если вы работаете в холодном климате, примите меры, чтобы генератор был заправлен "зимней дизельной смесью", или замените его авиационным топливом Jet A-1, чтобы предотвратить гелеобразование. Дизельное топливо является химически стабильным, но на нем можно выращивать водоросли, поэтому, если вы планируете хранить дизельное топливо долгое время, добавьте в него водоросли.

Генераторы и их инфраструктура — дорогое оборудование, но в некоторых ситуациях они могут сэкономить вам деньги. Если вы установите резервный генератор, то ваше устройство UPS должно быть достаточно большим, чтобы охватить промежуток времени между выключением электричества и запуском генератора.

Если вы используете устройства UPS или генераторы, чрезвычайно важно периодически их тестировать. Мы рекомендуем тестировать все компоненты резервной системы электроснабжения каждые полгода. Кроме того, вы (или ваш поставщик) должны выполнять регламентные работы на этом оборудовании, как минимум, раз в год.

## Требования к электроснабжению стоек

Планирование электроснабжения центра обработки данных — одна из наиболее сложных задач. Как правило, возможность создать новый центр обработки данных или значительно перестроить существующий возникает каждые десять лет, поэтому, обдумывая систему электроснабжения, важно заглянуть далеко в будущее.

Большинство архитекторов вычисляют мощность, необходимую для обеспечения центра обработки данных, умножают ее на его площадь, а затем на поправочный коэффициент. В реальных ситуациях этот подход не эффективен, потому что сам по себе размер центра обработки данных несет мало информации о типе оборудования, которое в нем будет работать. Мы рекомендуем использовать модель энергопотребления в расчете на стойку и не обращать внимания на площадь пола.

Традиционно центры обработки данных проектировались так, чтобы на каждую стойку приходилось от 1,5 до 3 кВт мощности. В настоящее время производители стали упаковывать серверы в стойки 1U и создавать шасси блэйд-серверов, в которых размещаются более 20 модулей, поэтому мощность, необходимая для питания всей стойки современного электронного оборудования, резко увеличилась.

Одно из возможных решений проблемы, связанной с плотностью мощности, — размещать в каждой стойке только несколько серверов 1U, оставляя остальную часть стойки пустой. Несмотря на то что эта технология устраняет необходимость обеспечения дополнительной мощности для стойки, она приводит к чрезмерному растрачиванию места. Лучше разработать реалистичный проект обеспечения мощности, которая может понадобиться для каждой стойки, и подумать, где ее взять.

Требования к мощности у разного оборудования различные, поэтому трудно предсказать, какая мощность понадобится в будущем. Целесообразно создать систему с разными уровнями потребления мощности, в которой на каждом уровне все стойки получают одинаковую мощность. Эта схема полезна не только для удовлетворения потребностей текущих моделей, но и для планирования будущего использования. Некоторые факторы, которые следует учитывать при определении уровней, перечислены в табл. 27.2.

**Таблица 27.2. Модель распределения мощности по уровням для стоек в центре обработки данных**

| Уровень мощности | Ватт/стойка |
|---|---|
| Сверхвысокая плотность [а] | 25 кВт |
| Очень высокая плотность (например, блэйд-серверы)[б] | 20 кВт |
| Высокая плотность (например, серверы 1U) | 16 кВт |
| Диски | 12 кВт |
| Сетевые коммутаторы | 8 кВт |
| Обычная плотность | 6 кВт |

[а] Прогнозируемый верхний уровень в 2015 г.
[б] Текущий верхний уровень в 2010 г.

Распределение мощности на верхних уровнях в табл. 27.2 может показаться слишком щедрым, но его не так трудно достигнуть, даже на современном оборудовании. Компания APC измерила потребление мощности на шасси, содержащем 14 серверов IBM BladeCenter HS20 по 4050 Вт.[4] Шесть из этих шасси в стойке потребляли 24,3 кВт. Без охлаждения этой энергии достаточно, чтобы расплавить примерно 23 кг стали, алюминия или силикона за 15 минут. Не стоит и говорить о том, что для таких конфигураций необходимо специальное охлаждающее оборудование.

Определив уровни мощности, оцените потребности в стойках для каждого уровня. Затем на плане помещения поместите рядом друг с другом стойки, относящиеся к одному и тому же уровню. Такое разделение на зоны концентрирует стойки с высоким потреблением мощности и позволяет правильно распределить ресурсы охлаждения.

### *Киловольт-ампер или киловатт*

Одна из основных причин непонимания между специалистами по информационным технологиям, техниками и инженерами UPS заключается в том, что каждая из этих групп использует разные единицы измерения мощности. Объем мощности, который может обеспечить устройство UPS, обычно измеряется в киловольт-амперах (кВА). Однако инженеры, обслуживающие компьютерное оборудование, и электротехники, обеспечивающие работу центра обработки данных, обычно выражают мощность в ваттах (Вт) или киловаттах (кВт). Возможно, вы помните из курса физики, что ватт=вольт×ампер. К сожалению, ваш учитель забыл напомнить, что ватт — это векторная величина, которая для мощности переменного тока, кроме напряжения (вольт) и силы тока (апмер), содержит "коэффициент мощности" (pf).

Если вы разрабатываете конвейерную линию по разливке пива на пивоваренном заводе, в которой используется много двигателей и другого тяжелого оборудования, то пропустите этот раздел и наймите квалифицированного инженера, который вычислит коэффициент мощности для вашей установки. При работе с современным компьютерным оборудованием вы можете схитрить и использовать константу. Для "приблизительного" преобразования кВА в кВт можно использовать следующие формулы.

кВА = вКт / 0,85

кВт =кВА × 0,85

И наконец, оценивая объем мощности, необходимой для центра обработки данных (или для вычисления размера устройства UPS), следует измерить потребление мощности приборами с помощью амперметра, такого как Fluke 902, а не полагаться на значения, указанные производителем на этикетке (на которой обычно указываются максимальные значения потребления).

## Удаленное управление

Вы можете оказаться в ситуации, в которой необходимо регулярно включать и выключать сервер из-за ошибок в работе ядра или оборудования. Кроме того, возможно, в вашем центре обработки данных установлены серверы, которые работают под управлением другой операционной системы, а не UNIX. В этом случае также часто возникает необходимость их регулярного включения и выключения. В любом случае следует рассмотреть возможность инсталляции системы, позволяющей выполнять эти операции в удаленном режиме.

[4] См. отчет "Power and Cooling for Ultra-High Density Racks and Blade Servers" на сайте apc.com.

Разумным решением является выбор продукции компании American Power Conversion (APC). Продукция MasterSwitch похожа на линейку бесперебойного питания, за исключением того, что ею можно управлять с веб-браузера с помощью встроенного порта Ethernet. Связаться с компанией APC можно по телефону (401) 789-0204 или через вебсайт `apc.com`.

## 27.4. Стойки

Времена традиционных центров обработки данных с фальшполами, в которых кабели электроснабжения, система охлаждения, сетевые соединения и телефонные линии спрятаны под полом, прошли. Можете ли вы отследить кабель, который теряется в подпольном лабиринте? Наш опыт подсказывает, что все это выглядит красиво только на картинке, а на самом деле комната с "классическим" фальшполом — это просто скрытая крысиная нора. Сегодня фальшполы используются только для того, чтобы скрыть электрические кабели, распределить охлажденный воздух, и *больше ни для чего*. Сетевые кабели (и медный, и оптоволоконный) должны проходить по внешним специальным каналам.

В специализированных центрах обработки данных размещение оборудования в стойках (в противоположность, например, расположению на столах или на полу) — единственный профессиональный выбор. Самые лучшие схемы расположения запоминающих устройств используют стойки, связанные друг с другом внешними каналами для кабелей маршрутизации. Этот подход обеспечивает высокую технологичность без потери возможности для сопровождения.

Лучшие внешние каналы для кабелей производятся компанией Chatsworth Products (Chatsworth, CA, (818) 882-8595; `chatsworth.com`). Использование стандартных 19-дюймовых монорельсовых телекоммуникационных стоек позволяет создавать блоки серверов, монтируемых на полках или в стойках.

Две смежные 19-дюймовые телекоммуникационные стойки создают высокотехнологичную разновидность "традиционной" стойки (для ситуаций, в которых вы вынуждены располагать в соседних стойках оборудование, ориентированное относительно передней и задней панелей). Компания Chatsworth поставляет стойки, каналы для кабелей и всякие штучки для кабелей, а также оборудование, необходимое для их монтажа в вашем здании. Поскольку кабели располагаются в доступных каналах, их легко отслеживать и содержать в порядке.

## 27.5. Инструменты

Эффективный системный администратор — это хорошо оснащенный системный администратор. Иметь набор специальных инструментов важно для минимизации времени простоя в случае отказа оборудования. В табл. 27.3 перечислены некоторые инструменты, которые следует включать в такой набор.

**Таблица 27.3. Инструментальный набор системного администратора**

| Универсальные инструменты | |
|---|---|
| Комплект шестигранных ключей (ключей Аллена) | Фигурный молоток, 4 унции |
| Ножницы | Нож электротехника или складной нож |
| Небольшой светодиодный фонарик | Крестообразная отвертка: #0, #1, and #2 |

*Окончание табл. 27.3*

| | |
|---|---|
| **Универсальные инструменты** | |
| Набор торцевых шестигранных ключей | Клещи, игловидные и обычные |
| Детектор неоднородностей | Инспекционная микрокамера Ridgid SeeSnake |
| Рулетка | Щелевая отвертка: 1/8", 3/16" и 5/16" |
| Набор звездообразных ключей Torx | Миниатюрные ювелирные отвертки |
| Миниатюрный пинцет | |
| **Специальные компьютерные инструменты** | |
| Цифровой мультиметр (DMM) | Кабельная стяжка (и их аналоги на "липучках") |
| Инфракрасный термометр | Комплекс винтов PC (например, на сайте `crazypc.com`) |
| Щипцы RJ-45 | Портативный сетевой анализатор/ноутбук |
| Концевые муфты SCSI | Запасные кабели с перекрещивающимися парами категорий 5 и 6A RJ-45 |
| Запасной кабель питания | Запасные разъемы RJ-45 (с твердым ядром и многожильный) |
| Заземляющий браслет для статического электричества | Клещи для удаления изоляции (с встроенным инструментом для резки проволоки) |
| **Разное** | |
| Баллон со сжатым воздухом | Зеркальце дантиста (желательно телескопическое) |
| Мобильный телефон | Комплект первой помощи, включающий ибупрофен и ацетаминофен |
| Изоляционная лента | Номер домашнего телефона и служебных телефонов сотрудников |
| Ватные палочки | Список контактов для срочной связи в случае опасной ситуации[a] |
| | Шесть банок пива (как минимум) |

[a] Номера контрактов на обслуживание.

## 27.6. Рекомендованная литература


- *Telecommunications Infrastructure Standard for Data Centers*. ANSI/TIA/EIA 942.
- ASHRAE Inc. *ASHRAE 2008 Environmental Guidelines for Datacom Equipment*. Atlanta, GA: ASHRAE, Inc., 2008.
- Eubank H., Swisher J., Burns C., Seal J., Emerson B. *Design Recommendations for High Performance Data Centers*. Snowmass, CO:Rocky Mountain Institute, 2003.


## 27.7. Упражнения

27.1. Зачем размещать компьютеры в стойках?

27.2. ✯ Окружающая среда влияет как на людей, так и на компьютеры. Добавьте свои факторы к тем, которые указаны в книге (например, пыль, шум, свет, помехи и др.). Выберите четыре показателя и оцените комфортабельность вашей лаборатории для людей и машин.

27.3. ★ На входе рабочей станции сила тока составляет 0,8 А, на входе монитора 0,7 А, а напряжение равно 120 В.

а) сколько мощности потребляет ваша система в ваттах?

б) допустим, что электричество стоит около 0,12 долл./(кВт·час). Сколько будет стоить круглогодичная работа такой системы?

в) сколько вы сэкономите за год, отключая мониторы на 16 часов в день (вручную или с помощью функциональных возможностей Energy Star, таких как стандарт DPMS)?

г) чему равна годовая стоимость охлаждения такой системы? (Сформулируйте свои предположения и проведите вычисления.)

27.4. ★★ Разработайте проект новой компьютерной лаборатории для вашей организации. Сформулируйте свои предположения о требуемой площади, количестве компьютеров, типе каждого компьютера и потребляемой мощности. Затем определите требования к мощности и системе терморегуляции в вашей лаборатории. Учтите в проекте как серверы, так и клиентские рабочие станции. Приложите схему комнаты, освещения и укажите ожидаемую тепловую нагрузку за счет людей.

# Глава 28

# Экологичные информационные технологии

Может показаться, что книга о системном администрировании — это неподходящее место для главы о защите окружающей среды и социальных вопросах. Однако в настоящее время, когда крупные организации, внедряющие информационные технологии, уже не редкость, вопросы, связанные с их влиянием на окружающую среду и потребление ресурсов, стали привлекать к себе внимание. Экологичные информационные технологии — это искусство и наука о сокращении скрытых и явных затрат ресурсов.

Несмотря на то что каждый из нас может внести маленький вклад в защиту окружающей среды, изменив свои привычки и предпочтения, основное влияние все же оказывают централизованные усилия. Например, никакие попытки следовать лозунгу "Выбирайте неэтилированный бензин! В целом, он намного лучше!" нельзя сравнить с эффектом от принятия федерального закона, останавливающего производство автомобилей, использующих свинец. Угадайте, кто может отдать подобный приказ вашей организации? Вы!

Однако зачем же беспокоиться? Например, чтобы испытывать гордость за то, что ты делаешь полезное дело для всей планеты, хотя существуют и практические причины, чтобы убедить руководителей вашей организации внедрить "зеленые" информационные технологии.

- **Снижение первоначальных издержек.** Минимизируя количество оборудования, приобретаемого и используемого вашей организацией, вы снижаете капитальные затраты. Минимизируя размер центра обработки данных, вы снижаете стоимость недвижимости.

- **Снижение эксплуатационных расходов.** Электроэнергия, управление и эксплуатация оборудования стоят денег. Эффективное использование меньшего количества устройств означает снижение прямых издержек вашего предприятия.
- **Экономия накладных расходов.** Вы платите за электричество дважды: один раз — за электроснабжение оборудования, а второй — за охлаждение оборудования после того, как оно преобразует дорогую энергию в тепло.[1] Чем меньше оборудования, тем меньше его требуется охлаждать, тем меньше нужно ему площади и меньше людей для обслуживания. Чем меньше людей, тем меньше затраты на аренду и кондиционирование офиса, меньше объем заработной платы, премий и материальной помощи.

В главе рассматриваются основные концепции, позволяющие уменьшить потребление электроэнергии и ресурсов вашей организацией. Мы имеем в виду организации, в центрах обработки данных которых работают от 1 до 500 серверов. Если ваша организация крупнее, следует нанять эксперта по проектированию экологичных центров обработки данных, чтобы достичь наибольшего эффекта.

## 28.1. Введение в экологичные информационные технологии

Что мы имеем в виду под "экологичными" технологиями?

- Более низкое потребление электроэнергии.
- Более скромные требования к оборудованию.
- Снижение потребления расходных материалов.
- Вторичная переработка отходов.

В области экологичных информационных технологий нет волшебного решения или единственно возможного пути решения проблем. Несмотря на заверения поставщиков, вы не можете купить одну продукцию, которая решила бы все экологические проблемы в мире. В частности, экологичные информационные технологии — это нечто большее, чем просто серверная виртуализация. Аналогично тому как системное администрирование имеет много аспектов, внедрение экологичных информационных технологий — это больше процесс, чем цель. Сначала необходимо представить себе, чего вы хотите достичь, наметить план достижения цели, а затем воплотить его в жизнь. Ключевыми элементами в этом плане должны стать постоянные измерения и мониторинг.

Для начала оцените экологичность вашей среды. Сделайте полный обзор всех информационных мощностей вашей организации, и не только для того, чтобы повысить эффективность вашего проекта, но и чтобы убедить людей, что вы не просто сотрясаете воздух. Например, с экологической точки зрения было бы замечательно удалить *все* серверы, но вы скоро поймете, что исключение 50 управляемых вами серверов приведет к тому, что ваши пользователи в ответ купят и установят 600 неконтролируемых серверных систем в своих комнатах.

В начале своих экологических исследований необходимо собрать следующую информацию.

[1] Информационная работа, выполняемая компьютерным оборудованием, не является значительной величиной с точки зрения термодинамики. Компьютеры, по существу, представляют собой эффективное устройство, которое преобразует 100% электроэнергии в тепло.

- **Учет оборудования.** Следует учесть все, включая серверы, ноутбуки, мониторы, принтеры, запоминающие устройства, сетевое оборудование, резервное оборудование, устройства UPS и модули охлаждения. Определите их местоположение, номер модели, "размер" (в единицах, характерных для выбранного оборудования), а также возраст.
- Полезно также узнать уровень потребления мощности для каждой единицы оборудования. Номинальный уровень потребления мощности может исказить реальное положение дел. Лучше измерить фактическое потребление энергии с помощью прибора Kill a Watt, который стоит около 20 долл.[2] Для устройств, имеющих активный и спящий режим (например, принтеров), можно записать средний уровень потребления энергии за день или неделю.
- **Учет расходных материалов.** К ним относятся бумага, тонер, диски.
- **Показатели работы организации.** Эти показатели включают в себя валовой доход, количество сотрудников, количество рабочих мест, общий объем потребления энергии, объем энергии, потребляемой компьютерным оборудованием (в центрах обработки данных), уровень потребления энергии системами терморегуляции в центре обработки данных, общий объем капиталовложений в информационные технологии, общий объем эксплуатационных издержек, связанных с информационными технологиями, и общую стоимость оборудования для центров обработки данных.

Собрав эти данные, определите от одной до трех целей для оптимизации. Эти цели должны быть связаны с общей стратегией развития вашего предприятия и демонстрировать прогресс в направлении более экологичных информационных технологий. Мы не можем указать, какая цель может оказаться наиболее подходящей для вашей организации, но приведем несколько примеров.

- Уровень потребления энергии центром обработки данных в расчете на доллар валового дохода.
- Количество сотрудников в расчете на один сервер.
- Количество листов бумаги, использованной сотрудниками за месяц.
- Среднее потребление энергии оборудованием, расположенным на рабочем месте сотрудника.
- Средний срок службы ноутбука.
- Уровень потребления энергии центром обработки данных по отношению к общему потреблению энергии всем оборудованием.[3]

Выполняйте повторную оценку состояния вашего компьютерного оборудования не реже одного раза в год, а уровень потребления мощности измеряйте ежемесячно.

---

[2] Этот прибор разработан для североамериканского рынка, но аналогичную продукцию можно найти и на других рынках. Например, аналоги для Великобритании можно найти на сайте `reuk.co.uk/Buy-UK-Power-Meter.htm`.

[3] Этот показатель, умноженный на 100, равен процентной доле мощности оборудования, направленной на потребление компьютерным оборудованием. В промышленности этот показатель называется DCiE. Это стандартный показатель, который можно использовать для сравнения организаций. Эффективность потребления мощности (PUE — power usage effectiveness) — это показатель, обратный к показателю DCiE. Он используется для оценки эффективности крупных центров обработки данных.

## 28.2. Экологическая пирамида

Понять, насколько экологичной является ваша организация, достаточно легко. Труднее достичь (и измерить) ее прогресс в направлении улучшения окружающей среды. Для того чтобы вы ориентировались в возможностях, описанных в этой главе, мы разделили стратегии экологического развития компьютерных технологий на три категории, как показано на рис. 28.1.


*Рис. 28.1. Подходы к экологичным информационным технологиям*

Мы представили эти категории в виде пирамиды, поскольку стратегии, указанные внизу, имеют наибольший эффект и чаще обеспечивают побочную выгоду. По мере подъема по пирамиде, стратегии становятся более дорогостоящими, а усилия менее эффективными.

Поиск путей к экологичным информационным технологиям всегда следует начинать с сокращения прямого потребления — лучше меньше да лучше. Если вы можете достичь своей цели с меньшими усилиями и затратами ресурсов, сократите капитальные и эксплуатационные расходы.

Второй по качеству является стратегия уменьшения вторичного потребления. Например, охлаждение, необходимое для работы серверов, относится к вторичному потреблению, поскольку оно возникает как следствие существования сервера. Оптимизация системы HVAC для минимизации затрат на охлаждение экономит деньги, но это не может сравниться с экономией от полного отказа от сервера.

Возможно, это не логично, но выбор продукции и технологий, называемых экологичными, мы отнесли к последней категории. Это можно описать такой аналогией: сначала мы уменьшаем количество автомобилей на дорогах до минимально возможного и только потом заменяем оставшиеся автомобили эффективными моделями.

## 28.3. Стратегии разработки экологичных информационных технологий: центр обработки данных

Центры обработки данных представляют собой превосходные объекты для экологических инициатив, потому что они обычно работают по принципу 7×24×365 и находятся под непосредственным контролем специалистов под информационным технологиям.

Исследование компании Lawrence Berkeley Laboratories показало, что центры обработки данных могут потреблять в 40 раз больше энергии, чем обычные офисы.[4]

При таком уровне потребления необходимы специальные стратегии. Как показано на рис. 28.2, стратегии сокращения прямого потребления, расположенные внизу пирамиды, являются самыми эффективными. В принципе, нет необходимости использовать в конкретной ситуации сразу все стратегии, но каждая из них может внести свой вклад.


*Рис. 28.2. Экологичные стратегии для центров обработки данных*

## Консолидация приложений

Со временем организации и департаменты информационных технологий стремятся аккумулировать приложения. Новые приложения поступают для поддержки специфических деловых инициатив и игрушечных проектов директоров, но старые приложения исчезают редко. Чаще они долго не уступают дорогу, и никто не берет на себя риск полностью их устранить. Какой бы ни была причина, в устоявшейся организации основной тенденцией является стремление к консолидации приложений до минимального набора, удовлетворяющего текущие производственные потребности.

Рассмотрим пример организации, в которой эксплуатируется три приложения: EmployeeLinq, AccountAwesome и ElectricClockster. Несмотря на то что это упрощенный пример, он похож на реальную ситуацию, которая сложилась в одной из изученных нами организаций. Каждое из этих приложений имеет сервер базы данных, сервер приложения и веб-сервер. В сумме получаем девять серверов, поддерживающих эти три приложения.

На первом этапе в направлении консолидации следует составить список функций, выполняемых каждым приложением. Функции, выполняемые приложениями из нашего примера, перечислены в табл. 28.1. Как видим, некоторые функции дублируют друг друга.

**Таблица 28.1. Функциональные возможности трех приложений**

| Функция | EL | AA | EC |
|---|---|---|---|
| Кредиторская/дебиторская задолженность | | X | |
| Управление прибылью | X | X | |
| Временные данные персонала | X | X | X |

[4] *См. сайт* `eetd.lbl.gov/emills/PUBS/PDF/ACEEE-datacenters.pdf`.

*Окончание табл. 28.1*

| Функция | EL | AA | EC |
|---|---|---|---|
| Главная бухгалтерская книга | | X | |
| Расчет заработной платы | X | | X |
| Оценка времени | X | X | X |
| Учет отпусков и отгулов | X | X | |

Эта организация имела три системы для отслеживания времени, две системы — для расчета заработной платы (хотя в каждый момент времени использовалась только одна из них) и многие другие дублирующие функции.

Такая ситуация сложилась потому, что три отдела — финансовый, кадров и производственный — использовали свое собственное приложение. Это не только привело к растрате энергии и компьютерных ресурсов, но и усложнило или вообще сделало невозможным интеграцию данных между отделами. В данном случае переход организации на одно прикладное приложение привело к сокращению аппаратного обеспечения и затрат энергии на 60%, при этом потоки данных внутри компании стали более согласованными.

Ваша ситуация, возможно, выглядит не так трагично, но если вы потратите время на составление таблицы функций, то, вероятно, обнаружите их дублирование.

Обосновать консолидацию приложений очень легко, потому что проектируемые результаты (по крайней мере, частично) можно выразить в сэкономленных долларах. Интеграция данных и улучшение функционирования предприятия — это просто дополнительное преимущество.

## Консолидация серверов

В большинстве организаций работает, по крайней мере, несколько узкоспециализированных серверов, уровень использования которых составляет 10% или меньше. Например, мы видели много организаций, в которых были выделенные серверы NTP (network time protocol). NTP — это низкоуровневый протокол, требующий небольших вычислительных усилий. Резервирование сервера для протокола NTP напоминает перелет Боинга 767 через всю страну с одним пассажиром на борту.

Консолидация серверов очень напоминает консолидацию приложений и является не менее эффективной, только вместо концентрации многих функций в одном приложении мы концентрируем несколько служб на одном серверном компьютере.

В отличие от системы Windows, системы UNIX и Linux изначально превосходно реализуют режим приоритетной многозадачности. Хорошим решением в случае протокола NTP является запуск демона NTP на одних и тех же серверах, обеспечивающих работу таких общих инфраструктурных служб, как DNS и Kerberos.[5]

Еще одну часто возникающую возможность консолидации серверов предоставляют серверы баз данных, предназначенные для работы с одним приложением. Если у вас работают компетентные системный администратор и администраторы баз данных (а также есть хорошие средства мониторинга), то отдельный сервер баз данных может хранить

[5] Протокол NTP — это особый случай, в котором задержка ответа может быть небольшой. Однако это не значит, что на той же самой машине вы сможете запустить другие службы. К доменам сервера NTP обычно применяется утилита nice, чтобы при необходимости предоставить им доступ к центральному процессору (см. раздел 5.6). Аналогичный результат — и даже более надежный — можно получить с помощью серверной виртуализации.

базы данных для многих приложений. Эта консолидация также снижает стоимость лицензий, капиталовложения и потребление электроэнергии.

В некоторых ситуациях сократить количество серверов можно, заменив старые и менее мощные серверы меньшим количеством новых и более мощных серверов, эффективно потребляющих электроэнергию.

## Сеть SAN

Признаком "ненасытности" информационных технологий является парк серверов, заполненных жесткими дисками. Представьте себе центр обработки данных, состоящий из 100 серверов, каждый из которых содержит шесть дисков емкостью один терабайт. Эти 600 дисков необходимо произвести, обслужить, снабдить электроэнергией, а потом вынуть и выбросить. Вероятность того, что средняя степень использования этих устройств превысит 50%, равна нулю.

Этот подход приводит к чрезмерным расходам, потому что он подразумевает разделение запоминающих устройств на изолированные части, которыми невозможно эффективно управлять, так чтобы для каждого сервера или приложения выделялся "правильный" объем памяти. На некоторых серверах может храниться меньше одного терабайта данных, в то время как другие серверы могут оказаться перегруженными, не имея возможности использовать простаивающие диски на соседних шасси. В типичном центре обработки данных с изолированными дисками на каждом сервере трудно достичь хотя бы 30% использования запоминающих устройств.

Хорошей альтернативой этому подходу является сеть хранения данных, или SAN; она описана в разделе 8.11. Технология SAN обеспечивает высоконадежное хранение данных, которое одновременно является экологичным, потому что системные администраторы могут эффективно выделять память на централизованных запоминающих устройствах. Многие организации превысили 90%-ный уровень использования своих сетей SAN. Это в три раза выше эффективности использования изолированных дисков. Сейчас, когда сети SAN функционируют на интерфейсах Ethernet, больше нет препятствий для развертывания этого превосходного инструмента.

## Серверная виртуализация

Одной из самых популярных экологичных информационных технологий является серверная виртуализация, впрочем, частично эта популярность подогревается рекламой, которая оплачивается компаниями, производящими платформы для виртуализации.

Серверная виртуализация (подробно описана в главе 24) — действительно фантастический инструмент. Ее влияние на экологию можно сравнить с консолидацией серверов. В обоих случаях на одном компьютере выполняется несколько приложений или служб. Виртуализация сокращает потребление электроэнергии за счет уменьшения количества шасси, задействованных в производстве, и более эффективного использования остального оборудования.

Виртуализация предлагает дополнительные функциональные возможности, которых нет у консолидации, например возможность легко масштабировать идентичные системы, резервировать часть мощности аппаратного обеспечения для конкретного сервера, а также переносить виртуальные серверы на другие физические шасси. Эти аспекты виртуализации являются ее преимуществами.

Однако у виртуализации есть и недостатки. Приложения с интенсивным вводом-выводом обычно плохо виртуализируются и медленнее работают в виртуализированной

среде. Процесс виртуализации сам по себе требует ресурсов, поэтому виртуализированные системы имеют накладные расходы, которых нет у физических систем. Дополнительные уровни абстракции, создаваемые из-за виртуализации, вызывают настороженность у некоторых системных администраторов, потому что самой виртуализацией необходимо активно управлять и она может влиять на функционирование систем, расположенных на конкретном узле.

Виртуализацию лучше всего проводить в компаниях с хорошо подготовленными специалистами в области информационных технологий и устоявшимися бизнес-процессами. На данный момент мы не рекомендуем осуществлять серверную виртуализацию начинающим системным администраторам. Однако эта технология быстро совершенствуется и становится более простой. Скоро она станет совершенно необходимой.

## Только самые необходимые серверы

При использовании только необходимых серверов, те серверы, которые в данный момент не используются, выключаются. Этот подход лучше всего зарекомендовал себя в ситуациях с предсказуемым циклическим потреблением электроэнергии; например, когда сервер связан с циклом бухгалтерских вычислений или работой, которая выполняется только ранним утром. Этот метод не получил широкого распространения, но иногда внедрить экологичные информационные технологии можно только с помощью этого трюка.

Для того чтобы реализовать эту технологию, нужно иметь несколько схем и удлинители, соединенные с интерфейсом Ethernet (управляемые). Такие платформы, как RightScale (`rightscale.com`), распространили эту концепцию на системы с повышенными требованиями. С ее помощью можно установить пороговые значения, достигнув которые дополнительные серверы автоматически включаются (или выключаются) в соответствии с нагрузкой центрального процессора или объемом трансакций.

## Использование детализированных данных и планирование потребления

В области экологичных информационных технологий, как и в других областях, можно управлять только тем, что можно измерить. Тщательный сбор данных играет важную роль в процессе оптимизации вашей среды.

Если вы регистрируете использование ресурсов вашей сети, например загрузку центральных процессоров и памяти (см. главу 29), можете планировать установку нового аппаратного обеспечения так, чтобы не покупать сервер "про запас".

Мониторинг и анализ требуют времени, но они позволяют правильно организовать управление работой центра обработки данных. Покупайте только то, что вам необходимо; используйте только то, что должны использовать.

## Конфигурация серверов, оптимизированная по потреблению энергии

Некоторые системы позволяют сэкономить энергию, управляя работой самой системы.

### *Возможности экономии энергии в системе Linux*

Центральные процессоры и ядра центральных процессоров можно останавливать, чтобы сократить потребление энергии. Для того чтобы свести потребление энергии к

минимуму, следует упаковать как можно больше потоков в одном ядре или центральном процессоре и не активизировать дополнительные ядра и центральные процессоры, пока они не понадобятся. И наоборот, чтобы достичь максимальной производительности, следует распределить потоки среди ядер и центральных процессоров как можно шире, чтобы минимизировать время простоя за счет переключения и поочередного использования кеша. Теоретически, следует пожертвовать частью производительности в пользу уменьшения потребления электроэнергии.

На практике возможность отключения части центрального процессора возникает только тогда, когда система не загружена. В этой ситуации дополнительные усилия на упаковку потоков в одном ядре могут не привести к осязаемому эффекту. Поэкспериментируйте, чтобы увидеть хоть какую-нибудь разницу на фоне вашей конкретной рабочей нагрузки.

Система управления потреблением электроэнергии, которая является частью планировщика процессов, проверяет две управляющие переменные. Обе они устанавливаются в файлах, расположенных в каталоге **/sys/devices/system/cpu**.

Переменная **sched_mc_power_savings** контролирует, все ли ядра центрального процессора используются, прежде чем активизировать другой центральный процессор, а переменная **sched_smt_power_savings** контролирует, все ли слоты потоков в ядре используются, прежде чем активизировать другое ядро. В обоих случаях значение 0 отключает режим экономии электроэнергии, а значение 1 включает его.

Например, для того чтобы включить оба режима экономии электроэнергии, можно выполнить следующие команды.

```
$ sudo sh -c 'echo 1 > /sys/devices/system/cpu/sched_mc_power_savings'
$ sudo sh -c 'echo 1 > /sys/devices/system/cpu/sched_smt_power_savings'
```

Для того чтобы эти изменения применялись при перезагрузке, выполните команду **sysctl** или добавьте эти строки в стартовый сценарий, например **/etc/init.d/local** в системах Ubuntu и SUSE (создайте его, если необходимо) или **/etc/rc.local** в системе Red Hat.

Центральный процессор компьютера — один из самых расточительных потребителей электроэнергии (его можно считать поглотителем тепла!), поэтому агрессивное управление его функционированием может значительно снизить потребление электроэнергии системой.

### *Экономия электроэнергии, потребляемой файловыми системами*

Сэкономить электроэнергию и повысить производительность можно, запретив файловым системам записывать "время последнего обращения" (st_atime) к каждому файлу. Эта информация не очень полезна и теоретически добавляет к каждой файловой операции одну операцию позиционирования и одну операцию записи. (Оценить ее влияние в реальных условиях намного труднее из-за блочного кеширования.)

В 2003 году Зедлевски и соавторы (Zedlewski et al.) проанализировали потребление электроэнергии дисками и пришли к выводу, что на диске IBM Microdrive на операцию позиционирования затрачивается 4 миллиджоуля; для стандартных дисков с более крупными деталями эту величину следует, по меньшей мере, удвоить. Суммируя затраты на позиционирование и запись, можно вычислить выгоду от запрета записи "времени последнего обращения", которая составит несколько киловатт на один диск в год. Это не много, но заслуживает внимания с точки зрения повышения производительности; экономия электроэнергии в данном случае представляет собой побочный эффект.

В большинстве файловых систем запретить запоминать время последнего обращения можно с помощью параметра **noatime** команды **mount**.

```
$ sudo mount -o remount,noatime /
```

Некоторые системы семейства Linux также поддерживают параметр **relatime** команды **mount**, обеспечивая гибридные функциональные возможности. В этом случае время последнего обращения обновляется, только если предыдущее значение предшествует моменту модификации файла. Этот режим позволяет таким инструментам, как программа для чтения почты, правильно распознавать ситуации, в которых интересующий их файл был изменен, но еще не прочитан.

## Облачные вычисления

🕮 Облачные вычисления описываются в разделе 24.1.

Сделайте глубокий вдох и посмотрите на проблему с иной точки зрения. Недавно появившаяся возможность "облачных вычислений" принесла массу выгод, одной из которых является экономия электроэнергии. Стремясь организовать дешевые и надежные службы, провайдеры наподобие компании Amazon создали сверхэффективные центры обработки данных и процессы для управления загрузкой оборудования. Эти облачные провайдеры могут предоставить компьютерные циклы, которые являются более экологичными, чем любой самый эффективный центр обработки данных, который вы можете спроектировать самостоятельно.

Если нет острой необходимости, чтобы ваши приложения (особенно веб-приложения) функционировали только в вашем здании, рассмотрите возможность перенести их инфраструктуру в центр облачных вычислений. Вы будете по-прежнему обладать полным административным контролем над виртуальными системами, которые будут функционировать в этой среде. Просто вы никогда не сможете их физически "пощупать".

## Свободное охлаждение

Представьте себе холодный зимний день. Вы прогуливаетесь вокруг центра обработки данных и видите, как, работая на полную катушку, кондиционер выводит тепло в атмосферу. Снаружи 10 градусов мороза, но совершенно очевидно, что инженер HVAC разрабатывал систему терморегуляции, использующую механическое охлаждение (и значительный объем электроэнергии), чтобы выводить тепло за пределы центра обработки данных независимо от температуры окружающей среды.

К счастью, некоторые современные инженеры HVAC, специализирующиеся на проектировании центров обработки данных, предлагают лучшее решение этой проблемы: использовать для охлаждения оборудования внешний воздух, если его температура достаточно низкая.

Разумеется, это решение подходит не везде и не всегда. Консорциум технологических компаний Green Grid, специализирующийся на повышении эффективности использования электроэнергии центрами обработки данных, выпускает карты "свободного охлаждения" для Северной Америки и Европы, на которых показано, сколько часов в год можно охлаждать внешним воздухом центры обработки данных в данной местности. Существует также более подробный калькулятор охлаждения, который можно найти на сайте `thegreengrid.org`.

## Эффективное охлаждение центра обработки данных

Для сокращения потребления электроэнергии системой охлаждения центра обработки данных существует несколько способов. Например, можно применить описанную выше схему теплых и холодных отсеков, которая концентрирует охлаждение там, где это необходимо, и позволяет другим частям вычислительного центра функционировать при более высокой температуре.

Более подробное обсуждение некоторых из этих советов изложено в главе 27.

## Режим ограниченной функциональности во время простоя

Многие организации зациклены на эксплуатационной готовности (например, Uptime). При этом они часто не учитывают дополнительные затраты энергии и ресурсов, необходимые для того, чтобы достичь требуемого уровня эксплуатационной готовности.

Внутренние потребители привыкают думать о услугах как о чем-то таком, что либо есть, либо нет. Представьте себе услуги с *ухудшенными* характеристиками как дополнительную возможность управления простоями и спросите себя, как это может удовлетворить потребности клиентов.

Например, вместо запуска абсолютно избыточного набора оборудования для каждой производственной среды, можно применить серверную виртуализацию и развернуть несколько приложений на одном шасси для работы во время простоя. Эта конфигурация может иметь все стандартные функциональные возможности, но работает медленнее, чем обычно. В некоторых ситуациях этот компромисс может сэкономить до 50% капитальных затрат предприятия и даже больше.

## Продление срока эксплуатации оборудования

Производство электроники потребляет энергию и генерирует ядовитые отходы, поэтому покупка нового оборудования влечет нагрузку на окружающую среду, которая не всегда отражается на ценнике. К сожалению, промышленность настолько привыкла к быстрым инновациям и появлению новых изделий, что производители часто прекращают поддержку своей продукции уже через несколько лет после ее выпуска.

Если ваше оборудование удовлетворяет ваши потребности и потребляет разумное количество электроэнергии, то подумайте о стратегии, основанной на продлении срока его эксплуатации.[6] Такая схема обычно сопровождается поисками на аукционе eBay и в других источниках дешевых подержанных запасных частей для аналогичных систем и превращением своего рабочего места в выставку антиквариата. Это позволяет продлить функционирование системы на два-три года, хотя в одном случае нам таким образом удалось заставить систему проработать на восемь лет больше положенного срока.

Если старое оборудование не удовлетворяет требованиям, предъявляемым к уровню производительности, или для него нет запасных частей, можно купить новое оборудование для производственного отдела, а старое передать отделу проектирования, где производительность и надежность не так важны. Этот подход не позволяет вообще избежать покупки нового оборудования, но откладывает покупку для отдела проектирования на один-два года.

[6] Если же ваше оборудование не является эффективным с энергетической точки зрения, то лучше заменить его немедленно, чтобы сэкономить на потреблении электроэнергии, даже с учетом стоимости его размещения и замены.

Если оборудование просто устарело, примите меры, чтобы передать его компании, занимающейся утилизацией компьютеров, которая разберет его на запасные части и каждую из них соответствующим образом утилизирует. Убедитесь, что у этой компании есть сертифицированная программа утилизации, чтобы ваши данные не попали кому-то еще.

Компьютеры содержат удивительно много ядовитых веществ. Что бы ни случилось, не выбрасывайте компьютер в мусорный контейнер, поскольку оттуда он, скорее всего, попадет на свалку, не предназначенную для хранения электронных устройств.

В некоторых регионах существуют организации, бесплатно осуществляющие утилизацию компьютеров. Например, в Портленде, штат Орегон, существует программа утилизации (`freegeek.org`).

## Более высокая температура в центре обработки данных

Примерно треть электроэнергии, потребляемой обычным центром обработки данных, уходит на охлаждение. Ранее температура воздуха в центрах обработки данных поддерживалась в диапазоне от 20 до 25°C. Сейчас эти оценки кажутся слишком консервативными.

В начале 2009 года Американская ассоциация инженеров по отоплению, охлаждению и кондиционированию воздуха (American Society of Heating, Refrigerating and Air-conditioning Engineers — ASHRAE) выпустила руководство, в котором указан приемлемый диапазон температуры для центров обработки данных от 18 до 27°C. Повышение температуры воздуха в центре обработки данных на три градуса экономит примерно 12% стоимости охлаждения.

Дополнительные советы по системе охлаждения можно найти в главе 27.

## Энергосберегающее оборудование

Покупая новое оборудование, посвятите время выбору продукции с минимальным влиянием на окружающую среду.

Организация IEEE стандартизировала критерии для оценки влияния электроники на окружающую среду в публикации IEEE P1680. Оценочная система, основанная на документе IEEE P1680, Electronic Products Environmental Assessment Tool (EPEAT), учитывает большой спектр потенциальных факторов, которые могут иметь место в производстве. Это позволяет осуществлять правильную оценку продукции. В настоящее время эта система охватывает производство настольных компьютеров и ноутбуков, "тонких клиентов", рабочих станций и компьютерных мониторов. Ее оценки являются обязательными при осуществлении правительственных закупок. (Посетите сайт EPEAT `epeat.net`.)

Обратите внимание на то, что от потребления электроэнергии система EPEAT требует согласованности со стандартом Energy Star (версия 5.0, 1 июля 2009 года).

Некоторые производители серверов (включая Dell, Sun, IBM и HP) предлагают экологически ориентированные линейки продукции. Однако даже экологичные серверы влияют на окружающую среду и потребляют энергию. Существование такой продукции еще не дает право приписывать к названию оборудования слова "экологичное". *В первую очередь* следует определить количество необходимых серверов *и только затем* выбирать самые экологичные.

## 28.4. Экологические информационные стратегии: рабочее место пользователя

Еще одна возможность для внедрения экологичных технологий — рабочее место пользователя. Некоторые подходы к реализации этих технологий проиллюстрированы на рис. 28.3.

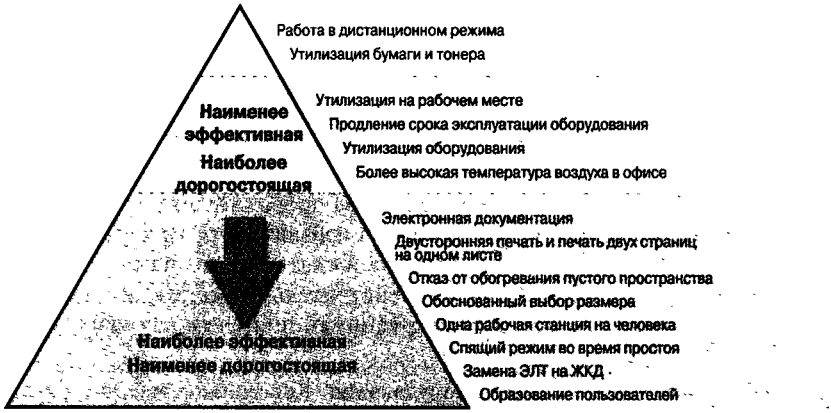



*Рис. 28.3. Стратегии для создания экологичных рабочих мест пользователей*

Ниже указано, на что может влиять экологичная информационная технология. Большинство соображений вполне очевидно, и многие из них можно найти в других публикациях. (Вполне возможно, что вы об этом уже знаете.)

- **Образование пользователей.** Поощряйте пользователей отключать оборудование, когда оно не нужно, думать, прежде чем распечатывать документы на принтере, и переводить настольные системы в энергосберегающий режим вместо запуска экранной заставки (а еще лучше, выключать их).
- **Мониторы.** Заменяйте электроннолучевые трубки (ЭКТ) на жидкокристаллические дисплеи (LCD). Последние тратят намного меньше электроэнергии и содержат меньше токсичных элементов.
- **Простои рабочих станций.** Централизованно настройте рабочие станции так, чтобы во время простоя они переходили в спящий режим или отключались на заданный период (например, на 30 минут).
- **Считайте рабочие станции.** Ограничьте количество рабочих станций — не больше одной станции на человека. Пользователи, заявляющие, что им нужно несколько станций, должны использовать виртуального клиента.
- **Размер станции должен соответствовать поставленной задаче.** Не покупайте одинаковые для всех рабочие станции. Выберите три-четыре уровня спецификаций, чтобы пользователи могли выбрать конфигурацию, соответствующую их заданиям.
- **Персональные обогреватели.** Вообще-то, эта тема не относится к информационным технологиям, но это досадный фактор, который замечают только компьютерщики. Не разрешайте использовать персональные обогреватели в офисах и рабочих отсеках. Объясните пользователям, что эти обогреватели создают порочный круг, в котором система терморегуляции офиса и обогреватель борются друг с другом,

преследуя противоположные цели. Если температура воздуха на рабочем месте пользователя действительно является некомфортной, поставьте этот вопрос перед группой обслуживания HVAC. (Возможно, в обмен на особые услуги в области информационных технологий.)

- **Двусторонняя печать.** По умолчанию настройте принтеры на двустороннюю печать, по две страницы на листе. Этого вполне достаточно для рутинных распечаток, и пользователи всегда могут выбрать другие настройки, если им понадобится иначе распечатать документы.
- **Электронные документы.** Начните кампанию или проведите конкурс внутри вашей организации, чтобы найти способ исключить использование бумажных документов.
- **Температура в офисе.** Поскольку компьютерное оборудование офиса предназначено для работы при значительно более высоких температурах, чем привыкли люди, поднимите уровень охлаждения в офисе до 27°C и выше.
- **Утилизация оборудования.** Один-два раза в год проводите дни утилизации оборудования, в течение которых сотрудники могут избавиться от нежелательного, ненужного или слишком старого оборудования и сдать его компании, занимающейся утилизацией компьютеров. Если вас действительно беспокоит состояние окружающей среды, разрешите вашим сотрудникам принести для утилизации старые компьютеры из дома.
- **Продление сроков эксплуатации оборудования.** Как только рабочая станция стала слишком старой или медленной, чтобы использовать ее для интенсивной работы, передайте ее сотрудникам, которых устроят ее характеристики. Они будут рады получить новую технику, а у вас появится один-два дополнительных года эксплуатации.
- **Утилизация на рабочем месте.** Начните программу утилизации использованной бумаги на рабочем месте. Многие компании по переработке отходов, кроме бумаги, принимают пластик (пластиковые бутылки и т.д.).
- **Утилизация бумаги и картриджей.** Станьте потребителем повторно используемых товаров. Покупайте для ваших принтеров и копировальных машин бумагу, изготовленную только из макулатуры, а также восстановленные картриджи. Рекомендуем универсальную и недорогую бумагу Boise Aspen 100, изготовленную из макулатуры и обладающую превосходными экологическими характеристиками.
- **Работа в дистанционном режиме.** Стимулируйте сотрудников один-два дня в неделю работать дома в дистанционном режиме, установив и эксплуатируя средства дистанционного доступа, такие как сеть VPN, служба VOIP и веб-приложения. Кроме выгоды для ваших сотрудников, работа в дистанционном режиме уменьшает нагрузку на транспорт и вспомогательные службы офиса. Однако примите меры, чтобы сотрудники, работающие в дистанционном режиме, выключили оборудование, которое они в данный день не используют. В противном случае эта стратегия принесет обратный эффект, по крайней мере с точки зрения энергопотребления.

## 28.5. Компании, поддерживающие экологичные информационные технологии

Если вас интересует, кто еще разрабатывает экологичные информационные технологии, вы можете найти и помощь единомышленников в разных организациях. Некоторые из них перечислены в табл. 28.2.

**Таблица 28.2. "Зеленая мафия"**

| Организация | Веб-сайт | Описание |
|---|---|---|
| Energy Star | `energystar.gov` | Стандарты потребительской продукции |
| EPEAT | `epeat.net` | Производство экологичной электроники |
| French Green IT | `greenit.fr` | Французский блог, посвященный экологичным информационным технологиям |
| Green IT Observatory | `greenit.bf.rmit.edu.au` | Исследования в области экологичных информационных технологий |
| Green IT Promo Council | `greenit-pc.jp` | Экологичные информационные технологии для Японии и Азии |
| Green Standards Trust | `greenstandards.org` | Утилизация офисного оборудования |
| IT Industry Council | `itic.org` | Наилучшие стратегии для информационных технологий |
| Less Watts | `lesswatts.org` | Экономия электроэнергии с помощью системы Linux |
| The Green Grid | `thegreengrid.org` | Центры обработки данных |

Кроме экологических идей, многие из этих организаций имеют собственные наборы стандартных показателей, которые можно использовать для сравнения организаций по размерам и сферам деятельности.

# 28.6. Упражнения

28.1. С помощью электроизмерительного прибора Kill a Watt измерьте уровень потребления электроэнергии вашей рабочей станцией при разных уровнях нагрузки, включая спящий режим и режим пониженного энергопотребления. Сколько энергии можно сэкономить, если отключать вашу рабочую станцию каждый вечер?

28.2. Напишите сценарий, который будет посылать системному администратору электронную почту, когда нагрузка центрального процессора достигнет уровня, при котором следует подключать новый сервер.

28.3. Создайте список основных приложений, используемых в вашей организации в настоящий момент. Какие из них дублируют друг друга?

28.4. Зайдите на сайт `thegreengrid.org` и определите, стоит ли для охлаждения вашего офиса использовать внешний воздух.

28.5. ★ Такие организации, как TerraPass и Carbonfund.org, продают "компенсаторы' $CO_2$, с помощью которых организации могут возместить свои выбросы углекислого газа. Например, организации, осуществляющие компенсацию, могут субсидировать разработку источников энергии, не связанной с выбросом углекислого газа (например, солнечной энергии или энергии ветра), чтобы снизить эмиссию этого газа в будущем.

Эти программы оказались спорными. Некоторые наблюдатели сомневаются в реальности заявленных целей, а другие критикуют программу с философских позиций.

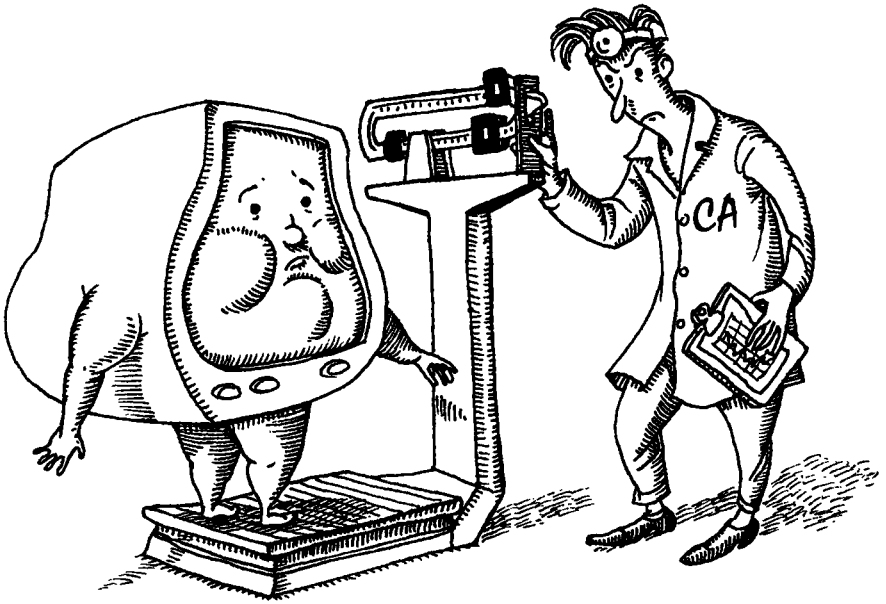
28.6. Выберите поставщика компенсаторов и оцените целесообразность предлагаемой им стратегии. Хорошо ли документированы его программы, чтобы правильно оценить их качество?[7]

Оценила ли этого провайдера независимая группа, и если да, к каким выводам она пришла?

[7] Разработчик из компании WorldPress Марк Жаке (Mark Jaquith) написал: "Представьте себе, что вас убили, а потом убийцу стали убеждать убить на одного человека меньше. Мы не должны вступать в переговоры с убийцами. Вас все равно убили. Убеждать кого-то сократить выбросы — совсем не значит, что ваши выбросы исчезнут". Мы не разделяем это мнение, но такая точка зрения на вопросы компенсации является довольно типичной.

# Глава 29

# Анализ производительности




Анализ производительности и ее настройку часто относят к магической стороне системного администрирования. Конечно же, магии здесь никакой нет, но можно говорить о симбиозе науки и искусства. "Научная" часть включает тщательное проведение количественных измерений и применение научного подхода. Составляющая "искусства" связана с необходимостью сохранять баланс ресурсов на практическом уровне, поскольку оптимизация для одного приложения или пользователя может отрицательно повлиять на другие приложения или на других пользователей. Как это часто бывает в жизни, невозможно сделать счастливыми всех сразу.

В блогосфере бытует мнение, что сегодня проблемы производительности кардинально отличаются от аналогичных проблем предыдущих десятилетий. Но это не совсем так. Да, системы стали гораздо сложнее, но определяющие факторы производительности и высокоуровневые абстракции, используемые для ее измерения и управления, не меняются. К сожалению, повышение производительности систем сильно коррелирует со способностью соответствующего сообщества создавать новые приложения, которые поглощают все доступные ресурсы.

В этой главе мы сосредоточим внимание на производительности серверных систем. В настольных системах проблемы, характерные для серверов, обычно не возникают, поэтому ответ на вопрос о том, как повысить производительность настольной системы, обычно прост: "Провести модернизацию". Пользователям такой ответ нравится, поскольку это означает, что у них появятся новые модные системы.

Одним из аспектов, отличающих UNIX и Linux от других основных операционных систем, является объем данных, которые они предоставляют о своих внутренних операциях. Детальная информация доступна буквально по каждому уровню системы, благода-

ря чему администраторы могут настраивать различные параметры именно так, как нужно. Если установить источник проблем с производительностью никак не удается, всегда можно просмотреть исходный код. По этим причинам UNIX и Linux часто считаются наиболее подходящими операционными системами для пользователей, которых особенно беспокоит тема производительности.

Даже при этих условиях настройка производительности не является простой задачей. Как пользователи, так и администраторы часто думают, что если бы они владели нужными "магическими" приемами, их системы работали бы в два раза быстрее. Одним из наиболее распространенных заблуждений является убеждение в том, что настроить производительность может помочь манипулирование переменными ядра, которые отвечают за управление системой страничного обмена и буферными пулами. В наши дни ядра основных дистрибутивов изначально настроены на достижение разумной (хотя и не оптимальной) производительности при различных уровнях загруженности.

Если попытаться оптимизировать систему по какому-то конкретному показателю производительности (например, по степени использования буферов), весьма вероятно, что поведение системы ухудшится в отношении других показателей и режимов работы. В этой главе будет рассказываться о том, как можно настроить производительность на уровне системы; возможность рассказать о настройке производительности на уровне приложений мы оставили другим. Как системному администратору, вам необходимо помнить о том, что разработчики — тоже люди. (Сколько раз вы говорили или думали, что проблема кроется в сети?) При таком уровне сложности современных приложений некоторые проблемы можно решить только посредством сотрудничества их разработчиков, системных администраторов, проектировщиков серверов, администраторов баз данных, администраторов памяти и архитекторов сети. В этой главе мы поможем вам определить, какие данные следует предъявить этим специалистам, чтобы они смогли решить проблемы производительности (если в действительности эти проблемы лежат в их области). Такой подход гораздо эффективнее, чем просто сказать: "Все выглядит прекрасно, это не моя проблема".

В любом случае, никогда не доверяйте полностью тому, что пишут в Интернете. Какие только аргументы не приводятся в дискуссиях о производительности систем! При этом сторонники всевозможных теорий не обладают, как правило, ни знаниями, ни дисциплиной, ни временем, необходимыми для проведения достоверных экспериментов. Широкая поддержка пользователей еще ничего не значит, ибо каждое глупое предложение очень часто сопровождается хором восторженных комментариев: "Я увеличил размер кеш-буфера в десять раз, как он предложил, и система заработала гораздо, гораздо быстрее!!!" Да уж, конечно!

Ниже перечислены некоторые правила, которыми следует пользоваться.

- Собирайте и анализируйте хронологические данные о работе системы. Если еще неделю назад производительность системы была нормальной, а сейчас все изменилось, то сравнение характеристик системы в двух состояниях позволит выявить источник проблемы. Всегда держите под рукой список оптимальных параметров производительности. А также первым делом просматривайте журнальные файлы, чтобы выяснить, не связана ли возникшая проблема с появлением какой-нибудь неполадки в оборудовании.
- В главе 21 рассказывалось о некоторых средствах трендового анализа, которые также подходят для мониторинга производительности. Утилита `sar`, которая будет описываться чуть позже в этой главе, тоже может использоваться в качестве средства трендового анализа.

- Всегда настраивайте систему так, чтобы потом иметь возможность сравнить результаты с предыдущими показателями производительности системы.
- Всегда проверяйте наличие плана отката на случай, если внесенные "чудодейственные" изменения окажутся вовсе не чудодейственными и лишь только ухудшат дело.
- Не перегружайте умышленно свои системы или свою сеть. Ядро создает для каждого процесса иллюзию бесконечности ресурсов. Но как только в дело вступают все 100% ресурсов системы, операционной системе приходится работать очень напряженно. На поддержку иллюзии расходуется ощутимая доля самих ресурсов. В результате выполнение процессов замедляется.
- Как в физике элементарных частиц, чем больше данных вы собираете с помощью утилит мониторинга системы, тем большее влияние вы оказываете на исследуемую систему. Для наблюдения за системой лучше всего полагаться на простые инструменты, которые работают в фоновом режиме (например, `sar` или `vmstat`). Если они обнаружат что-то существенное, то для более глубокого анализа можете воспользоваться и другими инструментами.

## 29.1. Способы повышения производительности

Ниже перечислены конкретные действия, которые можно выполнить, чтобы улучшить производительность.

- Убедиться в том, что память доступна в достаточном объеме. В следующем разделе вы увидите, что объем памяти очень сильно влияет на производительность. Устройства памяти сегодня стоят довольно недорого, так что обычно оснастить по максимуму любой компьютер, от которого требуется высокая производительность, — не проблема.
- Если UNIX- или Linux-система используется в качестве веб-сервера или сетевого сервера приложений, можно попробовать распределить трафик между несколькими компьютерами с помощью какого-нибудь коммерческого приложения для балансирования нагрузки, такого как Content Services Switch производства компании Cisco (`cisco.com`), ServerIron производства компании Brocade (`brocade.com`) или Alteon Web Switch производства компании Nortel (`nortelnetworks.com`).
- Эти устройства делают так, что несколько физических серверов выглядят для внешнего мира как один логический сервер. Они распределяют нагрузку согласно одному из нескольких выбираемых пользователем алгоритмов типа "минимальное время отклика" или "циклический алгоритм обслуживания".
- Тщательно проверить конфигурацию системы и отдельных приложений. Многие приложения могут резко увеличить производительность, если их правильно настроить (распределить данные между дисками, не осуществлять динамический поиск имен в DNS, запустить дополнительные экземпляры сервера и т.д.).
- Устранить проблемы, связанные с использованием ресурсов. Проблемы могут создаваться как "реальными заданиями" (одновременный запуск слишком большого количества серверов, неэффективные методы программирования, выполнение пакетов заданий с завышенным приоритетом, запуск ресурсоемких программ в неподходящее время), так и самой системой (ненужные демоны).
- Ликвидировать по возможности зависимость ресурсов памяти от механических операций. В настоящее время широко доступны полупроводниковые диски (Solid

state disk drives — SSDs), которые могут обеспечить быстрый рост производительности, поскольку они не требуют физического движения диска для считывания битов информации. SSD-диски легко устанавливаются вместо "старых добрых" дисковых накопителей[1].

- Организовать жесткие диски и файловые системы так, чтобы нагрузка на них была равномерной, и тем самым максимально повысить пропускную способность каналов ввода-вывода. При наличии специфических приложений, таких как базы данных, можно попробовать использовать какую-нибудь модную многодисковую технологию вроде RAID для оптимизации процессов обмена данными. За рекомендациями следует обращаться к производителю базы данных. Для систем Linux нужно удостовериться в том, что для диска выбран самый подходящий из доступных в Linux планировщик ввода-вывода.
- Обратить внимание на то, что разные приложения и СУБД ведут себя по-разному при размещении на нескольких дисках. Технология RAID поставляется в нескольких вариантах, поэтому следует потратить время и выяснить, какой из них лучше всего подходит для данного конкретного случая (если вообще подходит).
- Провести мониторинг сети, чтобы убедиться в том, что она не перегружена трафиком и что количество ошибок в ней минимально. Очень много такой полезной информации о сети можно собрать с помощью команды **netstat**, которая описывалась в разделе 21.5. Также можно еще раз перечитать главу 21.
- Определить ситуации, в которых система совершенно не удовлетворяет предъявляемым к ней требованиям. Нельзя настраивать производительность без учета таких ситуаций.

Эти меры перечислены в порядке убывания эффективности. Добавление памяти и распределение трафика между несколькими серверами может значительно повысить производительность. Степень эффективности остальных мер варьируется от значительной до нулевой.

Анализ и оптимизация структур данных и алгоритмов программ почти всегда ведет к существенному повышению производительности. Однако эти вопросы обычно находятся вне компетенции системного администратора.

## 29.2. Факторы, влияющие на производительность

Производительность системы во многом определяется базовыми характеристиками системных ресурсов и эффективностью их распределения и совместного использования. Определение термина "ресурс" весьма расплывчато. Оно может подразумевать даже такие компоненты, как кеш содержимого регистров центрального процессора и элементы таблицы адресов контроллера памяти. Однако в первом приближении серьезное влияние на производительность оказывают только четыре фактора:

- время использования центрального процессора;
- память;

[1] У современных SSD-дисков есть два основных недостатка. Во-первых, они на порядок дороже (в расчете на гигабайт), чем традиционные жесткие диски. Во-вторых, их можно перезаписывать только ограниченное число раз. Их перезаписывающая способность достаточно высока и вполне приемлема для настольных компьютеров (десятки тысяч записей на блок), но может оказаться потенциальным камнем преткновения для сервера с большой рабочей нагрузкой. Подробнее о SSD-дисках см. в разделе 8.2.

- пропускная способность дисковой подсистемы ввода-вывода;
- пропускная способность сетевой подсистемы ввода-вывода.

Если после выполнения активных процессов остались свободные ресурсы, можно считать, что производительность системы является удовлетворительной.

Когда ресурсов не хватает, процессы становятся в очередь. Процесс, не имеющий немедленного доступа к необходимым ресурсам, должен ждать, ничего при этом не делая. Время, затрачиваемое на ожидание ресурсов, — один из основных показателей ухудшения производительности.

Самый простой, с точки зрения учета, ресурс — использование центрального процессора (ЦП). В распоряжении системы всегда имеется примерно одна и та же часть его мощности. Теоретически эта величина составляет все 100% циклов ЦП, но "накладные расходы" и другие причины приводят к снижению этого показателя примерно до 95%. Для процесса, занимающего более 90% времени ЦП, ограничивающим фактором является быстродействие центрального процессора. Такой процесс потребляет большую часть вычислительных мощностей системы.

Многие считают, что быстродействие ЦП — основной фактор, влияющий на общую производительность системы. При наличии неограниченных объемов всех остальных ресурсов или для определенных типов приложений (например, программ численного моделирования) быстродействие ЦП действительно играет роль. Но в повседневной жизни этот показатель значит относительно мало.

Настоящим узким местом является канал обмена данными с жестким диском. Жесткие диски представляют собой механические системы, поэтому на поиск нужного блока, выборку его содержимого и активизацию ожидающего его процесса уходят десятки миллисекунд. Задержки такого порядка отодвигают на второй план все остальные факторы ухудшения производительности. Каждое обращение к диску вызывает приостановку активного процесса "стоимостью" несколько миллионов инструкций ЦП. Эту проблему можно решить, например, с помощью полупроводниковых дисковых накопителей, которые работают значительно быстрее, чем диски с движущимися деталями.

Благодаря использованию виртуальной памяти, скорость дискового обмена может быть непосредственно связана с объемом имеющейся памяти, если потребность в физической памяти превышает ее реальный объем. Ситуации, в которых физической памяти становится недостаточно, часто приводят к необходимости записывать содержимое ее страниц на диск, чтобы эти страницы можно было восстановить и использовать для другой цели. Это означает, что обращение к памяти по своей "стоимости" приближается к работе с диском. Избегайте таких ловушек, если характеристики производительности имеют для вас большое значение; побеспокойтесь о том, чтобы каждая система имела достаточно физической памяти.

Пропускная способность сети подвержена примерно тем же ограничениям, что и скорость дискового обмена. Ее ухудшение связано со всевозможными задержками. Но возникающие проблемы охватывают целые сообщества пользователей, а не отдельные компьютеры. Кроме того, сети восприимчивы к проблемам аппаратного обеспечения и перегрузкам серверов.

## 29.3. Как анализировать проблемы производительности

В сложной системе нелегко выделить именно проблемы производительности. Системные администраторы часто получают частные (несистематические) отчеты о пробле-

мах, которые содержат эмоциональные описания конкретных случаев или сложных ситуаций (например, "веб-сервер застопорился из-за всех этих проклятых AJAX-вызовов..."). Вы, конечно, должны обратить внимание на эту информацию, но не воспринимайте ее как достоверную и надежную; проводите собственное исследование.

Строгая и прозрачная научная методика поможет вам сделать правильные выводы, которым ваши сотрудники и вы сами можете доверять. Научный подход позволит другим оценить ваши результаты, повысит доверие к вашей работе и увеличит вероятность того, что предлагаемые вами изменения смогут реально решить проблему.

Применение научного подхода не означает, что вы должны собирать все релевантные данные сами. Весьма полезной бывает и "внешняя" информация. Не стоит тратить часы на изучение вопросов, ответы на которые можно легко почерпнуть из разделов FAQ (Frequently Asked Questions — часто задаваемые вопросы).

Мы предлагаем вам выполнить следующие пять действий.

**Шаг 1. Сформулируйте вопрос.**

Поставьте конкретный вопрос в определенной функциональной области или сформулируйте предварительное заключение или рекомендацию, которую вы обдумали. Будьте точны, говоря о технологиях, компонентах и альтернативах, которые вы предлагаете рассмотреть, и результатах, которые могут быть достигнуты.

**Шаг 2. Соберите и классифицируйте факты.**

Проведите систематический поиск в документации, базах знаний, известных вам изданиях, блогах, технических описаниях, форумах и прочих информационных ресурсах с целью выявления внешних фактов, имеющих отношение к вашему вопросу. В собственных системах зафиксируйте телеметрические данные и (по возможности или необходимости) оснастите инструментальными средствами конкретные интересующие вас области в системе или отдельных приложениях.

**Шаг 3. Критически оцените данные.**

Просмотрите каждый источник данных на предмет релевантности и критически оцените его с точки зрения достоверности. Выделите ключевые данные и обратите внимание на качество источников информации.

**Шаг 4. Подытожьте данные вербально и графически.**

Представьте сведения, взятые из нескольких источников, в словесной (вербальной) и по возможности графической форме. Данные, кажущиеся сомнительными при выражении в числовой форме, могут оказаться убедительными в виде диаграммы.

**Шаг 5. Сформулируйте заключение.**

Представьте свои выводы в лаконичной форме (как ответ на поставленный вами же вопрос). Поставьте оценку, чтобы обозначить уровень силы или слабости доказательств, которые поддерживают ваши заключения.

# 29.4. Проверка производительности системы

Хватит обобщать — пора рассмотреть некоторые конкретные инструменты и "зоны интереса". Прежде чем переходить к измерениям, необходимо знать, что именно вы хотите получить.

## Инвентаризуйте свое оборудование

Начните с оценки своего оборудования, особенно ЦП и ресурсов памяти. Инвентаризация поможет вам интерпретировать данные, представленные с помощью других

инструментов, а также построить реалистичные ожидания касательно верхних границ производительности.

В системах Linux именно в файловой системе **/proc** стоит искать ответ на вопрос, какое, с точки зрения вашей операционной системы, вы используете оборудование (более детальную информацию об аппаратуре можно найти в каталоге **/sys**; см. раздел 13.8). Некоторые ключевые файлы перечислены в табл. 29.1. Общую информацию о файловой системе **/proc** можно найти в разделе 13.3.

**Таблица 29.1. Источники информации об оборудовании в системах Linux**

| Файл | Содержимое |
|---|---|
| **/proc/cpuinfo** | Тип и описание ЦП |
| **/proc/meminfo** | Размер памяти и коэффициент загрузки |
| **/proc/diskstats** | Дисковые устройства и статистика их использования |

Четыре строки в файле **/proc/cpuinfo** помогут вам идентифицировать ЦП в системе: `vendor_id`, `cpu family`, `model`, и `model name`. Некоторые из этих значений непонятны, поэтому лучше всего узнать о них в интерактивной справочной системе.

Точная информация, содержащаяся в файле **/proc/cpuinfo**, зависит от системы и процессора. Рассмотрим типичный пример содержимого этого файла.

```
suse$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 15
model name      : Intel(R) Xeon(R) CPUE5310@ 1.60GHz
stepping        : 11
cpu MHz         : 1600.003
cache size      : 4096 KB
physical id     : 0
cpu cores       : 2
siblings        : 2
..
```

Этот файл содержит по одной записи для каждого процессорного ядра, видимого операционной системой. Конкретные данные незначительно варьируются в зависимости от версии ядра. Значение поля `processor` уникально идентифицирует ядро. Значения поля `physical id` являются уникальными для каждого физического сокета на печатной плате, а значения `id values` уникальны для ядра в физическом сокете. Ядра, которые поддерживают гиперпотоковый режим работы (дублирование контекстов ЦП без дублирования других характеристик обработки), идентифицируются значением `ht` в поле `flags`. Если в системе действительно реализован гиперпотоковый режим работы, поле `siblings` для каждого ядра показывает, сколько контекстов доступно на данном ядре.

Существует еще одна команда, которая позволяет получить информацию об аппаратных характеристиках вашего компьютера. Речь идет о команде **dmidecode**. Она выводит содержимое системной таблицы DMI (Desktop Management Interface — интерфейс управления настольными системами), также известной под именем SMBIOS. Самой полезной опцией этой команды является **-t** *тип* (допустимые типы представлены в табл. 29.2).

**Таблица 29.2. Значения типов для команды `dmidecode -t`**

| Значение | Описание |
|---|---|
| 1 | Система |
| 2 | Материнская плата |
| 3 | Корпус |
| 4 | Процессор |
| 7 | Кеш-память |
| 8 | Разъемы портов |
| 9 | Системные разъемы |
| 11 | OEM-строки |
| 12 | Опции системной конфигурации |
| 13 | Язык BIOS |
| 16 | Массив физической памяти |
| 17 | Устройство памяти |
| 19 | Отображаемый адрес массива памяти |
| 32 | Загрузка системы |
| 38 | IPMI-устройство |

Следующий пример демонстрирует получение типичной информации.

```
suse$ sudo dmidecode -t 4
# dmidecode 2.7
SMBIOS 2.2 present.

Handle 0x0004, DMI type 4, 32 bytes.
Processor Information
   Socket Designation: PGA 370
   Type: Central Processor
   Family: Celeron
   Manufacturer: GenuineIntel
   ID: 65 06 00 00 FF F9 83 01
   Signature: Type 0, Family 6, Model 6, Stepping 5
```

Биты информации о сетевой конфигурации разбросаны "по всей системе". Лучшим источником IP- и MAC-информации для каждого сконфигурированного интерфейса служит команда `ifconfig -a`.

В системах Solaris лучшим источником информации о ЦП и ресурсах памяти являются команды `psrinfo -v` и `prtconf` соответственно. Вот пример результата выполнения этих команд.

```
solaris$ psrinfo -v
Status of virtual processor 0 as of: 01/31/2010 21:22:00
   on-line since 07/13/2009 15:55:48.
   The sparcv9 processor operates at 1200 MHz,
      and has a sparcv9 floating point processor.
Status of virtual processor 1 as of: 01/31/2010 21:22:00
   on-line since 07/13/2009 15:55:49.
   The sparcv9 processor operates at 1200 MHz,
      and has a sparcv9 floating point processor.
```

```
solaris$ prtconf
System Configuration:  Sun Microsystems  sun4v
Memory size: 32640 Megabytes
System Peripherals (Software Nodes):

SUNW,Sun-Fire-T200
...
```

В системах HP-UX информацию о конфигурации аппаратных средств компьютера можно получить с помощью одной-единственной команды `machinfo`. Вот типичный пример ее выполнения.

```
hp-ux$  sudo machinfo
CPU info:
   1 Intel(R) Itanium 2 processor (1.5 GHz, 6 MB)
      400 MT/s bus, CPU version B1

Memory: 4084 MB (3.99 GB)

Firmware info:
   Firmware revision:      02.21
   FP SWA driver revision: 1.18
   BMC firmware revision:  1.50

Platform info:
   Model:       "ia64 hp server rx2600"
OS info:
   Nodename:    hpux11
   Release:    HP-UX B.11.31
Machine:       ia64
```

В системах AIX для получения информации о ЦП и памяти придется выполнить несколько действий. Сначала с помощью команды **lscfg** найдите имена инсталлированных процессоров.

```
aix$  lscfg | grep Processor
+ proc0                    Processor
+ proc2                    Processor
```

Затем для получения описания каждого процессора можно использовать команду `lsattr`.

```
aix$  lsattr -E -l proc0
frequency      1898100000      Processor    Speed          False
smt_enabled    true            Processor    SMT enabled    False
smt_threads    2               Processor    SMT threads    False
state          enable          Processor    state          False
type           PowerPC_POWER5  Processor    type           False
```

С помощью команды `lsattr` можно также узнать объем физической памяти в системе.

```
aix$  lsattr -E -l sys0 -a realmem
realmem       4014080       Amount of usable physical memory in Kbytes
```

## Сбор данных о производительности

Программы анализа производительности в большинстве своем сообщают о том, что происходит в системе в данный момент времени. Но следует учесть, что структура и характер нагрузки меняются в течение дня. Поэтому до принятия мер обязательно проведите всесторонний анализ данных, касающихся производительности. Истинная производительность выясняется только после длительного (месяц или даже больше) наблюдения за системой. Особенно важно собирать данные в периоды пиковой загруженности системы. Ограничения на использование ресурсов и ошибки в конфигурации системы часто выявляются только в таких условиях.

## Анализ использования центрального процессора

Обычно собирают три вида данных о центральном процессоре: общий показатель использования, показатели средней загруженности и потребление ресурсов отдельными процессами. Первая характеристика определяет, является ли быстродействие самого процессора узким местом. Показатели средней загруженности дают возможность оценить общую производительность системы. Данные, касающиеся отдельных процессов, позволяют выявить наиболее ресурсоемкие процессы.

Сводную информацию выдает команда `vmstat`. Она принимает два аргумента: время (в секундах), в течение которого необходимо наблюдать за системой для получения каждой строки выходной информации, и необходимое количество отчетов. Если не указать число отчетов, команда будет выполняться до тех пор, пока пользователь не нажмет комбинацию клавиш <Ctrl+C>.

В первой строке отображаемых данных сообщаются средние значения, измеренные с момента запуска системы. В последующих строках выдаются результаты по каждому очередному замеру, стандартная продолжительность которого составляет пять секунд.

```
$ vmstat 5 5
procs  -----------memory---------  -swap- --io--  -system--  ----cpu----
r b  swpd     free  buff     cache  si so   bi bo   in   cs  us sy id wa
1 0   820  2606356  428776  487092   0  0 4741 65 1063 4857  25  1 73  0
1 0   820  2570324  428812  510196   0  0 4613 11 1054 4732  25  1 74  0
1 0   820  2539028  428852  535636   0  0 5099 13 1057 5219  90  1  9  0
1 0   820  2472340  428920  581588   0  0 4536 10 1056 4686  87  3 10  0
3 0   820  2440276  428960  605728   0  0 4818 21 1060 4943  20  3 77  0
```

Несмотря на то что содержимое этих колонок может не совпадать в разных системах, статистические данные показателей использования ЦП практически не различаются среди платформ. Пользовательское время, системное время (время ядра), время простоя и время ожидания для каналов ввода-вывода (I/O) показаны в колонках `us`, `sy`, `id` и `wa` соответственно. Большие числа в колонке `us` обычно означают вычисления, а в колонке `sy` они свидетельствуют о том, что процессы осуществляют очень много системных вызовов или выполняют операции ввода-вывода.

Выработанное нами за многие годы эмпирическое правило для серверов общего назначения, справедливое для большинства систем, гласит следующее: система должна тратить примерно 50% рабочего времени на обслуживание пользовательских запросов и столько же на системные запросы. Общее время простоя не должно быть нулевым. Если сервер выделяется под одно единственное, но интенсивно использующее ЦП приложение, большая часть времени должна тратиться на обработку пользовательских запросов.

В колонке cs показано число переключений контекста за данный период, т.е. сколько раз ядро переключало процессы. В колонке in отображается число прерываний, генерируемых аппаратными устройствами или компонентами ядра. Слишком большие значения в этих колонках свидетельствуют о неправильно работающем аппаратном устройстве. Остальные колонки важны для анализа использования памяти и жесткого диска, о чем будет рассказываться позже в этой главе.

Длительные измерения показателей, характеризующих работу центрального процессора, позволяют определить, достаточно ли его ресурсов для нормальной работы системы. Если процессор регулярно часть времени простаивает, значит, есть запас по тактовой частоте. В этом случае увеличение быстродействия процессора ненамного улучшит общую производительность системы, хотя может и ускорить выполнение отдельных операций.

Как видно из показанного примера, центральный процессор постоянно переключается из режима интенсивного использования в режим полного бездействия и обратно. Поэтому необходимо наблюдать за показателями, соответствующими обоим режимам, и выводить среднее за определенный период времени. Чем меньше интервал наблюдения, тем ниже достоверность оценок.

В мультипроцессорных системах команды вроде **vmstat** сообщают средние значения по всем процессорам. Но существует и команда **mpstat**, которая выдает отчет по каждому процессору. В системах Linux, Solaris и AIX с помощью флага -P можно выбрать один конкретный процессор. Этой командой удобно пользоваться при отладке программ, поддерживающих симметричную многопроцессорную обработку. Интересно также узнать, насколько система эффективно (или неэффективно) работает с несколькими процессорами.

Рассмотрим пример отображения статуса каждого из четырех процессоров.

```
linux$ mpstat -P ALL
08:13:38 PM CPU %user %nice %sys %iowait %irq %soft %idle intr/s
08:13:38 PM   0  1.02  0.00 0.49    1.29 0.04  0.38 96.79 473.93
08:13:38 PM   1  0.28  0.00 0.22    0.71 0.00  0.01 98.76 232.86
08:13:38 PM   2  0.42  0.00 0.36    1.32 0.00  0.05 97.84 293.85
08:13:38 PM   3  0.38  0.00 0.30    0.94 0.01  0.05 98.32 295.02
```

Центральный процессор однопользовательской рабочей станции обычно простаивает большую часть времени. Когда запрашивается прокрутка содержимого окна или обновляется веб-страница, ЦП справляется с этой операцией за считанные секунды. В такой ситуации долгосрочный средний показатель использования ЦП практически лишен смысла.

Еще одним статистическим показателем, полезным для оценки интенсивности использования системы, является показатель "средняя загруженность", который представляет собой среднее количество выполняемых процессов. Он дает достаточное представление о том, сколько процессов требуют свою долю процессорного времени. Узнать его значение можно с помощью команды uptime.

```
$ uptime
11:10am  up 34 days, 18:42, 5 users, load average: 0.95, 0.38, 0.31
```

Выдаются три значения, которые соответствуют средней загруженности за пять, десять и пятнадцать минут. Как правило, чем выше средняя загруженность, тем важнее общая производительность системы. Если выполняется всего лишь один процесс, то он обычно ограничен одним ресурсом (центральным процессором или пропускной способ-

ностью дискового канала ввода-вывода). Пиковый спрос на этот ресурс и становится определяющим фактором производительности.

Когда в системе одновременно работает несколько процессов, нагрузка распределяется более равномерно. Если все работающие процессы потребляют ресурсы ЦП, диска и памяти, то зависимость производительности системы от ограниченных возможностей какого-то одного ресурса снижается. В такой ситуации гораздо важнее следить за средними показателями загруженности, в частности за общим временем использования центрального процессора.

🕮 О приоритетах рассказывалось в конце раздела 5.1.

Обычно в однопроцессорных системах значение 3 показателя средней загруженности свидетельствует о сильной загруженности системы, а значение 8 и выше — считается критическим. Такие значения являются сигналом о том, что пора начать поиск путей искусственного перераспределения нагрузки, например назначить процессам приоритеты с помощью команды `nice`.

Показатель средней загруженности отлично подходит для повседневного контроля системы. Если известен показатель работающей системы и он находится в диапазоне, характерном для перегрузки, значит, следует искать внешний источник проблемы (это может быть, к примеру, сеть). В противном случае нужно проверить процессы самой системы.

Еще один способ контроля над использованием ЦП — посмотреть, какую часть его времени отнимает каждый процесс. Для этого служит команда **`ps`** с аргументами (`-aux` — в системах Linux и AIX; `-elf` — в системах HP-UX и Solaris). Зачастую в интенсивно эксплуатируемой системе минимум 70% времени ЦП отнимается одним-двумя процессами. Запуск таких процессов в другое время суток или снижение их приоритетов позволит высвободить ЦП для выполнения других процессов.

🕮 Подробнее об утилите **`top`** рассказывалось в разделе 5.8.

Прекрасной альтернативой команде **`ps`** является утилита **`top`**. Она выдает примерно ту же информацию, что и `ps`, но в "живом", постоянно обновляемом формате, позволяющем наблюдать за изменением состояния системы во времени[2]. В системе AIX можно воспользоваться даже более эффективной командой **`topas`**.

В виртуализированных системах **`ps`**, **`top`** и другие команды, которые отображают данные о показателях использования ЦП, могут вводить в заблуждение. Виртуальная машина, которая не использует все виртуальные циклы своего центрального процессора, позволяет другим виртуальным машинам использовать (заимствовать) эти циклы. Любое измерение, которое имеет отношение к операционной системе (например, количество "тиков" системных часов в секунду), должно быть вами тщательно проанализировано, чтобы вы до конца понимали результаты своих измерений. О различных технологиях виртуализации можно подробнее см. в главе 24.

## Управление памятью в системе

В UNIX- и Linux-системах память организована в виде модулей, называемых *страницами*, размер которых составляет 4 Кбайт или больше. Когда процессы запрашивают у операционной системы память, ядро выделяет им виртуальные страницы. Каждой такой странице соответствует настоящее физическое хранилище, такое как ОЗУ или "резерв-

[2] Утилита *top* потребляет довольно много ресурсов, поэтому пользуйтесь ею в разумных пределах.

ное ЗУ" на жестком диске. (Резервное ЗУ обычно представляет собой просто пространство в области подкачки, но для страниц, которые содержат текст исполняемой программы, резервное ЗУ — это самый настоящий исполняемый файл. Аналогично резервное ЗУ для некоторых файлов данных может представлять собой сами файлы.) Информация о связях между виртуальными и реальными страницами хранится в таблице страниц.

Ядро способно эффективно обслуживать запросы процессов, выделяя для них столько памяти, сколько им нужно. Это реализуется дополнением реального ОЗУ областью подкачки. Поскольку процессы ожидают, что их виртуальные страницы будут отображаться в реальной памяти, ядру постоянно приходится перемещать страницы между ОЗУ и областью подкачки. Такие операции называются *страничным обменом* (paging)[3].

Ядро старается управлять памятью так, чтобы страницы, к которым недавно обращались, хранились в физической памяти, а менее активные страницы выгружались на диск. Этот алгоритм известен под названием LRU (least recently used — замещение наименее используемых элементов), поскольку те страницы, к которым долго никто не обращался, перемещаются на диск.

Впрочем, отслеживать все обращения к страницам было бы слишком накладно для ядра, поэтому оно использует страничный кеш-буфер, анализ содержимого которого и позволяет принять решение о том, какие именно страницы оставить в памяти. Точный алгоритм этого механизма зависит от конкретной системы, но везде действует примерно одинаковый принцип. Такой вариант гораздо проще в реализации, чем LRU-система, а результаты дает почти такие же.

В случае нехватки памяти ядро пытается определить, какие страницы из "неактивного" списка давно не использовались. Если при последнем обращении такие страницы модифицировались процессом, ядро считает их "грязными"; они обязательно должны быть выгружены на диск, прежде чем память будет повторно использована. Все остальные страницы считаются "чистыми" и могут быть переданы другому процессу.

Когда выполняется обращение к странице из "неактивного" списка, ядро возвращает ссылку на нее в таблицу страниц, обнуляет ее "возраст" и переводит страницу из "неактивного" списка в "активный". Страницы, выгруженные на диск, должны быть прочитаны с диска, прежде чем их можно будет активизировать. Когда процесс обращается к неактивной странице, находящейся в памяти, происходит *сбой* программной страницы (это так называемая "мягкая ошибка"). В случае обращения к нерезидентной (выгруженной) странице имеет место *отказ* страницы диска (а это уже "жесткая ошибка"). Другими словами, при возникновении ошибки второго типа (в отличие от первого) требуется прочитать страницу с диска.

Ядро старается прогнозировать потребности системы в памяти, поэтому операции выделения и выгрузки страниц не обязательно взаимосвязаны. Цель системы — обеспечить наличие достаточного объема свободной памяти, чтобы процессам не приходилось ждать выгрузки страниц всякий раз, когда им нужна свободная память. Если в периоды сильной загруженности системы страничный обмен резко возрастает, имеет смысл купить дополнительную память.

Система Linux по-прежнему быстро эволюционирует, и ее система виртуальной памяти продолжает развиваться, причем несколько неровно и даже неуклюже. Можно настроить параметр "подкачки" (**`/proc/sys/vm/swappiness`**) и тем самым дать ядру подсказку о том, насколько быстро оно должно делать

[3] Когда-то имела место иная операция, именуемая *подкачкой* (или *свопингом*), посредством которой все страницы некоторого процесса одновременно переписывались на диск. Теперь же во всех случаях обязательно используется страничный обмен (paging).

страницы пригодными для возврата из процесса в случае нехватки памяти. По умолчанию для этого параметра устанавливается значение 60. Если для него установить значение 0, ядро будет забирать страницы, которые были назначены процессу, только тогда, когда испробует все остальные возможности. Если для него установить значение 60 или выше (максимальным значением является 100), ядро, скорее всего, будет забирать страницы из процесса. Если возникает желание изменить этот параметр, значит, возможно, пришла пора увеличить ОЗУ.

Когда ядру не хватает как физической памяти, так и области подкачки, это означает, что виртуальная память исчерпана. В данной ситуации включается режим "принудительного уничтожения". Чтобы освободить память, системе приходится уничтожать целые процессы. И хотя выбираются наименее важные для системы процессы, все равно это очень неприятная процедура, ведь значительная часть ресурсов системы тратится не на полезную работу, а на управление памятью.

## Анализ использования памяти

Интенсивность операций с памятью количественно представляется двумя показателями: общим объемом задействованной виртуальной памяти и страничного обмена. Первый показатель характеризует общую потребность в памяти, а второй указывает на то, какая доля этой памяти активно используется. Задача администратора заключается в снижении интенсивности использования или увеличении объема памяти до тех пор, пока не будет достигнут приемлемый уровень страничного обмена. Страничный обмен — процесс неизбежный, поэтому не пытайтесь полностью избавиться от него.

У вас есть возможность определить размер текущей области подкачки. Для этого в Linux выполните команду **swapon -s**, в Solaris и AIX — команду **swap -l**, а в HP-UX — команду **swapinfo**.

```
linux$ swapon -s
Filename    Type        Size     Used  Priority
/dev/hdb1   partition   4096532  0     -1
/dev/hda2   partition   4096564  0     -2
solaris$ swap -l
swapfile             dev   swapl  blocks   free
/dev/dsk/c0t0d0s1   32,1      16  164400 162960
hp-ux$ swapinfo
             Kb     Kb       Kb    PCT START/       Kb
TYPE    AVAIL  USED     FREE  USED  LIMIT  RESERVE  PRI            NAME
dev   8388608     0  8388608    0%      0        -    1  /dev/vg00/lvol
```

При выполнении команд **swapinfo** и **swapon** значения выводятся в килобайтах, а команда **swap -l** использует 512-байтовые дисковые блоки. Значения размеров, выводимые этими командами, не включают содержимое памяти ядра, поэтому вам придется вычислить общий объем виртуальной памяти самостоятельно.

```
VM = ОЗУ + используемый_объем_области_подкачки
```

В системах UNIX вывод статистических показателей страничного обмена, полученных с помощью команды `vmstat`, подобен результату выполнения команды в системе Solaris.

```
solaris$ vmstat 5 5
procs       memory            page            disk          faults
```

```
r b w    swap   free  re  mf pi po fr de sr   s0 s6 s4 --  in  sy  cs
0 0 0  338216  10384   0   3  1  0  0  0  0    0  0  0  0 132 101  58
0 0 0  341784  11064   0  26  1  1  1  0  0    0  0  1  0 150 215 100
0 0 0  351752  12968   1  69  0  9  9  0  0    0  0  2  0 173 358 156
0 0 0  360240  14520   0  30  6  0  0  0  0    0  0  1  0 138 176  71
1 0 0  366648  15712   0  73  0  8  4  0  0    0  0 36  0 390 474 237
```

Из этого примера удалена информация об использовании центрального процессора. Под заголовком `procs` показано количество процессов, готовых к немедленному выполнению, заблокированных в ожидании ввода-вывода, а также готовых к выполнению, но перекачанных на диск. Если значение в колонке `w` когда-нибудь станет отличным от нуля, это будет означать, что объем системной памяти не соответствует текущей нагрузке.

Колонки, объединенные под общим заголовком `page`, содержат данные о выполнении страничного обмена. Во всех этих колонках представлены средние значения: количество в секунду (табл. 29.3).

В колонке `swap` выдается объем доступной виртуальной памяти в килобайтах. В колонке `free` отображается объем (в килобайтах) списка неиспользуемых страниц системы. Если приведенные в ней числа не достигают 3% от общего объема системной памяти, это говорит о наличии проблем.

**Таблица 29.3. Описание статистических показателей, выводимых командой `vmstat`**

| Колонка | Описание показателя |
|---|---|
| `re` | Количество восстановленных страниц, т.е. возвращенных из списка неиспользуемых |
| `mf` | Количество незначительных ошибок (связанных с небольшим числом страниц) |
| `pi` | Объем подкачанных данных в килобайтах |
| `po` | Объем выгруженных данных в килобайтах |
| `fr` | Объем списка неиспользуемых страниц в килобайтах |
| `de` | Объем "ожидаемого краткосрочного дефицита памяти" в килобайтах |
| `sr` | Количество страниц, обработанных по алгоритму часов |

Содержимое колонки `de` — самый надежный показатель возникновения серьезных проблем с памятью. Если находящееся в ней число зачастую "зашкаливает" за 100, это значит, что компьютеру нужно больше памяти. К сожалению, во многих версиях команды `vmstat` это значение не выводится.

В системах Linux статистические показатели, получаемые с помощью команды `vmstat`, могут иметь следующий вид.

```
linux$ vmstat 5 5
procs -------memory--------- -swap- ---io--  -system-  -----cpu------
 r b  swpd  free  buff  cache  si so   bi bo   in  cs  us sy id wa st
 5 0     0 66488 40328 597972   0  0  252 45 1042 278   3  4 93  1  0
 0 0     0 66364 40336 597972   0  0    0 37 1009 264   0  1 98  0  0
 0 0     0 66364 40344 597972   0  0    0  5 1011 252   1  1 98  0  0
 0 0     0 66364 40352 597972   0  0    0  3 1020 311   1  1 98  0  0
 0 0     0 66364 40360 597972   0  0    0 21 1067 507   1  3 96  0  0
```

Как и в результате выполнения команды **`vmstat`** при использовании системы UNIX, количество процессов, готовых к немедленному выполнению и заблокированных в ожидании ввода-вывода, выводится под заголовком **`procs`**. Статистические показатели

страничного обмена ограничены двумя колонками: si и so, которые представляют количество подкачанных и выгруженных страниц соответственно.

Кажущиеся несоответствия между колонками, большей частью, иллюзорны. В одних колонках указывается число страниц, в других — объем в килобайтах. Все значения являются округленными средними величинами. Более того, одни из них — средние скалярных величин, а другие — средние изменений.

С помощью полей si и so можно оценить интенсивность страничного обмена в системе. Операция загрузки (поле si) не обязательно означает, что из области подкачки восстанавливается страница. Это может быть исполняемый код, постранично загружаемый из файловой системы, или копия страницы, создаваемая в режиме дублирования при записи. Оба случая вполне типичны и не обязательно означают нехватку памяти. С другой стороны, операция выгрузки (поле so) всегда означает принудительное изъятие данных ядром.

Система, в которой непрерывно происходят операции выгрузки, скорее всего, выиграет от добавления памяти. Если же это случается лишь время от времени и не вызывает жалоб со стороны пользователей, то можно не беспокоиться. В остальных случаях дальнейший анализ будет зависеть от того, что нужно сделать, — оптимизировать систему для интерактивного режима (например, как рабочую станцию) или сконфигурировать ее для одновременной работы пользователей (например, как сервер приложений).

При использовании обычного жесткого диска можно считать, что каждые 100 операций выгрузки создают задержку примерно в одну секунду[4]. Соответственно, если для прокрутки окна требуется 150 операций выгрузки, придется ждать около полутора секунды. Исследователи пользовательских интерфейсов утверждают, что среднестатистический пользователь считает систему "медленной", когда время ее реакции превышает семь десятых секунды.

## Анализ операций обмена с диском

Производительность жесткого диска можно контролировать с помощью команды iostat. Как и vmstat, эта команда поддерживает дополнительные аргументы, задающие интервал времени в секундах между моментами выдачи статистических данных за истекший период и число повторений. Команда iostat выдает также сведения об использовании ЦП. Приведем небольшой пример для системы Solaris.

```
solaris$ iostat 5 5
   tty         sd0           sd1           nfs1           cpu
tin tout  kps tps serv  kps tps serv  kps tps serv  us sy wt  id
  0    1    5   1   18   14   2   20    0   0    0   0  0  0  99
  0   39    0   0    0    2   0   14    0   0    0   0  0  0 100
  2   26    3   0   13    8   1   21    0   0    0   0  0  0 100
  3  119    0   0    0   19   2   13    0   0    0   0  1  1  98
  1   16    5   1   19    0   0    0    0   0    0   0  0  0 100
```

Колонки разделены по темам (в данном случае их пять: tty, sd0, sd1, nfs1 и cpu). В разных системах данные, выводимые командой iostat, выглядят по-разному.

В разделе tty содержатся данные о терминалах и псевдотерминалах. В общем-то, это неинтересная информация, хотя она и может оказаться полезной для оценки пропускной способности модема. В колонках tin и tout дается среднее суммарное число символов, введенных и выведенных всеми терминалами системы за секунду.

[4] Мы считаем, что половину операций страничного обмена составляет выгрузка.

Информация о каждом жестком диске размещается в колонках kps, tps и serv: объем данных, пересланных за секунду (в килобайтах); общее количество пересылок за секунду; и среднее время позиционирования головки в миллисекундах. Одно обращение к диску может затронуть сразу несколько его секторов, поэтому соотношение между числами, приведенными в колонках kps и tps, говорит о структуре пересылок: то ли это несколько крупных пересылок, то ли множество мелких. Крупные пересылки более эффективны. Механизм вычисления времени позиционирования, похоже, работает только на некоторых дисковых накопителях и иногда дает странные результаты (к данному примеру это не относится).

Результат выполнения команды **iostat** в системах Linux, HP-UX и AIX может выглядеть примерно так.

```
aix$ iostat
...
Device:  tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
hdisk0  0.54        0.59        2.39    304483   1228123
hdisk1  0.34        0.27        0.42    140912    216218
hdisk2  0.01        0.02        0.05      5794     15320
hdisk3  0.00        0.00        0.00         0         0
```

Для каждого жесткого диска сообщается число операций ввода-вывода в секунду (tps), количество операций блочного чтения и блочной записи в секунду (Blk_read/s и Blk_write/s), общий объем прочитанных (Blk_read) и записанных (Blk_wrtn) блоков.

Размер дискового блока обычно равен 1 Кбайт (KiB), поэтому можно легко определить реальную скорость обмена данными с диском в килобайтах в секунду. В то же время понятие операции ввода-вывода несколько размыто. Один запрос на пересылку данных может включать в себя несколько запросов ввода-вывода к различным секторам.

Время, затрачиваемое на поиск блока, — основной фактор, влияющий на производительность жесткого диска. В первом приближении скорость вращения диска и быстродействие шины, к которой он подключен, не имеют особого значения. Современные диски могут пересылать сотни мегабайтов данных в секунду, если эти данные считываются из смежных секторов. Вместе с тем количество операций поиска составляет от 100 до 300 в секунду. Если после каждой операции поиска будет читаться один-единственный сектор, легко подсчитать, что в таком режиме задействуется менее 5% максимальной пропускной способности канала обмена данными с диском.

Затраты времени на поиск блока возрастают, если головке приходится перемещаться на большие расстояния. Когда есть диск с файловой системой, расположенной в нескольких разделах, и файлы считываются из каждого раздела в случайной последовательности, то для перехода из одного раздела в другой головка вынуждена преодолевать очень большой путь. С другой стороны, файлы в одном разделе расположены относительно близко друг к другу. Разбивая новый диск на разделы, необходимо принять во внимание факторы, влияющие на производительность, и постараться поместить файлы, обращение к которым осуществляется одновременно, в одну файловую систему.

Для достижения максимальной производительности нужно помещать файловые системы, которые используются одновременно, на разные диски. Хотя тип шины и драйверы устройств влияют на степень эффективности, большинство компьютеров может работать с несколькими дисками параллельно, что значительно повышает пропускную способность дисковой подсистемы. Например, данные и журнальные файлы веб-сервера имеет смысл размещать на разных дисках.

Особенно важно распределить область подкачки между несколькими дисками, если это возможно, поскольку страничный обмен обычно замедляет работу системы в целом. Многие системы позволяют создавать как выделенные разделы подкачки, так и файлы подкачки, записываемые в обычную файловую систему.

Кроме того, многие системы позволяют создавать "резидентные" файловые системы. Фактически это то же самое, что и виртуальный диск ПК. Специальный драйвер выдает себя за драйвер диска, хотя на самом деле записывает данные в память. Во многих организациях виртуальные диски применяются для хранения файловой системы `/tmp` и других часто используемых файлов, например журналов веб-сервера и почтовых буферов. Это приводит к уменьшению объема общедоступной памяти, но значительно ускоряет чтение и запись временных файлов. Как правило, такой подход оказывается весьма эффективным.

🕮 Подробнее о командах **lsof** и **fuser** см. в разделе 6.2.

Команда **lsof**, которая выводит список открытых файлов, и команда **fuser**, которая перечисляет процессы, использующие файловую систему, могут оказаться весьма полезными для выявления проблем, связанных с операциями обмена с диском. Эти команды отображают взаимодействия между процессами и файловыми системами и могут указать на непредусмотренные вами взаимосвязи. Например, если некоторое приложение регистрирует свои события на то же устройство, которое используется и журналами регистрации базы данных, то в результате этот диск может стать "узким местом" системы.

## Утилита xdd: анализ производительности дисковой подсистемы

Современные системы хранения информации могут включать сети или SAN-элементы, RAID-массивы и другие уровни абстракции. Для измерения и оптимизации этих сложных систем имеет смысл воспользоваться утилитой **xdd**, которая доступна под открытой лицензией (GPL) и выполняется во всех наших примерах систем (не говоря уже о Windows).

Утилита **xdd** измеряет параметры подсистемы ввода-вывода на отдельном компьютере и на кластерах систем. Она хорошо описана и обеспечивает точные и воспроизводимые измерения производительности. Подробнее о ней см. на сайте `ioperformance.com`.

## Команда sar: сбор статистических данных и генерирование отчетов по ним

Команда **sar** — предназначенный для мониторинга производительности инструмент, который "проверен временем" (эта команда появилась еще во времена AT&T UNIX) на системах UNIX и Linux и все еще остается актуальным, несмотря на использование "старого-доброго" синтаксиса командной строки.

На первый взгляд может показаться, что команда **sar** отображает ту же информацию, что и команды **vmstat** и **iostat**. Однако имеется одно очень важное отличие: **sar** "умеет" предоставлять отчеты как по старым (накопленным), так и текущим данным.

🕮 Пакет Linux, который содержит команду **sar**, называется **sysstat**.

Без опций команда **sar** предоставляет отчет о том, как ЦП использовался в течение того или иного дня каждые 10 минут, начиная с полуночи. Получение такой коллекции накопленных данных делает возможным сценарий **sal**, который является частью паке-

та **sar** и требует указания интервала времени, через который он должен запускаться из демона **cron**. Все данные, которые собирает команда **sar**, она сохраняет в бинарном формате в каталоге **/var/log**.

```
linux$ sar
Linux 2.6.18-92.ELsmp (bajafur.atrust.com)  01/16/2010

12:00:01  AM  CPU  %user  %nice  %system  %iowait  %idle
12:10:01  AM  all   .10   0.00    0.04     0.06   99.81
12:20:01  AM  all  0.04   0.00    0.03     0.05   99.88
12:30:01  AM  all  0.04   0.00    0.03     0.04   99.89
12:40:01  AM  all  0.09   0.00    0.03     0.05   99.83
12:50:01  AM  all  0.04   0.00    0.03     0.04   99.88
01:00:01  AM  all  0.05   0.00    0.03     0.04   99.88
```

Помимо информации о ЦП, **sar** также может предоставлять отчеты по метрическим показателям, таким как показатели дисковой и сетевой активности. Команда **sar -d** отображает сводку данных об активности диска за сегодняшний день, **sar -n DEV** — статистические данные о сетевом интерфейсе, а **sar -A** — всю доступную информацию.

Команда **sar** имеет некоторые ограничения и потому хорошо подходит только для быстрого получения кратких накопленных сведений. Тем, кто решил "всерьез и надолго" заняться мониторингом производительности, лучше установить специальную платформу с возможностями сбора коллекций данных и представления их в виде графиков, такую как платформа Cacti. Эта платформа "пришла к нам" из мира сетевого управления, но действительно умеет отображать произвольные метрические показатели системы, такие как показатели использования ЦП и памяти, в виде графиков. Пример отображаемого Cacti графика с дополнительными комментариями приводился в разделе 21.12.

## nmon и nmon_analyser: мониторинг производительности в системе AIX

В системах AIX утилиту `nmon` системные администраторы часто выбирают в качестве инструмента измерения и настройки производительности. Она во многом напоминает утилиту **sar**. Стивен Аткинс (Stephen Atkins), сотрудник компании IBM, разработал крупноформатную электронную таблицу **nmon_analyser**, которая обрабатывает данные, собираемые утилитой **nmon**. Данная утилита позволяет специалистам по производительности просматривать данные в формате большой таблицы, находить "плохие" данные и составлять графики, которые можно показать клиентам. Она выполняет более сложный анализ, чем утилита **sar**. Например, она вычисляет взвешенные средние числа для анализа пиковых точек, отображает загруженность устройства, размер считанных и записанных данных за день для системы хранения IBM и строит отдельные диаграммы для систем хранения EMC. И хотя утилита **nmon_analyser** официально не поддерживается компанией IBM, вы можете найти ее на сайте IBM:

`ibm.com/developerworks/aix/library/au-nmon_analyser`.

## Выбор Linux-планировщика ввода-вывода

В системах Linux используется алгоритм планирования подсистемы ввода-вывода, который выступает в роли "арбитра" между соревнующимися за дисковый ввод-вывод процессами. Он оптимизирует порядок и время запросов

для обеспечения наилучшей возможной производительности подсистемы ввода-вывода для данного приложения или ситуации.

В ядро Linux 2.6 встроены четыре разных алгоритма планирования. Можно выбирать любой. К сожалению, алгоритм планирования устанавливается во время загрузки (при помощи атрибута ядра `elevator=алгоритм`) и поэтому его не легко изменить. Алгоритм планирования системы обычно указывается в файле конфигурации начального загрузчика GRUB **`grub.config`**.

Все доступные алгоритмы перечислены ниже.

Comletely Fair Queuing (**`elevator=cfq`**). Этот алгоритм используется по умолчанию и обычно является наиболее подходящим вариантом для серверов общего назначения. Он старается равномерно предоставлять доступ к подсистеме ввода-вывода. (Во всяком случае этот алгоритм заслуживает награды за маркетинг: кто скажет "нет" совершенно справедливому планировщику?)

Deadline (**`elevator=deadline`**). Этот алгоритм "старается" сводить к минимуму время задержки для каждого запроса. Он изменяет порядок запросов для повышения производительности.

NOOP (**`elevator=noop`**). Этот алгоритм реализует простую очередь типа FIFO (First In, First Out — первым пришел, первым обслужен). Он предполагает, что запросы к подсистеме ввода-вывода уже были (или еще будут) оптимизированы или переупорядочены драйвером или устройством (каковым вполне может быть какой-нибудь контроллер со встроенной логикой). Он может быть наиболее подходящим вариантом в некоторых SAN-средах и для SSD-устройств.

Определив, какой из этих алгоритмов планирования является наиболее подходящим для данной среды (что может потребовать перепробовать все четыре варианта), возможно, вам удастся улучшить производительность подсистемы ввода-вывода.

## Программа oprofile: универсальный профилировщик системы Linux

Программа **`oprofile`** — это чрезвычайно мощная, интегрируемая в систему программа профилирования для систем Linux с ядром версии 2.6 или выше. Профилированию могут подвергаться все компоненты системы Linux: аппаратные и программные обработчики прерываний, модули ядра, само ядро, совместно используемые библиотеки и приложения.

При наличии свободного времени и желания точно знать, как используются ресурсы системы (до самых мельчайших деталей), стоит рассмотреть вариант установки **`oprofile`**. Эта утилита особенно полезна для тех, кто разрабатывает свои собственные приложения или код ядра.

В состав дистрибутива **`oprofile`**, который доступен для загрузки на сайте `sourceforge.net`, входят как модули ядра, так и ряд пользовательских утилит.

В начале 2010 года на горизонте появилась новая система отслеживания производительности. Известная как подсистема событий производительности ("perf events"), она обеспечивает уровень оснащенности, никогда ранее не виданный в ядре Linux. По всей вероятности, эта система в будущем заменит программу профилирования производительности в системах Linux **`oprofile`**.

## 29.5. Помогите! Моя система почти остановилась!

В предыдущих разделах речь шла, в основном, о средней производительности системы. Для ее повышения требуется корректировать конфигурационные параметры или модернизировать систему.

Но даже правильно сконфигурированные системы иногда работают медленнее обычного. К счастью, нерегулярные проблемы обычно легко диагностируются. В большинстве случаев они создаются "ненасытным" процессом, который потребляет так много ресурсов процессора, жесткого диска или сетевой подсистемы, что остальные процессы буквально останавливаются. Иногда процесс намеренно захватывает ресурсы, чтобы замедлить работу системы или сети. Это называется атакой типа "отказ в обслуживании".

Очень часто, чтобы определить, какой конкретно ресурс исчерпан, не нужно даже запускать диагностическую программу. Если система начинает "подвисать"[5] или подозрительно долго обращаться к диску, то проблема, скорее всего, связана с пропускной способностью дисковой подсистемы или дефицитом памяти. Если же производительность приложений падает ниже критического уровня, проблема, возможно, в загрузке ЦП.

Первый шаг в диагностировании проблемы — запуск команды **ps auxww** (**ps -elf** в Solaris и HP-UX) или **top** для выявления явно неуправляемых процессов. Любой процесс, отнимающий более 50% времени ЦП, можно с большой долей вероятности считать ненормальным. Если столь непомерную долю ресурсов ЦП не получает ни один процесс, посмотрите, сколько процессов получают минимум по 10%. Когда их больше двух-трех (не считая самой команды **ps**), средняя загруженность, скорее всего, будет очень высокой. Такая ситуация сама по себе является причиной низкой производительности. Проверьте среднюю загруженность системы с помощью команды **uptime**, а затем выполните команду **vmstat** или **top**, чтобы узнать, простаивает ли когда-нибудь процессор.

Если конкуренции за право использования центрального процессора не наблюдается, выполните команду **vmstat**, чтобы посмотреть, какова интенсивность страничного обмена. Следует обращать внимание на все операции обмена с диском: множество операций выгрузки может означать соперничество за память, а наличие дискового трафика в отсутствие страничного обмена говорит о том, что какой-то процесс монополизировал диск, непрерывно читая и записывая файлы.

Нельзя непосредственно сопоставить дисковые операции с выполняющими их процессами, но с помощью команды **ps** можно сузить круг "подозреваемых". Любой процесс, работающий с диском, должен отнимать какую-то часть времени ЦП. Как правило, всегда можно сделать обоснованное предположение о том, какой конкретно из активных процессов захватил ресурсы[6]. Для проверки своей теории на практике выполните команду **kill -STOP**.

Предположим, процесс-виновник найден. Что с ним делать дальше? Как правило, ничего. Некоторые операции действительно требуют много ресурсов и неизбежно замедляют работу системы, но это вовсе не означает, что они незаконны. Но с помощью

[5] Другими словами, требуется больше времени на переключение между приложениями, хотя скорость выполнения отдельных заданий приемлема.

[6] Ранее признаками этого служили большой размер области данных процесса, размещенной в виртуальной памяти, либо неестественно большой объем занимаемой процессом физической памяти, но появление совместно используемых библиотек сделало эти показатели не столь полезными. Команда *ps* не умеет отделять общесистемные совместно используемые библиотеки от адресного пространства отдельных процессов. Кажется, что многие процессы занимают десятки мегабайтов физической памяти, хотя на самом деле это не так.

команды **renice** можно изменить приоритет процесса, для которого ограничивающим фактором является быстродействие ЦП. Иногда правильная настройка приложения может привести к значительному снижению потребления ресурсов. Этот эффект особенно заметен в сетевых серверных программах, например в веб-приложениях.

Процессы, интенсивно использующие диск и память, нелегко поддаются воздействию. Команда **renice** обычно не помогает. Можно уничтожить или остановить процесс, но, на наш взгляд, такая мера оправдана только в экстренных ситуациях. Лучше прибегнуть к административной мере: попросить владельца запустить процесс позже.

Ядро позволяет процессу ограничивать объем потребляемой им самим физической памяти при помощи системного вызова **setrlimit**[7]. Эта возможность также доступна в оболочке C в виде встроенной команды **limit**. Например, команда

```
% limit memoryuse 32m
```

ограничивает использование физической памяти для всех последующих пользовательских команд тридцатью двумя мегабайтами (в системе Solaris вместо команды **memoryuse** используется команда **memorysize**). Ее действие примерно эквивалентно действию команды **renice** в отношении процессов, ограничивающим фактором для которых является объем памяти.

Если неуправляемый процесс не является источником снижения производительности, нужно проанализировать две другие возможные причины. Первая — перегрузка сети. Многие программы настолько тесно связаны с сетью, что иногда трудно понять, где речь идет о производительности системы, а где — о производительности сети. Подробная информация о средствах контроля над работой сети приведена в главе 21.

В некоторых случаях проблемы, связанные с перегрузкой сети, выявить сложно, потому что они возникают и исчезают очень быстро. Например, если на всех компьютерах сети каждый день в определенное время с помощью демона **cron** запускается какая-то сетевая программа, то в сети может возникать кратковременный, но серьезный затор. Все компьютеры "зависнут" секунд на пять, после чего ситуация нормализуется.

Вторая причина — задержки, связанные с серверами. UNIX- и Linux-системы постоянно обращаются к удаленным серверам NFS, Kerberos, DNS и т.п. Если сервер не работает или какая-то иная проблема привела к тому, что связь с ним стала ненадежной, это ощущают на себе все клиентские системы.

Предположим, что в интенсивно эксплуатируемой системе какой-то процесс каждые несколько секунд вызывает библиотечную функцию **gethostent**. Если сбой в работе DNS заставляет эту функцию тратить на свое выполнение две секунды, будет заметно общее снижение производительности системы. На удивление, большое число проблем с производительностью серверов связано с неправильной конфигурацией прямого и обратного поиска в DNS.

## 29.6. Рекомендуемая литература


- Cockcroft Adrian and Bill Walker. *Capacity Planning for Internet Services.* Upper Saddle River, NJ: Prentice Hall, 2001.
- Drepper Ulrich. *What Every Programmer Should Know about Memory.* `lwn.net/Articles/250967`.


[7] Более тонкое управление ресурсами может быть достигнуто посредством CKRM-функций (Class-based Kernel Resource Management — управление ресурсами ядра на базе классов); см. *ckrm.sourceforge.net*.

- Ezolt Phillip G. *Optimizing Linux Performance*. Upper Saddle River, NJ: Prentice Hall PTR, 2005.
- Johnson S., et al. *Performance Tuning for Linux Servers*. Indianapolis, IN: IBM Press, 2005.
- Loukides, Mike and Gian-Paolo D. Musumeci. *System Performance Tuning (2nd Edition)*. Sebastopol, CA: O'Reilly & Associates, 2002.
- Tufte, Edward R. *The Visual Display of Quantitative Information (2nd Edition)*. Cheshire, CT: Graphics Press, 2001.


## 29.7. Упражнения

29.1. Определите возможную причину проблем в каждом из перечисленных ниже случаев.

а) при переключении между приложениями возникает заметная пауза и слышно, как диск интенсивно перекачивает данные;

б) программа численного моделирования выполняется дольше, чем нужно, но системная память практически свободна;

в) пользователи интенсивно эксплуатируемой ЛВС жалуются на медленный доступ через NFS, но показатель средней загруженности сервера остается очень низким;

г) В процессе работы команда часто сообщает о нехватке памяти.

29.2. ★ Системы балансирования нагрузки способны существенно влиять на производительность серверов. Перечислите несколько механизмов, которые могут использоваться для распределения нагрузки.

29.3. ★ Перечислите четыре основных фактора, влияющих на производительность. Для каждого из них приведите пример приложения, способного легко захватить весь ресурс. Укажите способы решения проблемы в каждом конкретном случае.

29.4. ★ Напишите три простые программы (или сценарии). Первая должна увеличить процессорное время, выделяемое на выполнение системных задач, а вторая — на выполнение пользовательских задач. Третья не должна влиять ни на один из этих параметров, но должна иметь большую продолжительность выполнения (elapsed time). Используйте свои программы в сочетании с командами, описанными в разделе "Анализ использования центрального процессора" (выше в этой главе), чтобы узнать, что произойдет при воздействии на систему различными способами.

29.5. ★ Напишите две простые программы (или сценарии). Первая должна интенсивно выполнять операции чтения, а вторая — операции записи. Используйте свои программы с командами, приведенными в разделе "Анализ операций обмена с диском" (выше в этой главе), чтобы узнать, что произойдет при воздействии на систему различными способами. (В качестве дополнительного условия предложите каждой из своих программ по выбору использовать модель либо произвольного, либо последовательного доступа.)

29.6. ★ Выберите любые две программы, потребляющие заметный объем системных ресурсов. Воспользуйтесь командой `vmstat` и другими рассмотренными в этой главе средствами для анализа работы каждой программы. Укажите, какие действия программы приводят к повышенному потреблению ресурсов. Подтвердите свои выводы конкретными числами.

# Глава 30

# Взаимодействие с системой Windows

Сегодня администраторы очень часто работают со средами, в которых установлены одновременно и Windows-, и UNIX-системы. Поэтому они должны знать о том, что существует немало способов, которыми эти операционные системы могут помогать друг другу. Например, Windows-приложения, помимо всего прочего, могут запускаться с рабочего стола UNIX или получать доступ к находящимся на сервере UNIX принтерам и файлам, а UNIX-приложения могут отображать свои пользовательские интерфейсы на рабочем столе Windows.

Обе платформы имеют сильные стороны, и их можно заставить работать совместно. Windows — это популярная и удобная настольная система, способная выступать в роли "мостика" между пользователем и выходящим из стены сетевым кабелем. С другой стороны, UNIX — это надежная и масштабируемая инфраструктурная платформа. Так что давайте не будем ссориться, ладно?

## 30.1. Вход в систему UNIX из Windows

У пользователей может возникнуть желание поработать в сеансе оболочки C или **bash**, не вставая из-за своего компьютера Windows. Лучшим инструментом для удаленного доступа в системах UNIX и Linux является протокол оболочки безопасности SSH.

📖 О протоколе SSH более подробно рассказывалось в разделе 22.10.

Существует несколько реализаций протокола SSH для системы Windows. Протокол SSH также поддерживает передачу файлов, а пакет PuTYY включает две предназначенные для этой цели клиентские программы с интерфейсом командной строки: **psftp**

и **pscp**. Приверженцам системы Windows, которые никогда не имели дела с командной строкой, скорее всего, придется по душе имеющая графический интерфейс клиентская программа WinSCP, доступная по адресу: `winscp.net`.

Еще один неплохой вариант — установить общий пакет UNIX-on-Windows Cywin и запускать доступные в нем SSH-утилиты из эмулятора **rxvt**. Пакет Cygwin описывается в разделе 30.4.

Модную, не имеющую пока никаких негативных отзывов Java-реализацию протокола SHH под названием MindTerm предлагает компания AppGate (`appgate.com`). Для личного использования программа MindTerm доступна бесплатно. Она запускается на любой системе, которая поддерживает язык Java, и может конфигурироваться различными способами.

Среди коммерческих реализаций клиента протокола SSH нашим фаворитом является программа SecureCRT компании VanDyke Software. Ее можно приобрести на сайте `vandyke.com`. Программа SecureCRT поддерживает все необходимые терминальные функции. Кроме того, компания VanDyke предлагает прекрасное обслуживание заказчиков и стимулирует их предлагать новые функции. Как и в программе PuTTY, функции программы SecureCRT встроены в программное обеспечение протокола для передачи файлов SFTP.

Интересной функцией протокола SSH является способность перенаправлять порты TCP между клиентом и сервером. Например, эта функция позволяет администратору настроить на клиенте локальный порт, который будет перенаправлять входящее соединение на другой порт, находящийся на машине, доступ к которой возможен только с сервера. Хотя эта функция открывает множество новых возможностей, она потенциально опасна и потому должна учитываться администратором при предоставлении SSH-доступа к своему серверу. К счастью, ее можно отключить на стороне сервера и тем самым ограничить возможности SSH только доступом к терминалу и передачей файлов.

## 30.2. Получение доступа к удаленным настольным средам

Графические настольные среды в системе UNIX связаны с распространяемой бесплатно системой X Window System, которая не имеет никакого отношения к Microsoft Windows. Система X Windows была разработана в исследовательских центрах Массачусетсского технологического института в средине восьмидесятых годов прошлого столетия и принята как стандарт всеми производителями рабочих станций UNIX и дистрибутивов системы Linux. Она претерпела несколько серьезных обновлений; стабильная база была окончательно достигнута лишь в версии 11, которая была впервые опубликована в начале девяностых годов прошлого века. Номер версии протокола был добавлен к ее названию — "X11"; под этим именем она известна сегодня. (Слово "Windows", используемое само по себе, всегда — как в этой главе, так и в реальном мире — означает операционную систему Microsoft Windows.)

X11 — это клиент-серверная система. Сервер X Server отвечает за отображение данных на экране пользователя и за получение входных данных с клавиатуры или от мыши пользователя. Он взаимодействует с клиентскими приложениями через сеть. Вовсе не обязательно, чтобы сервер и клиенты запускались на одной и той же машине.

🕮 Архитектура X Windows более подробно описывалась в главе 25.

## Запуск сервера X Server на компьютере Windows

X11 — это богатый протокол, в который за последние годы было добавлено много расширений. Поэтому реализация сервера X Server является довольно сложной. Несмотря на это, сегодня реализации сервера X Server существуют практически для каждой операционной системы. Сама система X Windows безразлична к операционной системе, так что клиенты протокола X11, работающие на UNIX-компьютере, могут отображать сервер X Server, работающий под управлением системы Microsoft Windows, и позволять пользователю управлять ими точно так же, как если бы этот пользователь сидел за системной консолью.

К сожалению, первые разработчики протоколов X не уделили должного внимания вопросам безопасности. Каждая программа, которая подключается к серверу X Server, может считывать все, что вводит с клавиатуры обслуживающий этот сервер администратор, и видеть все, что он отображает на своем экране. Еще больше ухудшает дело то, что удаленным программам даже не нужно отображать окно при получении доступа к серверу X Server: они могут просто " тихонько прокрадываться" на заднем фоне.

Со временем для защиты протокола X11 было предложено несколько методов, но все они оказались несколько сложными. Вывод таков: лучше не разрешать никаких удаленных подключений к своему серверу X Server, если нет уверенности в том, что делаешь. Большинство серверов X по умолчанию конфигурируется так, чтобы они не допускали удаленных соединений, так что угрозы безопасности быть не должно, по крайней мере до тех пор, пока не будет запущена программа **xhosts** (или ее аналог), предоставляющая права на удаленный доступ.

К сожалению, предоставление прав на удаленный доступ — это единственный путь, когда нужно сделать так, чтобы программы выполнялись в системе UNIX, а их интерфейсы отображались в системе Windows. Как же запустить удаленное приложение, не предоставив разрешения на удаленный доступ серверу X Server? Наиболее распространенный метод — использовать функциональную возможность протокола SSH, которая была специально разработана для поддержки протокола X11. В таком случае между клиентами системы X, работающими на удаленном хосте, и локальным сервером X будет создан безопасный канал. Программы, запускаемые на удаленном хосте, будут автоматически отображаться на локальной машине, но благодаря чудодейственным операциям протокола SSH локальный сервер X будет воспринимать их так, будто они были запущены локально.

📖 О протоколе SSH более подробно рассказывалось в разделе 22.10.

Обратите внимание на то, что переадресация системы X будет работать, только если ее функции переадресации включены как на стороне сервера SSH, так и на стороне клиента SSH. Например, если используется SSH-клиент PuTTY в системе Windows, функцию переадресации протокола X11 можно включить прямо на его экране установки и потом на стороне сервера SSH (соответственно, на стороне клиента X11, т.е. на компьютере с системой UNIX), удостовериться в том, что файл **/etc/ssh/sshd_config** содержит такую строку.

```
X11Forwarding yes
```

В случае изменения конфигурации SSH-сервера, следует перезапустить процесс **sshd** для того, чтобы новая конфигурация вступила в силу. Параметр `X11Forwarding` предусмотрен в большинстве систем UNIX, реализующих протокол OpenSSH.

Как заметил наш технический редактор Дан Фостер (Dan Foster), перенаправление соединений в системе X в рамках протокола X11 "может быть мучительно медленным, даже в системе LAN, причем ситуацию ухудшает наличие задержек в сети". Альтернативой является система VNC.

В то время как компания Apple предоставляет бесплатную версию сервера X Server для операционной системы Mac OS X, компания Microsoft, к сожалению, ничего подобного для системы Windows не предлагает. Однако существует одна бесплатная версия сервера X Server, доступная на сайте проекта Cygwin (`cygwin.com`), которая работает очень хорошо после того, как ее сконфигурировать. Прекрасной альтернативой, допускающей намного более простое конфигурирование, является сервер Xming для системы Windows. К коммерческим версиям X Server для Windows относятся Exceed и X-Win32. Эти версии предлагают более простую процедуру конфигурации, но по довольно высокой цене.

## VNC: система виртуальных сетей

В конце девяностых годов предыдущего столетия несколько человек из исследовательского центра AT&T Labs в Кембридже (Великобритания) разработали систему для удаленного доступа к настольным системам под названием VNC. Их идеей было "подружить" считающиеся уже примитивными, но зато простые терминалы с современными оконными системами. В отличие от протокола X11, протокол VNC не работает с отдельными приложениями. Он создает виртуальный рабочий стол (или предоставляет удаленный доступ к существующему) в виде отдельного компонента. В системе VNC специальный сервер X11 функционирует на центральной машине, а для получения к нему доступа используется специальное приложение типа программы просмотра.

Компания AT&T опубликовала программное обеспечение VNC под либеральной лицензией на программное обеспечение с открытым исходным кодом. Это позволило другим разработчикам подхватить эту идею и создать дополнительные реализации сервера и программы просмотра, а также улучшить ограниченную пропускную способность протокола. Сегодня VNC-программы просмотра доступны для большинства устройств, имеющих хоть какие-нибудь средства для графического отображения. VNC-серверы для систем UNIX, Linux и Windows тоже доступны повсеместно.

Реализация VNC-сервера для системы UNIX, по сути, представляет собой эмулятор графического адаптера, который подключается к серверу X.Org X Windows. Запуск **`vncserver`** через учетную запись системы UNIX приводит к созданию нового виртуального рабочего стола, который функционирует на машине UNIX в своем собственном, отдельном, мире. После этого пользователи могут использовать VNC-программу просмотра и получать доступ к этому рабочему столу удаленно. Мы рекомендуем перед первым запуском сервера выполнить команду **`vncpasswd`**, чтобы установить пароль для соединения.

Протокол VNC не запоминает состояние и использует двоичное отображение. Следовательно, пользователи могут свободно присоединяться и отсоединяться. Более того, несколько программ для просмотра могут иметь доступ к одному и тому же серверу одновременно. Последнее свойство особенно полезно для поддержки удаленного доступа и отладочных настроек. Кроме того, оно облегчает доступ к консоли для системных администраторов.

VNC-серверы в мире Windows обычно не создают никакого дополнительного рабочего стола: они просто экспортируют стандартный рабочий стол системы Windows в том

виде, в котором он отображается на экране. Главным приложением для этой технологии является удаленная поддержка.

В наши дни первые создатели VNC уже владеют собственной компанией, которая называется *RealVNC* (`realvnc.com`). Авторы проекта UltraVNC (`ultravnc.com`) занимаются разработкой очень быстрой и многофункциональной реализации VNC-сервера для системы Windows, а участники проекта TightVNC (`tightvnc.com`) работают над повышением степеней сжатия. Все они обязательно обмениваются информацией между собой, поэтому между разными приложениями происходит взаимное обогащение.

Протокол VNC совершенствуется с учетом возможности расширения. Все программы просмотра и серверы могут работать друг с другом в любой комбинации: они подбирают наилучший вариант протокола, понятный обеим сторонам. Функциональные возможности, наличие которых зависит от конкретной реализации (такие, как возможность передачи файлов), обычно доступны только при использовании сервера и клиента из одного и того же проекта.

## Протокол RDP в системе Windows

Начиная с версии Windows 2000 Server каждая система Windows имеет техническую возможность предоставлять графический удаленный доступ нескольким пользователям одновременно. Компонент удаленного доступа, называемый *Remote Desktop,* использует протокол *RDP* (Remote Desktop Protocol — протокол для удаленных дисплеев) для обеспечения связи между клиентом и сервером. Клиенты протокола RDP в системе UNIX позволяют администраторам управлять системой Windows с рабочего стола UNIX. Это неоценимый инструмент для системных администраторов UNIX, которым приходится иметь дело с системой Windows.

Для того чтобы воспользоваться протоколом RDP, необходимо активизировать серверную сторону (систему Windows) и обеспечить клиенту доступ к ней. Для этого в системе Windows 7 следует зайти на панель управления System, щелкнуть на пиктограмме Remote и выбрать параметр на панели Remote Desktop.

# 30.3. Запуск системы Windows и Windows-приложений

Как указывалось в главе 24, коммерческий продукт VMware Server (`vmware.com`) позволяет запускать на оборудовании персонального компьютера несколько операционных систем одновременно. Программа VMware эмулирует все виртуальные "гостевые" машины поверх базовой операционной системы, в роли которой выступает либо Linux, либо Windows. Какой бы ни была базовая операционная система, на виртуальных машинах VMware может быть установлена любая Intel-совместимая операционная система. С точки зрения гостевой машины, операционная система работает точно так же, как она работала бы на "родном" аппаратном обеспечении, и приложения инсталлируются нормально. Другими средствами виртуализации являются программы KVM и VirtualBox, которые также запускают систему Windows и должны рассматриваться как средства для запуска Windows-приложений.

Система Wine (`winehq.com`) работает иначе. Она реализует в среде UNIX API-интерфейс программирования Windows, тем самым позволяя запускать Window-приложения прямо поверх системы X. Эта бесплатно доступная программа транслирует "родные" вызовы API-интерфейса Windows в их UNIX-аналоги и может делать это, не используя

никакого кода компании Microsoft. Кроме того, она поддерживает протоколы TCP/IP, устройства последовательного доступа и звуковые платы. Она работает в системах Linux, BSD, Mac OS и Solaris.

Многие приложения Windows запускаются без проблем, но некоторые можно заставить работать лишь при помощи определенных уловок; подробнее об этом можно узнать на официальном веб-сайте проекта Wine. К сожалению, заставить приложения выполняться под управлением системы Wine часто не так-то просто. Талантливые разработчики на сайте `codeweavers.com` написали небольшую коммерческую программу-инсталлятор, которая умеет вынуждать "упрямые" приложения Windows работать так, как надо.

Если выбранное приложение поддерживается программой CodeWeavers — замечательно. Если нет, нужно дать ему шанс, а вдруг оно сможет приятно удивить. Но если приложение не хочет работать само и никакие заготовленные подсказки отыскать не удается, будьте готовы потратить время, чтобы придать ему нужную форму, если уж решили сделать это своими силами. При наличии бюджетных средств можете попробовать договориться с CodeWeavers о помощи.

Коммерческой альтернативой программе Wine является программа Win4Lin компании NeTraverse. Разработчики программы Win4Lin утверждают, что она более устойчива и поддерживает немного больше приложений компании Microsoft, чем программа Wine. Однако она требует модификации ядра, в то время как программа Wine обходится без этого. Программу Win4Lin можно приобрести на сайте `win4lin.com`.

## Двухвариантная загрузка и почему ею не стоит пользоваться

Тем, кто хоть когда-нибудь устанавливал систему Linux на компьютер, на котором ранее была установлена Windows, несомненно, предлагалось создать конфигурацию с двухвариантной загрузкой. В наши дни такие конфигурации функционируют во многом так, как обещают. Сегодня даже можно монтировать Windows-разделы под управлением системы Linux и получать доступ к файловой системе Linux под управлением Windows. О том, как именно можно настроить конфигурацию с двухвариантной загрузкой, подробно рассказывалось в разделе 3.3.

Но не стоит спешить! Тем, кто действительно работает как с Windows, так и с UNIX и нуждается в доступе к ним, следует очень скептично относиться к конфигурации с двухвариантной загрузкой. Такие конфигурации представляют собой воплощение закона Мэрфи в его самом худшем виде: они, кажется, почти всегда загружают не ту операционную систему, и из-за этого даже самая простая операция обычно заканчивается выполнением нескольких повторных загрузок. Сегодня компьютеры стоят так дешево и удаленный доступ настраивается так легко, что, как правило, нет никакого смысла подвергать себя таким бесконечным мучениям.

## Альтернативы Microsoft Office

Несколько лет назад компания Sun выпустила новую версию пакета StarOffice, который является аналогом пакета Microsoft Office. Эта версия, также распространяемая с открытым исходным кодом, называется OpenOffice.org. В этот пакет входят основные офисные утилиты, такие как редактор электронных таблиц, текстовый редактор, пакет средств для создания презентаций и приложение для рисования. Эти программы могут

читать и записывать файлы, сгенерированные их Microsoft-аналогами. Загрузить пакет OpenOffice.org можно по адресу: `openoffice.org`.

Пакет OpenOffice.org доступен на всех основных платформах, включая Windows, Linux, Solaris, Mac OS X и многие другие версии UNIX. Если нужен пакет с контрактом о поддержке, можно приобрести пакет StarOffice, который отличается от OpenOffice.org только наличием более мощного механизма проверки правописания и базы данных.

Компания Google выпустила конкурентное приложение Google Apps. Кроме мощных приложений Gmail и Google Calendar, компания разработала текстовый процессор и электронные таблицы. Эти свободно распространяемые программы включают взаимодействующие функции, позволяющие пользователям одновременно редактировать документы, расположенные в разных местах. Поскольку все приложения компании Google запускаются в веб-браузере, их можно использовать буквально в любой системе. Кроме того, можно экспортировать и импортировать содержимое файлов в разных форматах, включая форматы приложений Microsoft Office.

## 30.4. Использование утилит командной строки в системе Windows

Чего многим пользователям системы UNIX не хватает при работе на системах Windows, так это командной строки. И не просто старого консольного приложения или чего-то вроде DOS Box, а настоящей программы **`xterm`**, поддерживающей изменение размеров, цветовые схемы, управление мышью и все модные управляющие последовательности.

Хотя автономного (т.е. не зависящего от системы X) родного порта **`xterm`** для системы Windows не существует, имеется небольшая аккуратная программа **`rxvt`**, которая очень близка к идеалу. Она является частью системы Cygwin, которую можно загрузить с сайта `cygwin.com`. Если вы установите сервер X Server, поставляемый с системой Cygwin, сможете пользоваться настоящей программой **`xterm`**.

Эта система распространяется под общедоступной лицензией GNU General Public License и включает довольно обширный дополнительный набор общих команд UNIX, а также переносимую библиотеку, которая реализует API-интерфейсы POSIX под Windows. Ее подход к согласованию обозначений командной строки и файловых систем UNIX и Windows довольно хорошо продуман и "умудряется" наделять родные команды Windows многими удобными возможностями оболочки UNIX. Cygwin не только позволяет пользователям UNIX чувствовать себя в Windows "как дома", но и упрощает запуск программного обеспечения UNIX под Windows. Подробнее о Cygwin можно узнать на сайте `cygwin.com`.

Пакет MKS Toolkit представляет собой коммерческий аналог Cygwin. Более подробную информацию о нем можно найти на сайте MKS, который находится по адресу: `mkssoftware.com`.

Список программ UNIX, которые могут сами запускаться на Windows, постоянно растет; на текущий момент в него уже входят Apache, Perl, BIND, PHP, MySQL, Vim, Emacs, Gimp, Wireshark и Python. Прежде чем пытаться заставить приложение работать под управлением системы Windows с помощью инструмента вроде Cygwin, поищите его "родную" реализацию.

## 30.5. Совместимость Windows со стандартами электронной почты и веб

В идеальном мире все бы пользовались для связи открытыми стандартами и были счастливы. Но наш мир не идеален, и многие ругают систему Windows за то, что она представляет собой кучу запатентованных протоколов и неисправных реализаций интернет-стандартов. Пожалуй, они частично правы, но, тем не менее, существует много областей, благодаря которым система Windows вполне может нормально функционировать в мире стандартов. Этими областями являются службы электронной почты и веб.

За всю богатую историю веб, многие корпорации пытались охватить и расширить веб-возможности, чтобы защититься от конкуренции и увеличить свои доходы во много раз. Компания Microsoft по-прежнему ведет такую борьбу на уровне браузера с его многочисленными расширениями, подходящими исключительно для Internet Explorer. На базовом уровне протокола HTTP, однако, система Windows и Windows-браузеры практически не зависят от платформы.

Компания Microsoft предоставляет свой собственный веб-сервер — IIS, но его производительность всегда была ниже производительности сервера Apache, запускаемого в системе Linux, причем намного. Тем, кто не "зациклился" на серверной технологии типа ASP, использовать Windows-машины в качестве веб-серверов вовсе не обязательно.

Что касается электронной почты, то компания Microsoft усиленно предлагает в качестве предпочтительной серверной технологии свой продукт Exchange Server. По правде сказать, возможности программы Exchange Server действительно превосходят возможности почтовых систем интернет-стандарта, особенно когда почтовые клиенты представляют собой компьютеры Windows, запускающие Microsoft Outlook. И это еще не все: программа Exchange Server также может использовать протокол SMTP для входящей и исходящей почты и передавать почту UNIX-клиентам при помощи стандартных протоколов IMAP и POP.

На стороне клиента программа Outlook и ее бесплатный аналог Outlook Express могут подключаться к серверам UNIX IMAP и POP (как и большинство аналогичных программ, созданных сторонними разработчиками). Экспериментируйте с любыми комбинациями и выбирайте наиболее подходящую. О протоколах POP и IMAP более подробно рассказывалось в разделе 20.1.

## 30.6. Совместное использование файлов при помощи Samba и CIFS

В начале 1980-х годов компания IBM разработала API-интерфейс, который позволял компьютерам в одной и той же подсети общаться друг с другом, используя не замысловатые номера, а имена. Этот интерфейс получил такое название: Network Basic Input/Output System (Сетевая базовая система ввода-вывода), сокращенно — NetBIOS. Его расширенная версия, включающая исходный базовый сетевой протокол транспортного уровня, была названа так: NetBIOS Extended User Interface (Расширенный пользовательский интерфейс NetBIOS), сокращенно — NetBEUI. API-интерфейс NetBIOS стал довольно популярным и потому был адаптирован для использования поверх ряда различных сетевых протоколов, таких как IPX, DECNet и TCP/IP.

Компании Microsoft и Intel разработали на базе NetBIOS протокол совместного использования файлов и назвали его "основным протоколом" (core protocol). Позже он был переименован в протокол Server Message Block (Блок сообщений сервера) или, сокращенно, SMB. Далее протокол SMB эволюционировал в систему CIFS (Common Internet File System — Общая межсетевая файловая система), которая, по сути, представляла собой улучшенную и настроенную для работы в глобальных сетях версию SMB. Эта система CIFS по сей день остается главной технологией для совместного использования файлов на компьютерах, работающих под управлением операционной системы Windows.

В мире Windows файловая система или каталог, который доступен через сеть, является совместно используемым ресурсом. Этот термин звучит немного странно для пользователей UNIX, но при описании файловых систем CIFS мы будем пользоваться именно им.

## Samba: сервер CIFS для UNIX

Чрезвычайно популярный пакет Samba распространяется на условиях открытой лицензии GNU и реализует серверную часть CIFS в системах UNIX и Linux. Он был создан австралийцем Эндрю Триджеллом (Andrew Tridgell), который путем "обратного" проектирования воссоздал код протокола SMB и опубликовал полученный результат в 1992 году. Здесь мы рассмотрим пакет Samba версии 3.

Сегодня пакет Samba хорошо поддерживается и постоянно совершенствуется для расширения его функциональных возможностей. Он предоставляет собой стабильный, надежный механизм для интеграции в сеть UNIX, состоящую из компьютеров, работающих под управлением системы Windows. Прелесть пакета заключается в том, что устанавливать его нужно только на сервере — на Windows-компьютере никакого дополнительного программного обеспечения не требуется.

Система CIFS предоставляет пять основных служб.

- Совместное использование файлов.
- Сетевая печать.
- Аутентификация и авторизация.
- Разрешение имен.
- Объявление служб ("обзор" сервера файлов и принтера).

Пакет Samba не только обрабатывает файлы с помощью системы CIFS, но и выполняет основные функции контроллера службы Windows Active Directory. Как контроллер домена, пакет Samba поддерживает такие функциональные возможности, как регистрация доменов Windows, роуминг профилей пользователей Windows и буферизация заданий на печать в системе CIFS.

Большинство функциональных возможностей пакета Samba реализуется двумя демонами: **smbd** и **nmbd**. Демон **smbd** реализует обслуживание файлов и заданий на печать, а также выполняет аутентификацию и авторизацию. Демон **nmbd** обслуживает другие основные компоненты системы CIFS: разрешение имен и объявление служб.

В отличие от файловой системы NFS, которая тесно связана с ядром, пакет Samba не требует модификации ядра и запускается исключительно как пользовательский процесс. Он устанавливает соединения с сокетами, через которые посылаются NBT-запросы, и ждет клиентских запросов на доступ к ресурсам. Как только запрос поступил и был аутентифицирован, демон **smbd** создает новый экземпляр самого себя, который работает от имени пользователя, сделавшего запрос. В результате все права доступа к файлам

(включая групповые права) остаются ненарушенными. Единственная дополнительная функциональная возможность, которую демон **smbd** реализует, — это сервис блокировки файлов, позволяющий клиентским персональным компьютерам придерживаться привычной для них семантики блокировок.

## Установка Samba

Пакет Samba может работать со всеми рассмотренными нами операционными системами.[1] Это относится и к системе Linux. Заплатки, документация и другие файлы можно загрузить с сайта `samba.org`. Обязательно следует убедиться в том, что используются новейшие из доступных для данной системы пакеты Samba, потому что дефекты предыдущих версий чреваты потерей данных и возникновением проблем с безопасностью.

Во всех системах придется отредактировать файл **smb.conf** (который должен находиться либо в каталоге **/etc/samba/smb.conf**, либо **/etc/smb.conf**), чтобы задать поведение пакета Samba. В этом файле указываются каталоги и принтеры, которые должны использоваться совместно, права доступа к ним и общие рабочие параметры пакета Samba. Кроме того, в состав пакета Samba входит образец файла **smb.conf**, который содержит подробные комментарии и может стать хорошей отправной точкой для задания новой конфигурации. Обратите внимание на то, что после запуска Samba проверяет свой конфигурационный файл каждые несколько секунд и загружает все появившиеся изменения.

Также важно помнить о проблемах безопасности, которые могут возникать при совместном использовании файлов и других ресурсов по сети. Для типичного сайта можно обеспечить базовый уровень безопасности, выполнив два действия.

- Явно указав, каким клиентам разрешается получать доступ к ресурсам Samba. За эту часть конфигурации отвечает находящаяся в файле **smb.conf** конструкция `hosts allow`. Всегда проверяйте, чтобы в ней содержались только необходимые IP-адреса (или диапазоны адресов).
- Заблокировав возможность получения доступа к серверу людьми, не работающими в данной организации. Samba использует шифрование только для аутентификации пароля. При транспортировке данных шифрование не применяется. В зависимости от природы хранящихся на сервере Samba данных, администратор может заблокировать возможность получения доступа к серверу за пределами организации, дабы исключить вероятность случайной загрузки файлов посторонними людьми. Это обычно делается на уровне сетевого брандмауэра; Samba использует UPD-порты 137-139 и TCP-порты 137, 139 и 445.

После выхода версии Samba 3 превосходная документация стала доступна на сайте `samba.org`.

Пакет Samba поставляется с разумными значениями по умолчанию для большинства его конфигурационных опций, поэтому для большинства сайтов будет вполне достаточно небольшого конфигурационного файла. При помощи команды **testparm -v** можно отобразить перечень всех конфигурационных опций Samba и значений, которые для них установлены на текущий момент. Этот перечень будет включать как параметры, заданные в файле **smb.conf**, так и параметры, установленные по умолчанию.

Параметры в файле **smb.conf** лучше не изменять, если только они не отличаются от параметров по умолчанию или имеется веская причина их заблокировать. Преиму-

[1] Компания HP предлагает производный от пакета Samba продукт, который называется HP CIFS Server. Его можно загрузить из хранилища программного обеспечения компании HP.

ществом такого подхода является то, что в таком случае при обновлении до более новой версии Samba текущая конфигурация будет автоматически адаптировать параметры, рекомендуемые авторами Samba.

Приняв во внимание сказанное выше, обязательно проверяйте, чтобы была включена функция шифрования паролей.

```
encrypt passwords = true
```

Эта опция отвечает за шифрование паролей, которыми обмениваются клиенты Windows и сервер Samba. На текущий момент она является параметром по умолчанию, и серьезной причины, по которой ее действительно лучше было бы отключить, не существует.

Функция шифрования вынуждает сервер Samba сохранять специальный, имеющий вид хеш-значения Windows-пароль для каждого пользователя. Windows-пароли работают совершенно не так, как UNIX-пароли, и поэтому использовать их из каталога **/etc/shadow** невозможно.[2]

Пакет Samba предоставляет для установки этих паролей специальную утилиту — **smbpasswd**. Например, попробуем добавить пользователя **tobi** и установить для него пароль.

```
$ sudo smbpasswd -a tobi
New SMB password: <пароль>
Retype new SMB password: <пароль>
```

Пользователи тоже могут изменять свои собственные пароли в системе Samba при помощи **smbpasswd**.

```
$ smbpasswd -r smbserver -U tobi
New SMB password: <пароль>
Retype new SMB password: <пароль>
```

В этом примере на сервере **smbserver** изменяется пароль пользователя **tobi**.

## Кодирование имен файлов

Начиная с версии 3.0 все имена файлов в пакете Samba кодируются в формате UTF-8. Тем, у кого сервер работает в режиме кодировки UTF-8, очень повезло: у них будет идеальная совместимость.[3] Тем, кто находится в Европе и до сих использует на своем сервере один из режимов кодировки ISO 8859, повезло меньше: их имена файлов, содержащие специальные символы, наподобие a, o, u или e, будут выглядеть довольно странно при вводе команды **ls** в каталоге, в котором эти файлы были созданы в пакете Samba с помощью кодировки UTF-8. Решением является указать пакету Samba использовать тот же режим кодировки символов, что и на сервере.

```
unix charset = ISO8859-15
display charset = ISO8859-15
```

Правильность режима кодировки лучше проверять сразу. Иначе количество файлов со странно закодированными именами начнет расти, и их исправление позже может оказаться довольно нелегкой задачей.

---

[2] При установлении соединения система Windows передает мандаты пользователя серверу Samba, поэтому пароли пользователя в пакете Samba выбираются так, чтобы они совпадали с их паролями в системе Windows.

[3] Проверить, может ли система работать в режиме кодировки UTF-8, можно при помощи команды **echo $LANG**.

## Аутентификация пользователей

В системах аутентификации Windows клиент не доверяет серверу, т.е. пароль пользователя никогда не передается по сети в виде открытого текста. Вместо этого система Windows использует алгоритм типа "запрос-ответ" протокола Kerberos. Клиент Windows также может аутентифицироваться у пакета Samba с помощью протокола Kerberos.

Если при входе в систему Windows предоставить имя пользователя и пароль, Windows сохранит их и будет пытаться использовать эту информацию для аутентификации везде, где это необходимо, при помощи соответствующего аутентификационного запроса. Таким образом, если пользователь имеет такое же имя пользователя и пароль на Windows-компьютере, как и на Samba-сервере, Samba не будет явно запрашивать у него пароль при получении им доступа к совместно используемым ресурсам Samba. Вся процедура аутентификации будет выполняться незаметно на заднем фоне.

Недостатком подхода с аутентификацией типа "запрос-ответ" является то, что серверу приходится сохранять пароли в формате, эквивалентном открытому тексту. В действительности серверные копии паролей локально шифруются, но это защищает их только от случайного просмотра. Злонамеренный пользователь, получивший доступ к зашифрованным таким образом паролям, сможет использовать их для получения доступа к ассоциируемым с ними учетным записям без какого-либо дополнительного "взламывания". Пароли Samba следует защищать даже более тщательно, чем файл **/etc/shadow**.

В сложных средах со множеством серверов Samba имеет смысл использовать централизованную службу каталогов, проверяющую, чтобы на всех серверах был указан одинаковый пароль. В качестве аутентификационных служб Samba позволяет использовать службу LDAP и Windows. Служба LDAP описана в главе 19.

Существует два основных способа объединить системы аутентификации Windows и UNIX. Во-первых, можно сконфигурировать сервер Samba так, чтобы он выступал в роли первичного контроллера домена Windows Active Directory (см. раздел 30.9). Во-вторых, можно установить на Windows-клиентах приложение pGina (`sourceforge.net/projects/pgina`). Это интеллектуальное приложение заменяет стандартную систему регистрации Windows на каркас, который поддерживает все стандартные службы аутентификации, включая LDAP и NIS.

## Совместное использование основных файлов

Когда у каждого пользователя имеется свой домашний каталог, имеет смысл сделать доступными для совместного использования домашние каталоги.

```
[homes]
comment = Home Directories (Домашние каталоги)
browseable = no
valid users = %S
writeable = yes
guest ok = no
```

Эта конфигурация, например, позволит пользователю **oetiker** получать доступ к своему домашнему каталогу через путь *****сервер_Samba***\oetiker** с любой системы Windows.

На некоторых сайтах установленные стандартные разрешения на домашних каталогах системы UNIX позволяют людям просматривать файлы друг друга. Поскольку пакет Samba ограничивает доступ на основании разрешений, имеющихся у файлов системы UNIX, пользователи системы Windows, заходящие через CIFS, тоже смогут просматри-

вать домашние каталоги друг друга. Однако, как показывает опыт, такое поведение часто смущает пользователей Windows и вызывает у них чувство, будто бы они находятся у всех на виду. Строка `valid users` в показанном выше фрагменте конфигурации указывает Samba предотвращать подключения к домашним каталогам других людей. Не добавляйте ее, если такое поведение вас не устраивает.

Пакет Samba обращается к своему волшебному разделу `[homes]` только в крайнем случае. Если в конфигурации для домашнего каталога пользователя явно указан какой-нибудь определенный совместно используемый ресурс, установленные там параметры перекрывают значения, указанные в разделе `[homes]`.

## Групповые ресурсы

Пакет Samba может отображать списки управления доступом в системе Windows либо в разрешения файлов, либо в списки ACL (если лежащая в основе файловая система поддерживает их). На практике концепция списков ACL часто оказывается слишком сложной для большинства пользователей. Поэтому администраторы обычно просто создают отдельный совместно используемый ресурс для каждой группы пользователей, которые в таковом нуждаются, и конфигурируют сервер Samba так, чтобы он сам заботился о предоставлении нужных разрешений. Дальше, когда пользователь пытается смонтировать этот сетевой каталог, Samba проверяет, является ли он членом соответствующей группы пользователей UNIX, и заменяет его уникальный идентификатор (UID) на идентификатор владельца этого группового сетевого ресурса (т.е. созданного специально для этой цели псевдопользователя). Рассмотрим пример.

```
[eng]
comment = Group Share for engineering
; Каждый, кто является членом группы eng, может получать доступ к этому
; совместно используемому ресурсу
comment = Group Share for engineering
; Каждый, кто состоит в группе eng, может получать доступ к этому
; совместно используемому ресурсу
; Пользователям придется входить в систему при помощи своей
; учетной записи Samba)
valid users = @eng
; Мы создали специальную пользовательскую учетную запись по имени "eng".
; Все записываемые в этот каталог файлы будут принадлежать этой учетной
; записи, а также группе eng
force user = eng
force group = eng
path = /home/eng
; Отключаем все ACL-списки NT, потому что мы не собираемся
; использовать их здесь
nt acl support = no
; Делаем так, чтобы у всех файлов были приемлемые разрешения
create mask = 0660
force create mask = 0660
security mask = 0000
directory mask = 2770
force directory mask = 2770
directory security mask = 0000
;Обычные параметры для совместного пользования
browseable = no
```

```
writeable = yes
guest ok = no
```

Подобного эффекта можно добиться и при помощи такой опции пакета Samba, как `inherit permissions`. Если активизировать эту опцию на совместно используемом ресурсе, все новые файлы и каталоги будут наследовать свои параметры от родительского каталога.

```
[eng]
comment = Group Share for engineering
path = /home/eng
nt acl support = no
browseable = no
writeable = yes
inherit permissions = yes
```

Поскольку теперь пакет Samba будет брать параметры из родительского каталога, важно не забыть правильно установить разрешения на корневом каталоге этого совместно используемого ресурса.

```
$ sudo chmod u=rw,g=rws,o= /home/eng
$ sudo chgrp eng /home/eng
$ sudo chown eng /home/eng
```

Обратите внимание на то, что при такой конфигурации все равно нужно создать псевдопользователя **eng**, который будет выступать в роли владельца данного совместно используемого каталога.

## Прозрачная переадресация при помощи MS DFS

MS DFS (Microsoft's Distributed File System — Распределенная файловая система производства компании Microsoft) позволяет каталогам внутри совместно используемого ресурса заставлять клиентов автоматически монтировать другие совместно используемые ресурсы, как только к ним будет получен доступ. Для постоянных пользователей систем UNIX и Linux в этом нет ничего удивительного, но для системы Windows эта концепция, в целом, является довольно революционной и неожиданной.

Приведем пример.

```
[global]
; Включить поддержку MS DFS для данного сервера Samba
host msdfs = yes
...
[mydfs]
; Эта строка указывает Samba искать символические ссылки
; в каталоге данного совместно используемого ресурса
msdfs root = yes
path = /home/dfs/mydfs
```

Для того чтобы настроить автоматическое монтирование, нужно создать соответствующие символические ссылки в каталоге **/home/dfs/mydfs**. Например, показанная ниже команда превращает "каталог" **jump** в ссылку на один из двух каталогов на других серверах. (Обратите внимание на одинарные кавычки. Они необходимы для защиты обратной косой черты.)

Если указывается несколько источников (как здесь), Windows будет выбирать какой-нибудь один из них. Другими словами, пользователи, получающие доступ к каталогу **\\server\mydfs\jump**, теперь фактически будут считывать файлы либо из источника

shareX на сервере serverX, либо из источника shareY на сервере shareY, в зависимости от того, какой из них будет доступен. Если файловые системы экспортируются с разрешениями не только на чтение, но и на запись, обязательно нужно позаботиться о механизме для синхронизации файлов. На эту роль может подойти приложение **rsync**.

Пакет Samba также позволяет делать так, чтобы все клиенты, получающие доступ к определенному источнику совместного пользования, перенаправлялись на какой-нибудь другой сервер. Windows-сервер такого не позволяет.

```
[myredirect]
msdfs root = yes
msdfs proxy = \\serverZ\shareZ
```

Обратите внимание на то, что система DFS будет работать только для тех пользователей, которые имеют одинаковое имя пользователя и пароль на всех задействованных серверах.

## Программа smbclient: простой клиент CIFS

Помимо множества серверных функциональных возможностей, в состав пакета Samba также входит удобная программа для пересылки (передачи) файлов с интерфейсом командной строки, которая называется **smbclient**. Этой программой можно пользоваться для получения прямого доступа к любому серверу Windows или Samba.

```
$ smbclient //redmond/joes -U joe
Password: <пароль>
Doman=[REDMOND] OS=[Windows 5.0] Server=[Windows 2000 LAN Manager]
smb: \>
```

После успешной регистрации на файловом сервере можно приступать к навигации и передаче файлов, используя для этого стандартные команды в стиле ftp (такие, как **get**, **put**, **cd**, **lcd** и **dir**).

### *Поддержка системы CIFS на клиентской стороне системы Linux*

Система Linux включает прямую клиентскую поддержку для файловой системы SMB/CIFS. Это значит, что в любой файловой системе Linux можно смонтировать CIFS-ресурс совместного пользования, который ядро будет понимать непосредственно.

```
# mount -t smbfs -o username=joe //redmond/joes /home/joe/mnt
```

Хотя эта функциональная возможность и является полезной, следует помнить о том, что система Windows воспринимает монтируемые сетевые каталоги как устанавливаемые определенным пользователем (отсюда и опция **username=joe** в показанной выше строке), в то время как система UNIX воспринимает их больше как принадлежащие системе в целом. Windows-серверы обычно не могут понять концепцию, подразумевающую, что к смонтированному сетевому Windows-каталогу могут получать доступ несколько разных пользователей.

С точки зрения UNIX-клиента, все файлы в смонтированном каталоге принадлежат пользователю, который его смонтировал. Другими словами, если сетевой каталог смонтировать от лица пользователя **root**, тогда все файлы в нем будут принадлежать пользователю **root** и, следовательно, обычные пользователи не смогут записывать в него никакие файлы на Windows-сервере.

Опции монтирования **uid**, **gid**, **fmask** и **dmask** позволяют настраивать эти параметры так, чтобы права владения и доступа более точно отражали предусматриваемую для

данного сетевого каталога политику. Подробнее об этом поведении можно узнать на странице руководства **mount.smbfs**.

Для того чтобы позволить пользователю самостоятельно монтировать сетевой Windows-каталог, можно добавить в файл **/etc/fstab** следующую строку.

```
//redmond/joes /home/joe/mnt smbfs
   username=joe,fmask=600,dmask=700,user,noauto 0 0
```

Благодаря указанной здесь опции `user`, пользователи теперь смогут монтировать файловую систему путем выполнения следующей команды.

```
$ mount /home/joe/mnt
```

**Команда mount** будет предлагать пользователю ввести пароль, прежде чем монтировать сетевой каталог.

Хотя стандартной сетевой файловой службой для UNIX является NFS, в некоторых ситуациях для обмена файлами между компьютерами UNIX и Linux лучше использовать Samba или CIFS. Например, опасно позволять пользователям выполнять NFS-монтирование корпоративных файловых систем со своих персональных ноутбуков[4]. Однако можно безопасно использовать CIFS для предоставления этим ноутбукам доступа к домашним каталогам их владельцев.

🕮 Об NFS более подробно рассказывалось в главе 18.

## 30.7. Совместное использование принтеров при помощи Samba

Простой способ обеспечить возможность совместного использования принтеров — добавить в файл **smb.conf** раздел `[printers]`:. Это заставит пакет Samba сделать доступными для совместного пользования все локальные принтеры. Для выполнения своей работы пакет Samba пользуется командами системы печати, но поскольку печать в системе UNIX не очень стандартизирована, может понадобиться указать Samba, какая именно система печати используется на сервере, путем установки для опции `printing` соответствующего значения. Перечень поддерживаемых на текущий момент систем печати доступен на справочной странице **smb.conf**.

```
[printers]
; Где должны сохраняться файлы, прежде чем они будут переданы системе печати?
path = /var/tmp
; Пользоваться принтерами разрешается всем
guest ok = yes
; Пусть Samba знает, что в данном случае совместно используемым ресурсом
; является принтер
printable = yes
; Показывать принтеры всем, кто их ищет
browseable = yes
; Указать Samba, какая подсистема печати используется в системе
printing = LPRNG
```

[4] Механизм безопасности в службе NFSv3 основан на идее, что пользователь не имеет прав суперпользователя на клиентском компьютере и что идентификаторы на клиентской и серверной сторонах совпадают. Для самоуправляемых машин это неестественно. Система NFSv4 лучше осуществляет отображение пользовательских идентификаторов и намного безопаснее.

Windows-клиенты теперь смогут пользоваться этими принтерами как сетевыми, точно так же как если бы те обслуживались Windows-сервером. Однако существует небольшая проблема. Windows-клиенты захотят знать, принтер какого именно типа им достался, и пригласят пользователя выбрать подходящий драйвер принтера. Это чревато поступлением довольно большого количества просьб о помощи от пользователей, которые не знают, как поступать в такой ситуации. А если какой-то из имеющихся принтеров требует драйвер, не поставляемый с Windows, тогда этих просьб будет еще больше.

🕮 Более подробно о печати рассказывалось в главе 26.

К счастью, сервер Samba можно сконфигурировать так, чтобы он предоставлял необходимые Windows-драйверы принтеров всем Windows-клиентам. Однако прежде чем это делать, нужно выполнить кое-какие подготовительные операции. Следует гарантировать, что пакет Samba будет вести себя как сервер печати, добавив соответствующие записи в раздел `[global]` файла **`smb.conf`**.

```
[global]
;Кто у нас отвечает за администрирование принтеров
printer admin = printadm
; Следующий параметр по умолчанию всегда имеет указанное справа значение
disable spoolss = no
; Не пытайтесь отобразить эту информацию; вы все равно
; не можете добавлять принтеры
show add printer wizard = no
; Если нужно, чтобы выполнять печать могли все
guest ok = yes
browseable = no
```

Теперь программа Samba знает, что является сервером печати, и будет воспринимать пользователя `printadm` как своего администратора.

Если вы собираетесь предоставлять драйверы принтера для своих Windows-клиентов, тогда у вас должно быть место для хранения этих драйверов. Таким местом может быть специальный сетевой каталог (share) `[print$]`.

```
[print$]
comment = Printer Driver Area
; Место для хранения драйверов принтеров
path = /var/lib/samba/printers
browseable = yes
guest ok = yes
read only = yes
;Кому разрешено администрировать репозиторий драйверов принтеров
write list = printadm
```

Прежде чем загружать драйверы принтеров на новый сервер печати, следует позаботиться еще о нескольких деталях на уровне системы. Во-первых, нужно удостовериться в том, что учетная запись **`printadm`** существует и имеет права на получение доступа к серверу Samba.

```
$ sudo useradd printadm
$ sudo smbpasswd -a printadm
```

Во-вторых, система Samba сможет сохранять драйверы принтеров только при условии, если соответствующая структура каталогов уже существует и ее владельцем является пользователь **`printadm`** (как указано в опции `write list`).

```
$ sudo mkdir -p /var/lib/samba/printers
$ sudo cd /var/lib/samba/printers
$ sudo mkdir W32X86 WIN40
$ sudo chown -R printadm .
```

Здесь возможно два варианта: либо загрузить драйверы принтеров с Windows-компьютера, либо воспользоваться средствами системы Samba и сделать это все из командной строки. К сожалению, простого способа узнать, что именно должно устанавливаться для конкретного драйвера, нет, поэтому в большинстве случаев рекомендуется первый вариант. Только если предстоит повторно устанавливать драйвер на множестве серверов, имеет смысл изучить процедуру его установки и научиться воспроизводить ее при помощи средств командной строки.

## Установка драйвера принтера из системы Windows

Для того чтобы установить драйверы принтеров с Windows-клиента, установите соединение с сервером Samba путем ввода в диалоговом окне Run, появляющемся после выбора в меню Start команды Run, следующей строки: `\\`*`сервер-Samba`*`.example.com`. Windows попросит ввести имя пользователя и пароль для входа на сервер Samba. Введите имя пользователя и пароль учетной записи `printadm`. Если все пойдет хорошо, появится окно со списком имеющихся на этом сервере сетевых папок.

Отыщите подпапку Printers; в ней должны отображаться принтеры, которые были сделаны доступными для совместного использования. Щелкните правой кнопкой мыши на каком-нибудь пустом месте между пиктограммами принтеров, чтобы отобразить диалоговое окно Server Properties, и добавьте свои любимые драйверы принтеров через вкладку Drivers.

Загружаемые драйверы будут помещаться в каталог, указанный в `[print$]`. На этом этапе можно будет быстренько заглянуть в свойства того или иного загруженного драйвера. Именно этот список файлов придется предоставлять утилите командной строки Samba, если когда-нибудь возникнет желание автоматизировать процесс загрузки этого драйвера.

Загрузив все необходимые драйверы, следует связать их с конкретными принтерами. Для того чтобы сделать это, отобразите панель Properties каждого принтера по очереди (щелкая правой кнопкой мыши и выбирая в появляющемся контекстном меню команду Properties) и выберите подходящий драйвер на вкладке Advanced. Затем откройте диалоговое окно Printing Defaults и измените параметры печати. Даже если указанные параметры печати вас устраивают, все равно внесите хоть какое-нибудь небольшое изменение, чтобы вынудить систему Windows сохранить структуры данных конфигурации на сервере Samba. Тогда система Samba будет предоставлять эти данные получающим доступ к принтеру клиентам. Невыполнение последнего шага может привести к сбоям на клиентах, из-за невозможности отыскать подходящую конфигурацию при попытке использования принтера.

## Инсталляция принтера из командной строки

Как вы уже, наверняка, догадались, некоторые из перечисленных шагов довольно трудно воспроизвести, не используя систему Windows, особенно это касается установки принтерных параметров по умолчанию. Но если на сервере Samba нужно установить сотни принтеров, все-таки имеет смысл попробовать сделать это из командной строки. Такой метод особенно хорошо подходит для PostScript-принтеров, потому что Windows-

драйвер для PostScript-принтеров работает правильно и без информации о конфигурации по умолчанию.

Итак, если вы знаете, какие файлы требуются для данного драйвера, сможете установить этот драйвер из командной строки. Для того чтобы сделать это, сначала скопируйте требующиеся файлы в каталог `[print$]`.

```
$ cd ~/mydriver
$ smbclient -U printadm '//samba-server/print$' -c 'mput *.*'
```

Далее назначьте драйвер определенному принтеру. Предположим, что используется простой PostScript-принтер со специальным PPD-файлом.

```
$ rpcclient -U printadm -c "\
adddriver \"Windows NT x86\" \"Our Custom PS:\
PSCRIPT5.DLL:CUSTOM.PPD:PS5UI.DLL:PSCIPT.HLP:NULL:NULL:PSCRIPT.NTF\"" \
samba-server
```

Символы обратной косой черты в конце строк позволяют разбить команду на несколько отдельных строк для ясности; вы можете опустить их и ввести команду в виде одной строки, если хотите. Символы обратной косой черты перед двойными кавычками указывают на наличие вложенных наборов кавычек.

В длинной строке приведенного выше примера содержится информация, отображаемая в диалоговом окне свойств принтерного драйвера, которое доступно, когда драйвер принтера устанавливается из Windows.

- Длинное имя принтера.
- Имя файла драйвера.
- Имя файла данных.
- Имя конфигурационного файла.
- Имя справочного файла.
- Имя языкового монитора (указывайте здесь значение NULL, если языкового монитора нет).
- Тип данных по умолчанию (указывайте здесь значение NULL, если такого типа данных нет).
- Разделенный запятыми список дополнительных файлов.

Сконфигурировать принтер так, чтобы он использовал один из загруженных драйверов, можно, выполнив такую команду.

```
$ rpcclient -U printadm -c "\
set driver \"myprinter\" \"Our Custom PS\"" samba-server
```

# 30.8. Отладка сервера Samba

Обычно сервер Samba работает нормально, не требуя никакого особого внимания. Но если проблема все-таки появилась, попробовать выявить ее источник можно при помощи двух таких основных источников отладочной информации, как журнальные файлы отдельных клиентов и команда **smbstatus**. Для начала нужно удостовериться в наличии соответствующих параметров журнальных файлов в своем конфигурационном файле.

```
[global]
; Комбинация символов %m указывает, что для каждого клиента должен
; записываться отдельный файл.
```

```
log file = /var/log/samba.log.%m
max log size = 1000
; Объем подлежащей регистрации (занесению в журнал) информации.
; Также можно указывать уровень регистрации для компонентов системы.
; (Здесь указывается, что в общем должен использоваться уровень 3,
; но для аутентификации должен использоваться уровень 10.)
log level = 3 auth:10
```

Чем выше уровень регистрации, тем больше объем отладочной информации. На занесение данных в журналы тратится определенное время, поэтому не стоит требовать слишком большого количества деталей до тех пор, пока они не понадобятся для отладки, потому что это может замедлить работу системы.

Ниже показаны журнальные записи, сгенерированные после одной неуспешной и одной успешной попыток соединения.

```
[2004/09/05 16:29:45, 2] auth/auth.c:check_ntlm_password(312)
  check_ntlm_password: Authentication for user [oetiker] -> [oetiker] FAILED
    with error NT_STATUS_WRONG_PASSWORD
[2004/09/05 16:29:45, 2] smbd/server.c:exit_server(571)
  Closing connections
[2004/09/05 16:29:57, 2] auth/auth.c:check_ntlm_password(305)
  check_ntlm_password: authentication for user [oetiker] -> [oetiker] ->
    [oetiker] succeeded
[2004/09/05 16:29:57, 1] smbd/service.c:make_connection_snum(648)
  etsuko (127.0.0.1) connect to service oetiker initially as user oetiker
    (uid=1000, gid=1000) (pid 20492)
[2004/09/05 16:29:58, 1] smbd/service.c:close_cnum(837)
  etsuko (127.0.0.1) closed connection to service oetiker
[2004/09/05 16:29:58, 2] smbd/server.c:exit_server(571)
  Closing connections
```

Команда **smbcontrol** позволяет изменять уровень отладки на функционирующем сервере Samba без внесения изменений в файл **smb.conf**. Рассмотрим пример.

```
$ sudo smbcontrol smbd debug "4 auth:10"
```

Эта команда установила бы для общих отладочных данных уровень 4, а для данных, связанных с аутентификацией, — уровень 10. Аргумент **smbd** указывает, что такие уровни отладки должны быть установлены для всех демонов **smbd** в системе. Выполнить отладку конкретного установленного соединения можно следующим образом: использовать команду **smbstatus** для выяснения того, какой демон **smbd** обслуживает данное соединение, а затем передать его идентификатор (PID) команде **smbcontrol**, чтобы та выполнила отладку именно этого соединения. При уровнях журнализации свыше 100 в журналах начнут появляться (зашифрованные) пароли.

Команда **smbstatus** отображает информацию об активных соединениях и заблокированных файлах. Эта информация может особенно пригодиться при выявлении проблем блокировки (например, для выяснения, кто из пользователей открыл файл **xyz** для чтения/записи в монопольном режиме). В первом разделе ее выходных данных перечисляются все ресурсы, к которым подключался пользователь, а во втором — любые активные блокировки файлов.

```
Samba version 3.0.5
PID         Username      Group        Machine
-------------------------------------------------------------------
13036       zauck         ee           zhaka (192.168.1.228)
29857       milas         guests       beshil (192.168.1.123)
```

```
Service      pid           machine      Connected at
-------------------------------------------------------------------------
milasa       29857         beshil       Fri Sep 3 17:07:39 2004
zaucker      13036         zhaka        Thu Sep 2 12:35:53 2004

Locked files:
Pid       DenyMode       Access       R/W     Oplock Name
-------------------------------------------------------------------------
29857     DENY_NONE    0x3          RDWR    NONE /home/milasa/hello.dba
13036     DENY_NONE    0x2019f      RDWR    NONE /home/zaucker/aufbest.doc
```

Если уничтожить демон **smbd**, ассоциируемый с определенным пользователем, все его блокировки исчезнут. Некоторые приложения умеют справляться с этим и будут просто заново ставить блокировку, если она им нужна. Другие приложения (такие, как MS Access) будут зависать или "умирать страшной смертью", требуя выполнения множества щелчков на стороне Windows только для того, чтобы их закрыли. И как бы странно это ни звучало, но такая процедура все-таки пока еще никогда не приводила к повреждению. Но в любом случае к заявлениям системы Windows о том, файлы были заблокированы каким-нибудь другим приложением, все-таки следует относиться с осторожностью. Нередко система Windows оказывается права, и проблему можно устранить, просто закрыв приложение на клиентской стороне, вместо того чтобы применять к нему "грубую силу" на стороне сервера.

# 30.9. Аутентификация службы Active Directory

Рабочие столы системы Windows, притаившиеся в вашей сети, скорее всего, используют систему Active Directory компании Microsoft для аутентификации, доступа к каталогам и выполнения других сетевых операций. Служба Active Directory собирает пользователей, группы, компьютеры и правила операционной системы под общим зонтиком, централизуя и упрощая системное администрирование.

Кроме того, это одна из основных причин, по которым система Windows занимает устойчивое положение на многих предприятиях. Система UNIX собрала воедино некоторые фрагменты этой головоломки, но ни одно из реализованных в системе UNIX решений не было так отшлифовано и широко признано, как служба Active Directory.

🕮 Информацию о системе PAM можно найти в разделе 22.5.

Пользуясь хитроумными способами, разработчики проекта Samba сделали много шагов в направлении поддержки службы Active Directory в системах UNIX и Linux. С помощью пакета Samba системы семейства Linux могут присоединять домен Active Directory и открывать доступ в систему по учетным записям, определенным в службе Active Directory и не отраженным в файле **/etc/passwd**.

Идентификаторы пользователей в системе Linux выведены из своих аналогов, существующих в системе Windows и известных как идентификаторы безопасности, или SID. С помощью системы PAM можно автоматически создать домашний каталог для пользователя, который его еще не имел. Эта система интеграции позволяет даже выполнять команду **passwd**, чтобы изменить пароль пользователя в службе Active Directory. Вся эта магия интеграции, присущая системе Windows, осуществляется компонентом пакета Samba под названием **winbind**.

Служба Active Directory включает в себя и расширяет несколько стандартных протоколов, в частности LDAP и Kerberos. Стремясь максимально удовлетворить запросы

администраторов информационных систем, компания Microsoft, к сожалению, пожертвовала совместимостью с оригинальными протоколами, создав токсичные зависимости сети веб от лицензионных технологий удаленного вызова процедур (RPC — Remote Procedure Call).

🕮 Информация о переключении службы имен изложена в разделе 19.5.

Для эмуляции поведения клиента службы Active Directory демон **winbind** вносит изменения в протоколы PAM, NSS и Kerberos. Он преобразует запросы на аутентификацию и получение системной информации в соответствующие форматы, характерные для продуктов компании Microsoft. С точки зрения системы UNIX, служба Active Directory — это просто другой источник информации о каталоге LDAP и данных аутентификации по протоколу Kerberos.

Вы должны выполнить следующие процедуры, связанные с конфигурированием, прежде чем система Linux сможет войти в сферу действия службы Active Directory.

- Инсталлируйте пакет Samba с поддержкой службы Active Directory и преобразованием идентификаторов.
- Конфигурируйте переключение службы имен **nsswitch.conf**, чтобы использовать демона **winbind** как источник информации о пользователе, группе и пароле.
- Конфигурируйте систему PAM для обслуживания запросов на аутентификацию с помощью демона **winbind**.
- Конфигурируйте службу Active Directory как сферу действия протокола Kerberos.

🕮 Более подробно система DNS описана в главе 17.

Клиенты службы Active Directory в системах UNIX и Linux также должны использовать контроллеры этой службы, чтобы обслуживать свои DNS-запросы и согласовывать свою работу с протоколом NTP. Кроме того, следует принять меры, чтобы полностью определенное системное имя домена было указано в файле **/etc/hosts**. В некоторых ситуациях в файл **etc/hosts** необходимо также добавить IP-адрес контроллера домена. Однако мы не рекомендуем это делать, если вы используете службу Active Directory для работы с системой DNS.

Демон **winbind** отлично работает в системе Linux, но система UNIX осталась за бортом. Нам известны отчеты о сайтах UNIX, в которых успешно развернута общая схема, описанная выше, но каждая система имеет свои тонкости, а интеграция в системе UNIX выполняется не так безукоризненно, как в системе Linux. Для систем UNIX мы предлагаем один из описанных ниже вариантов.

## Подготовка к интеграции службы Active Directory

Пакет Samba по умолчанию включен в большинство дистрибутивов системы Linux, но некоторые дистрибутивы не содержат службы отображения идентификаторов, необходимые для полной реализации клиента службы Active Directory. Компилируя модули из исходного кода, придется тщательно настраивать эти компоненты, поэтому мы рекомендуем инсталлировать готовые пакеты исполняемых модулей, если есть такая возможность.

С другой стороны, компоненты Samba должны быть как можно более современными. Интеграция службы Active Directory представляет собой одну из новейших функциональных возможностей пакета Samba, поэтому загрузка последней версии кода с сайта `samba.org` поможет избежать неприятных ошибок.

Если вы создаете систему Samba на основе исходного кода, настройте ее с помощью совместно используемых модулей **idmap_ad** и **idmap_rid**. Соответствующий аргумент в сценарии **./configure** должен иметь следующий вид.

```
--with-shared-modules=idmap_ad,idmap_rid
```

Соберите и инсталлируйте систему Samba с помощью уже знакомой вам последовательности команд **make**, **sudo make install**.

Если система инсталлирована правильно, библиотека **winbind** будет размещена в каталоге **/lib**.

```
ubuntu$ ls -l /lib/libnss_winbind.so.2
-rw-r--r-- 1 root root 21884 2009-10-08 00:28 /lib/libnss_winbind.so.2
```

Демон **winbind** останавливается и запускается с помощью обычных системных процедур. После изменения файлов **nsswitch.conf**, **smb.conf** или файла конфигурации протокола Kerberos **krb5.conf** его следует запустить повторно. Его не следует запускать, пока не выполнено конфигурирование других служб.

## Конфигурирование протокола Kerberos для интеграции службы Active Directory

Протокол Kerberos не удобен для сложной конфигурации, особенно на серверной стороне. К счастью, нам достаточно настроить только клиентскую сторону протокола Kerberos, что представляет собой намного более простую задачу. Файл конфигурации имеет имя **/etc/krb5.conf**.

Во-первых, следует убедиться, что полное системное имя домена включено в файл **/etc/hosts** и что протокол NTP использует сервер Active Directory для привязки по времени. Затем отредактируйте файл **krb5.conf** и добавьте в него приведенные ниже строки. Замените свой домен Active Directory на `ULSAH.COM`.

```
[logging]
     default = FILE:/var/log/krb5.log
[libdefaults]
     clockskew = 300
     default_realm = ULSAH.COM
     kdc_timesync = 1
     ccache_type = 4
     forwardable = true
     proxiable = true
[realms]
     ULSAH.COM = {
         kdc = dc.ulsah.com
         admin_server = dc.ulsah.com
         default_domain = ULSAH
     }
[domain_realm]
    .ulsah.com = ULSAH.COM
    ulsah.com = ULSAH.COM
```

В этом примере представляют интерес несколько значений. Даже несмотря на привязку по времени с помощью протокола NTP, допускается пятиминутное расхождение. Это дает определенный запас времени, если с протоколом NTP возникнет проблема. По умолчанию областью считается домен Active Directory, а центр распределения ключей

(KDC — key distribution center) конфигурируется как домен контроллера Active Directory. Для отладки удобно пользоваться файлом **krb5.log**.

Запросите билет у контроллера службы Active Directory, выполнив команду **kinit**. Укажите корректную учетную запись пользователя домена. Для проверки подойдет имя "administrator", хотя учетная запись может быть любой. Когда получите приглашение, введите пароль домена.

```
ubuntu$ kinit administrator@ULSAH.COM
Password for administrator@ULSAH.COM: <password>
```

Для демонстрации билета Kerberos следует использовать команду **klist**.

```
ubuntu$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: administrator@ULSAH.COM

Valid starting      Expires             Service principal
10/11/09 13:40:19   10/11/09 23:40:21   krbtgt/ULSAH.COM@ULSAH.COM
renew until 10/12/09 13:40:19

Kerberos 4 ticket cache: /tmp/tkt1000
klist: You have no tickets cached
```

Если билет выведен на экран, значит, аутентификация прошла успешно и протокол Kerberos сконфигурирован правильно. В данном случае билет действует еще 10 часов и может быть возобновлен через 24 часа. (Для отмены билета можно использовать команду **kdestroy**.)

Остальные параметры конфигурации можно найти на справочной странице **krb5.conf**.

## Программа Samba как член домена Active Directory

Как и остальные компоненты пакета Samba, конфигурация демона **winbind** записана в файле **smb.conf**. Конфигурируйте пакет Samba как член домена Active Directory с помощью параметра `security = ads`.

Ниже приведена работоспособная конфигурация. Мы задали область Kerberos и указали аутентификацию пакета Samba в контроллере домена. Мы также задали отображение идентификаторов с помощью параметра `idmap` в файле **smb.conf**. Обратите внимание на то, что конфигурация отдельных общих ресурсов в файле **smb.conf** производится отдельно от конфигурации служб аутентификации Active Directory.

```
[global]
   security = ads
   realm = ULSAH.COM
   password server = 192.168.7.120
   workgroup = ULSAH
   winbind separator = +
   idmap uid = 10000-20000
   idmap gid = 10000-20000
   winbind enum users = yes
   winbind enum groups = yes
   template homedir = /home/%D/%U
   template shell = /bin/bash
   client use spnego = yes
   client ntlmv2 auth = yes
   encrypt passwords = yes
```

```
winbind use default domain = yes
restrict anonymous = 2
```

Большинство параметров очевидно, а детали можно найти на справочной странице **smb.conf**.

Особого внимания заслуживает параметр `winbind use default domain`. Если вы используете несколько доменов Active Directory, то это значение должно быть равным `no`. Однако если вы используете только один домен, то присвоение этому параметру значения `yes` позволит пропустить домен в ходе аутентификации (например, можно использовать домен "ben" вместо "ULSAH\ben"). Кроме того, значение `winbind separator` задает альтернативу обратной косой черте при указании имен пользователей. Значение `workgroup` должно быть коротким именем домена. Например, значение параметра `workgroup` для домена `linux.ulsah.com` может быть равным `LINUX`. Задав конфигурацию системы Samba, запустите ее и службы **winbind** снова, чтобы новые установки вступили в силу.

И наконец, присоединим систему к домену; воспользуемся инструментом **net** из пакета Samba, который позаимствовал синтаксис у команды системы Windows с тем же именем. Команда **net** принимает в качестве аргументов протоколы для взаимодействия с системой Windows. Для службы Active Directory мы используем параметр **ads**.

Убедитесь, что билет существует, выполнив команду **klist** (и запросите его с помощью команды **kinit**, если билета еще нет), а затем для присоединения к домену используйте следующую команду.

```
ubuntu$ sudo net ads join -S DC.ULSAH.COM -U administrator
Enter administrator's password: <password>
Using short domain name -- ULSAH
Joined 'UBUNTU' to realm 'ulsah.com'
```

В командной строке мы указали сервер службы Active Directiry `dc.ulsah.com` (что необязательно) и учетную запись администратора. По умолчанию служба Active Directory добавляет новую систему в раздел `Computer` иерархии домена. Если система оказывается в разделе `Computers OU` внутри оснастки `Users and Computers` службы Active Directory, то операция присоединения к домену прошла успешно. Состояние системы можно также проверить с помощью команды **net ads status**. Дополнительные параметры, в том числе операции поиска протокола LDAP, указаны на справочной странице **net**.

Конфигурация переключения службы имен проста. Сначала следует проверить системные файлы **passwd** и **group**, а затем запустить службу Active Directory с помощью демона **winbind**. Этот трюк описывается в файле **nsswitch.conf** следующим образом.

```
passwd:   compat winbind
group:    compat winbind
shadow:   compat winbind
```

После конфигурирования системы NSS можно выполнить распознавание пользователя и группы в службе Active Directory с помощью команды **wbinfo**. Для того чтобы увидеть список пользователей домена, используется команда **wbinfo -u**, а список групп выводит команда **wbinfo -g**. Команда **getent passwd** показывает учетные записи пользователей во всех источниках в формате **/etc/passwd**.

```
ubuntu$ getent passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
...
bwhaley:x:10006:10018::/home/bwhaley:/bin/sh
```

```
guest:*:10001:10001:Guest:/home/ULSAH/guest:/bin/bash
ben:*:10002:10000:Ben Whaley:/home/ULSAH/ben:/bin/bash
krbtgt:*:10003:10000:krbtgt:/home/ULSAH/krbtgt:/bin/bash
```

Отличить локальных пользователей от учетных записей домена можно только по идентификатору пользователя и по пути `ULSAH` в домашнем каталоге, который можно увидеть в последних трех записях.

Если ваш сайт использует несколько доменов или параметр `winbind use default domain` не установлен, к учетной записи домена приписывается его короткое имя (например, `ULSAH\ben`).

## Конфигурация системы PAM

🕮 Общее описание системы PAM дано в разделе 20.5.

В данный момент мы настроили систему на взаимодействие со службой Active Directory с помощью программы Samba, но аутентификация еще не сконфигурирована. Настроить систему PAM на аутентификацию с помощью службы Active Directory довольно сложно, в основном, потому, что ее специфика зависит от конкретного дистрибутива системы Linux.

Общая идея заключается в конфигурировании демона **`winbind`** как модуля аутентификации для всех служб, которым необходима поддержка службы Active Directory. В некоторых дистрибутивных пакетах, например в системе Red Hat, все службы удобно настраиваются в одном файле. В других, например в системе Ubuntu, параметры конфигурации распределены по нескольким файлам. В табл. 30.1 перечислены соответствующие файлы для каждого дистрибутивного пакета системы Linux.

**Таблица 30.1. Файлы конфигурации системы PAM для поддержки демона winbind**

| Система | Аутентификация | Сеанс |
|---|---|---|
| Ubuntu | **common-account, common-auth, sudo** | **common-session** |
| SUSE | **common-auth, common-password, common-account** | **common-session** |
| Red Hat | **system-auth** | **common-auth** |

Для того чтобы включить аутентификацию **`winbind`**, добавьте строку

```
auth sufficient pam_winbind.so
```

в начало каждого файла. Исключение составляет файл **`common-password`** в системе SUSE, в котором ключевое слово `auth` необходимо заменить словом `password`.

```
password sufficient pam_winbind.so
```

Система PAM может автоматически создавать домашние каталоги при регистрации новых (для системы) пользователей. Поскольку пользователей службы Active Directory невозможно добавить стандартной командой **`useradd`**, которая обычно создает домашние каталоги, эта функциональная возможность оказывается довольно удобной. Добавьте в файл конфигурации сессии системы PAM, указанный в табл. 30.1, следующую строку.

```
session required pam_mkhomedir.so umask=0022 skel=/etc/skel
```

В этой конфигурации система PAM создает домашние каталоги с разрешением на доступ в восьмеричном виде, равным 755, и профилями учетных записей, скопированными из каталога **`/etc/skel`**.

Вы можете также ограничить доступ к локальной системе для пользователей, входящих в конкретную группу Active Directory. Для этого следует добавить в файл конфигурации сессии службы PAM следующие строки.

```
session required /lib/security/$ISA/pam_winbind.so use_first_pass
   require_membership_of=unix_users
```

В данном случае регистрироваться могут только пользователи из группы `unix_users` службы Active Directory.

## Альтернатива демону winbind

Несмотря на то что свободное подключение к службе Active Directory, описанное выше, работает вполне хорошо, оно уязвимо к ошибкам и выглядит довольно сложным. Для администраторов, желающих осуществить относительно простую инсталляцию и получить поддержку при устранении ошибок, существует альтернатива.

Программы компании Likewise Software автоматизируют работу демона **winbind**, переключение службы имен, протокола PAM и конфигурирование отображения идентификаторов для более чем 100 дистрибутивных пакетов системы Linux и вариантов системы UNIX, включая все системы, рассмотренные в книге. Кроме того, они включают в себя агента групповой политики, который позволяет проводить централизованное конфигурирование систем UNIX, оснащенных службой Active Directory. Несколько надстроек к графическим пользовательским интерфейсам, включая консоль управления и оснастку службы Active Directory, также упрощают инсталляцию и конфигурирование. Версии этого программного обеспечения с ограниченными функциональными возможностями доступны бесплатно, а полнофункциональные варианты и сопровождение предлагаются на коммерческой основе. Детали можно найти на сайте `likewise.com`.

Еще одна возможность, которая позволяет полностью обойтись без демона **winbind**, — это комплект инструментов Quest Authentication Services. Он предлагает те же самые функциональные возможности, что и программы компании Likewise, но имеет дополнительные функции для управления групповой политикой. Приготовьте свои бумажники, потому что инструменты Quest недешевы. Полный список продукции можно найти на сайте `quest.com/authentication-services`.

# 30.10. Рекомендуемая литература


- Terprstra, John H. *Samba-3 by Example: Practical Exercises to Successful Deployment 2nd Edition*. Upper Saddle River, NJ: Prentice Hall PTR, 2006. (Электронная версия этой книги доступна на сайте `samba.org`.)
- Terprstra, John H., Jelmer R. Vernooij. *The Official Samba-3 HOWTO and Reference Guide. (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2006. (Электронная версия этой книги доступна на сайте `samba.org`.)


# 30.11. Упражнения

30.1. Зачем может понадобиться блокировать доступ через Интернет к портам 137-139 и 445 на сервере Samba?

30.2. Инсталлируйте программу Cygwin на компьютере с системой Windows и воспользуйтесь утилитой **ssh** в эмуляторе **rxvt**, чтобы подключиться к компьютеру с системой UNIX. Какие изменения появились в программе PuTTY?

30.3. ☆ Сравните производительность работы клиента, имеющего доступ к файлам с помощью системы Samba, с производительностью клиента, имеющего доступ к файлам через "родной" сервер CIFS (т.е. компьютер с системой Windows). Если на этих двух серверах установлено разное аппаратное обеспечение, целесообразно нивелировать влияние аппаратного обеспечения, чтобы влияние программного обеспечения проявилось ярче. (Может потребоваться доступ с правами суперпользователя.)

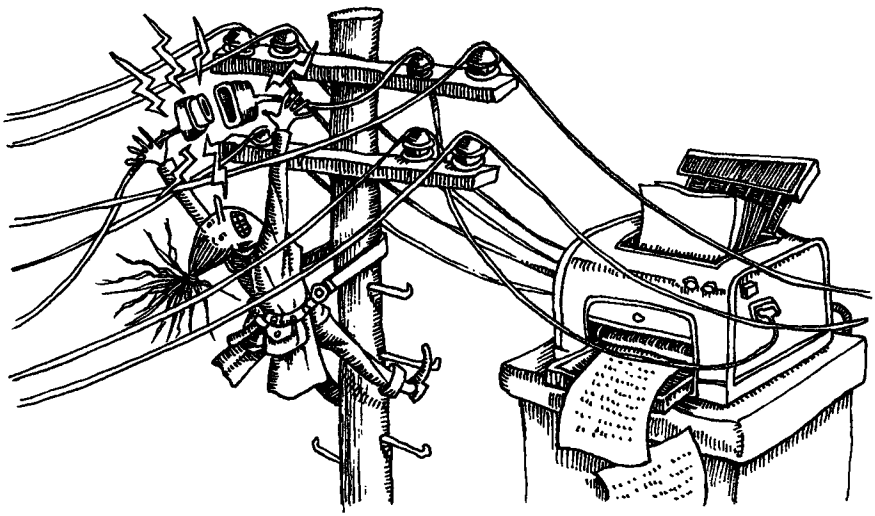
30.4. ★★ Воспользуйтесь анализатором пакетов (типа **tcpdump** или Wireshark) для перехвата **telnet**-сеанса между Windows и UNIX. Инсталлируйте программу PuTTY и повторите процедуру. Проанализируйте результаты обеих проверок. (Необходим доступ с правами суперпользователя.)

30.5. ★★ Сконфигурируйте сервер печати Samba так, чтобы он использовал драйверы принтеров Windows для всех обслуживаемых им принтеров. Удостоверьтесь в том, что принтеры имеют приемлемую конфигурацию по умолчанию.

30.6. ★★ Настройте систему так, чтобы она аутентифицировала среду Active Directory. Примите меры, чтобы была возможность изменять пароль и при регистрации новых пользователей домашние каталоги создавались автоматически.

# Глава 31

# Последовательные устройства и терминалы



Операционная система с сорокалетней историей, естественно, обросла хламом. К категории хлама можно отнести поддержку последовательных устройств, аргументируя это тем, что эту технологию доисторических времен лучше забыть. По сравнению с современными мультимегабитовыми последовательными интерфейсами, такими как USB, традиционные последовательные порты кажутся медленными и пустячными.

Понимание принципов работы последовательных интерфейсов — важная составляющая подготовки любого системного администратора. Хорошо ли, плохо ли, но интерфейс командной строки в системе UNIX основан на старинной концепции последовательного терминала и связанных с ней командах и управляющих устройствах, используемых и по сей день. Даже если вы никогда не приближались ближе 50 метров к кабельному терминалу, по-прежнему используете функциональные возможности операционной системы, которые его поддерживают. Например, консольное окно на рабочем столе системы UNIX или Linux на самом деле представляет собой псевдотерминал, поскольку оно является устройством, с которым вы устанавливаете соединение, регистрируясь в сети.

Настоящие последовательные порты RS-232 также используются до сих пор. Они больше не относятся к основным функциональным возможностям, но в некоторых ситуациях по-прежнему важны. Их можно использовать в качестве "общего знаменателя" для всех типов аппаратного обеспечения, от серверных менеджеров промышленного класса до встроенных систем миниатюрного масштаба, включая специальные проекты. Эту среду можно использовать для работы с устаревшими системами. В некоторых ситуациях их можно даже обнаружить под фальшполом.

В этой главе рассказывается о том, как подключать к системе устройства последовательного доступа RS-232 и использовать их в современной среде. В начале главы познакомимся с устройствами последовательного доступа и соответствующими кабель-

ными системами. Затем речь пойдет о программном обеспечении, которое традиционно применялось для поддержки аппаратных и псевдотерминалов, которые их эмулируют. Оставшаяся часть главы содержит базовые сведения об использовании последовательных консолей в системах UNIX и Linux.

# 31.1. Стандарт RS-232C

Большинство медленных последовательных портов работает в соответствии с одним из вариантов стандарта RS-232C. Этот стандарт определяет электрические характеристики и назначение каждого сигнального провода, а также разводку контактов традиционного 25-контактного последовательного разъемного соединения DB-25 (рис. 31.1).

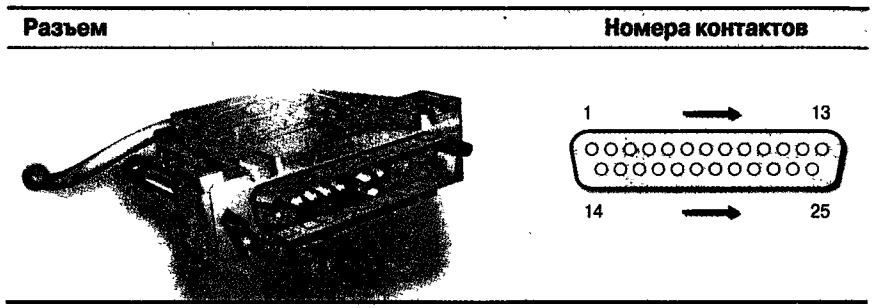



*Рис. 31.1. Вилка разъема DB-25*

Поскольку стандарт RS-232C[1] предназначен для распространения сигналов, многие из которых не используются в основных режимах передачи данных, в практической работе полный набор сигнальных проводов не используется. Кроме того, разъемные соединения DB-25 слишком громоздки. Поэтому сейчас широко применяются альтернативные модели 9-контактных разъемов, а не оригинальные 25-контактные. Кроме того, если возникает необходимость применять структурированные кабельные системы, альтернативой является разъем RJ-45. Эти разъемы описаны в разделе 31.2.

На рис. 31.1 изображена вилка DB-25. Во всех последовательных разъемных соединениях номера контактов на розетке зеркально отражают номера на вилке, чтобы при стыковке разъемов контакты с одинаковыми номерами совпадали. Схема в правой части рисунка соответствует показанной в левой части ориентации вилки.

Обратите внимание: у разъема, изображенного на рисунке, всего семь штырьков. Это наиболее типичный случай. Сигналы интерфейса RS-232 и соответствующие им контакты разъемного соединения DB-25 перечислены в табл. 31.1. На практике используются только сигналы 1–8, остальные можно игнорировать.

В отличие от стандартов разъемов, таких как USB и Ethernet, имеющих "защиту от дурака", разъем RS-232 требует от пользователя, чтобы тот знал, какой тип устройства к нему подключен. Для последовательных устройств существует две конфигурации кабельной системы: DTE (Data Terminal Equipment — терминальное оборудование) и DCE (Data Communications Equipment — коммуникационное оборудование). Эти конфигурации определяют, какие сигналы устройство будет ожидать на тех или иных контактах разъемного соединения.

[1] С технической точки зрения сегодня правильнее называть его стандартом EIA-232-E. Но если вы будете так говорить, вряд ли вас кто-нибудь поймет.

**Таблица 31.1. Сигналы интерфейса RS-232 и соответствующие им контакты разъемного соединения DB-25**

| Контакт | Сигнал | Контакт | Сигнал |
|---|---|---|---|
| 1 | FG — корпусная земля | 14 | STD — вторичный сигнал TD |
| 2 | TD — передаваемые данные | 15 | TC — синхронизация передачи |
| 3 | RD — принимаемые данные | 16 | SRD — вторичный сигнал RD |
| 4 | RTS — готовность к передаче | 17 | RC — синхронизация приема |
| 5 | CTS — готовность к приему | 18 | Не назначен |
| 6 | DSR — готовность данных | 19 | SRTS — вторичный сигнал RTS |
| 7 | SG — сигнальная земля | 20 | DTR — готовность терминала |
| 8 | DCD — обнаружение несущей | 21 | SQ — детектор качества сигнала |
| 9 | Положительное напряжение | 22 | RI — индикатор вызова |
| 10 | Отрицательное напряжение | 23 | DRS — селектор скорости передачи данных |
| 11 | Не назначен | 24 | SCTE — внешняя синхронизация передачи |
| 12 | SDCD — вторичный сигнал DCD | 25 | BUSY — занято |
| 13 | SCTS — вторичный сигнал CTS | | |

Устройства конфигурируются либо в режиме DTE, либо в режиме DCE, хотя некоторые поддерживают оба варианта (но не одновременно). Компьютеры, терминалы и принтеры чаще всего относятся к типу DTE, тогда как модемы являются DCE-устройствами. Последовательные порты DTE- и DCE-устройств могут взаимодействовать друг с другом в произвольных сочетаниях, но в каждом конкретном случае требуются разные кабели.

Смысла в одновременном существовании двух конфигураций нет, поскольку для всего оборудования может использоваться одна и та же разводка контактов. Просто это одно из многих бессмысленных наследий стандарта RS-232.

Ниже перечислены особенности обеих конфигураций.

- Разводка контактов в любом разъемном соединении RS-232 всегда одинакова, независимо от того, вилка это или розетка (штырьки всегда совпадают с соответствующими отверстиями) и где находится разъем: на кабеле, DTE- или DCE-устройстве.
- Спецификация RS-232 основана на модели прямого соединения DTE- и DCE-устройств. Под "прямым" соединением понимается подключение линии TD DTE-устройства к линии TD DCE-устройства и так далее, т.е. все одноименные контакты соединяются друг с другом.
- Названия сигналов отражают работу DTE-устройств. Например, название сигнала TD (transmitted data — передаваемые данные) в действительности означает "данные, передаваемые от DTE-устройства к DCE-устройству". Несмотря на это контакт TD служит для *приема* данных на DCE-устройстве. Аналогичным образом контакт RD является входным на DTE-устройстве и выходным на DCE-устройстве.
- Когда кабелем соединяются два DTE-устройства (компьютер и терминал либо два компьютера), их нужно заставить воспринимать противоположную сторону как DCE-устройство. Например, оба устройства будут предполагать передачу данных по линии TD и прием — по линии RD, поэтому необходимо соединить провода крест-накрест, связав выходной контакт одного устройства с входным контактом другого и наоборот.

- Подобное "перекрещивание" при соединении двух DTE-устройств требуется для трех групп сигналов: 1) TD и RD, о чем говорилось выше; 2) RTS и CTS; 3) контакт DTR должен быть соединен с контактами DCD и DSR на противоположном конце.

Кабель, соединяющий два DTE-устройства, называют *нуль-модемным*. Несмотря на это подключать к нему модемы нельзя, ведь модем — это DCE-устройство! Кабель для модемов называется просто "модемным" или "прямым".

На рис. 31.2 изображены разводки контактов и схемы соединений нуль-модемным и прямым кабелями. Показаны только "полезные" сигналы, используемые на практике.

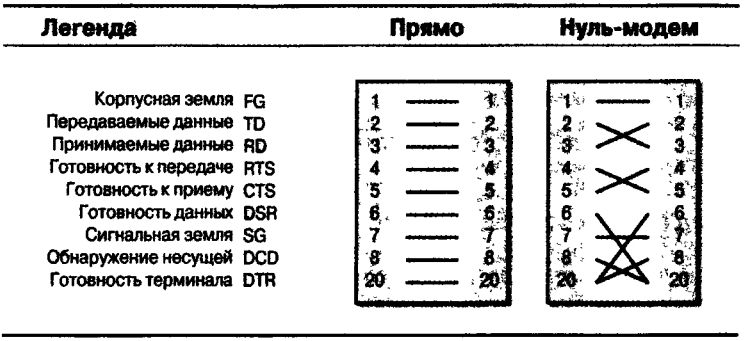



*Рис. 31.2. Разводки контактов и схемы соединения кабелей для разъемов DB-25*

# 31.2. Альтернативные разъемные соединения

В следующих подразделах описываются наиболее распространенные альтернативные разъемные соединения: DB-9 и RJ-45. Несмотря на конструктивные различия, эти соединения обеспечивают передачу тех же электрических сигналов, что и разъем DB-25. Устройства, в которых используются разные разъемы, всегда совместимы, если правильно выбран переходный кабель.

## Разъем DB-9

Девятиконтактный разъем DB-9 (внешне напоминающий уменьшенный DB-25) обычно применяется в персональных компьютерах. Он обеспечивает передачу восьми сигналов стандарта RS-232 (рис. 31.3).

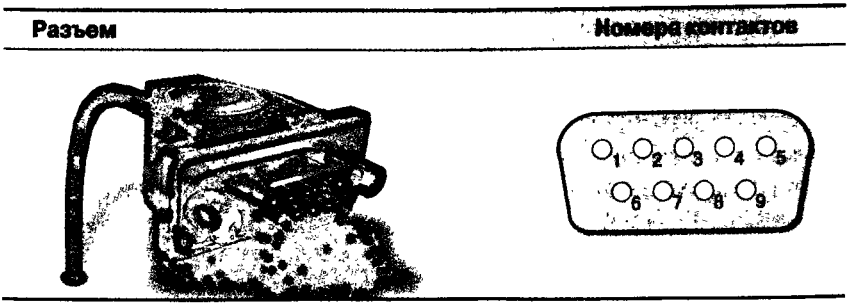



*Рис. 31.3. Вилка разъема DB-9*

У местных поставщиков персональных компьютеров обычно можно приобрести готовые переходные кабели DB-9/DB-25. Соответствующая разводка контактов представлена в табл. 31.2.

**Таблица 31.2. Разводка контактов для прямого кабельного перехода DB-9**

| DB-9 | Сигнал |
|---|---|
| 3 | TD — передаваемые данные |
| 2 | RD — принимаемые данные |
| 7 | RTS — готовность к передаче |
| 8 | CTS — готовность к приему |
| 6 | DSR — готовность данных |
| 5 | SG — сигнальная земля |
| 1 | DCD — обнаружение несущей |
| 4 | DTR — готовность терминала |

## Разъем RJ-45

RJ-45 — это 8-проводной модульный телефонный разъем. Он облегчает прокладку последовательных соединений через существующую разводку, если здание оборудовано витой парой Ethernet.

Гнезда RJ-45 практически не встречаются в компьютерах и обычных последовательных интерфейсах, но их часто применяют в качестве промежуточных соединителей при разводке линий последовательной передачи данных через коммутационные панели. Иногда их можно встретить в тех устройствах, где на небольшой площади размещено много портов (например, в терминальных серверах). Разъемы RJ-45, как правило, используются не с витой парой, а с плоским телефонным кабелем. Оба варианта допустимы в последовательных соединениях, но витая пара обеспечивает лучшее качество сигнала при передаче данных на большие расстояния. Для работы с разъемными соединениями RJ-45 требуется недорогой инструмент.

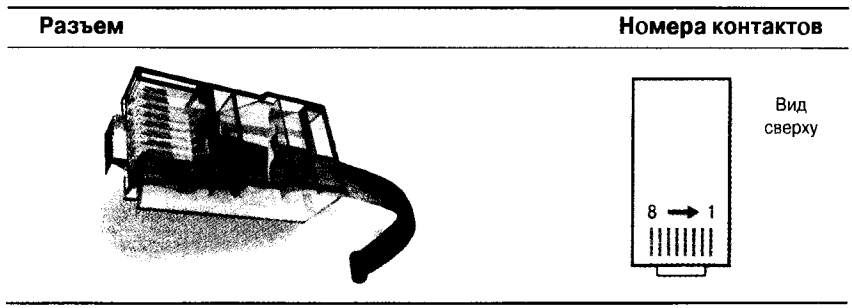



*Рис. 31.4. Вилка разъема RJ-45*

Существует несколько стандартов, определяющих соответствие контактов разъема RJ-45 контактам разъема DB-25. В табл. 31.3 описан официальный стандарт RS-232D, который почти не используется.

Другой стандарт — это система Дэйва Йоста (Dave Yost), в которой каждое устройство снабжено гнездом RJ-45 и используются стандартизированные кабели, позволяющие соединять как DCE-, так и DTE-устройства (`yost.com/computers/RJ45-serial`).

Таблица 31.3. Разводка контактов для прямого кабельного перехода RJ-45/DB-25

| RJ-45 | DB-25 | Сигнал |
|---|---|---|
| 1 | 6 | DSR — готовность данных |
| 2 | 8 | DCD — обнаружение несущей |
| 3 | 20 | DTR — готовность терминала |
| 4 | 7 | SG — сигнальная земля |
| 5 | 3 | RD — принимаемые данные |
| 6 | 2 | TD — передаваемые данные |
| 7 | 5 | CTS — готовность к приему |
| 8 | 4 | RTS — готовность к передаче |

## 31.3. Аппаратная и программная несущие

При подсоединении и включении устройства система UNIX ожидает, что сигнал обнаружения несущей (DCD) перейдет на высокий уровень (положительное напряжение). Этот сигнал подается на восьмой контакт разъемного соединения DB-25. Если в последовательном кабеле есть линия DCD и компьютер "обращает на нее внимание", значит, используется *аппаратная несущая*. В большинстве систем допускается также применение *программной несущей*, когда компьютер "делает вид", что сигнал DCD всегда присутствует.

Для некоторых устройств (в частности, для терминалов) программная несущая — это подарок судьбы. Она позволяет в каждом последовательном соединении обходиться всего тремя линиями: передаваемых данных, принимаемых данных и сигнальной земли. Но в модемных соединениях сигнал DCD обязателен. Если терминал подключен через модем и сигнал обнаружения несущей пропадает, модем должен разрывать соединение (особенно при междугородных звонках).

Режим программной несущей для последовательного порта обычно задается в конфигурационном файле (например, **/etc/gettydefs** или **/etc/inittab**, в случае регистрационного терминала, и **/etc/printcap**, в случае принтера) той клиентской программы, которая обслуживает этот порт. Если нужно включить этот режим "на лету", воспользуйтесь командой **stty -clocal**. Например, команда

```
suse$ sudo stty -clocal < /dev/ttyS1
```

включает программную несущую для порта **ttyS1**.

## 31.4. Аппаратный контроль передачи данных

Назначение сигналов CTS и RTS — обеспечить такую скорость передачи данных, чтобы приемное устройство успевало их обрабатывать. Например, если существует опасность переполнения буфера модема (возможно, из-за того, что соединение с удаленным узлом работает медленнее, чем последовательный канал между локальным компьютером и модемом), модем может приказать компьютеру сделать паузу, пока буфер не освободится.

Контроль передачи данных имеет большое значение для принтеров и высокоскоростных модемов. В системах, где аппаратный контроль передачи отсутствует (из-за того, что последовательные порты не поддерживают этот режим, либо из-за того, что в последовательном кабеле выводы CTS и RTS не подключены), его иногда можно имитировать программным путем с помощью управляющих ASCII-символов XON и XOFF. Однако

программный контроль должен явно поддерживаться высокоуровневым программным обеспечением, хотя даже в этом случае его реализация затруднена.

Символам XON и XOFF соответствуют комбинации клавиш <Ctrl+Q> и <Ctrl+S>. Это представляет проблему для пользователей редактора **emacs**, где при нажатии комбинации клавиш <Ctrl+S> по умолчанию вызывается команда поиска. Во избежание этого необходимо связать команду поиска с другой комбинацией клавиш или воспользоваться командами **stty start** и **stty stop** для изменения установок драйвера терминала.

Большинство терминалов игнорирует сигналы CTS и RTS. Те немногие терминалы, которые для установления связи требуют, чтобы по этим линиям была проведена процедура квитирования, можно обмануть, соединив перемычкой контакты 4 и 5 на том конце кабеля, который подключен к терминалу. В результате, когда терминал посылает сигнал на вывод 4, заявляя "я готов", с вывода 5 он получает этот же сигнал обратно, что означает "начинай". Таким же способом можно решить вопрос с квитированием по линиям DTR/DSR/DCD.

Как и в случае программной несущей, параметрами аппаратного контроля передачи можно управлять посредством конфигурационных файлов или команды **stty**.

На аппаратном обеспечении компании Sun контроль передачи данных для встроенных последовательных портов можно настроить с помощью команды **eeprom**.

На некоторых платформах компании HP возникает необходимость настраивать встроенные последовательные порты с помощью системы Guardian Service Processor (GSP).

## 31.5. Файлы последовательных устройств

Последовательные порты представляются в системе файлами устройств, расположенными в каталоге **/dev** или его подкаталогах. Даже сегодня у большинства компьютеров имеется два встроенных последовательных порта: **/dev/ttya** и **/dev/ttyb**, впрочем их часто называют **/dev/ttyS0** и **/dev/ttyS1** соответственно.

Иногда на один и тот же последовательный порт ссылается несколько устройств. Например, в системе **/dev/cua/a** означает тот же порт, что и **/dev/term/a**. Однако порт **/dev/cua/a** имеет другой, младший номер устройства.

```
solaris$ ls -lL /dev/term/a /dev/cua/a
crw - - - - - - - 1  uucp  uucp  37, 130172  Jan 11 16:35 /dev/cua/a
crw-rw-rw-        1  root  sys   37, 0       Jan 11 16:35 /dev/term/a
```

Как обычно, имена файлов устройств не имеют значения. Отображение устройства определяется его старшими и младшими номерами, а имена файлов устройств введены просто для удобства пользователей.

Несколько файлов устройств одновременно используется, в основном, для поддержки модемов, обрабатывающих входящие и исходящие звонки. В схеме Solaris драйвер позволяет открыть порт **/dev/term/a**, только когда модем получает сигнал на контакт DCD, означающий наличие активного (входящего) соединения (при условии что на порт не установлена программная несущая). Порт **/dev/cua/a** может открываться независимо от состояния контакта DCD; он используется при соединении с модемом, чтобы тот принял звонок. Если используется одно устройство, то доступ другого блокируется.

В системе HP-UX файлы последовательных устройств не всегда создаются автоматически. Для того чтобы заставить систему их найти, можно использовать следующую команду.

```
hp-ux$ sudo ioscan -C tty -fn
```

Файлы устройств можно также создать с помощью такой команды.

```
hp-ux$ sudo mfsf -H порт-с-вывода-устройства-ввода-вывода -d asio0 -a0 -i -v
```

Похоже, что система AIX целиком отказалась от поддержки последовательных интерфейсов. В частности, если в вашей системе несколько LPAR (см. главу 24), то последовательные интерфейсы не доступны по умолчанию. В этом случае вам, видимо, придется купить специальное оборудование.

## 31.6. Команда setserial: передача драйверу параметров последовательного порта в системе Linux

Последовательным портам персональных компьютеров можно назначать различные адреса ввода-вывода и номера запросов на прерывание (IRQ). Параметры портов считываются из BIOS при включении питания компьютера. Менять их приходится, только если какое-нибудь "вредное" устройство соглашается работать лишь при выделении ему ресурсов, обычно закрепленных за одним из последовательных портов. К сожалению, драйвер последовательных портов не всегда способен распознать подобные изменения конфигурации без посторонней помощи.

В системе UNIX эта проблема традиционно решается указанием параметров последовательного порта на этапе компиляции ядра. К счастью, в системе Linux этого можно избежать, воспользовавшись командой **setserial**. При наличии флага **-g** она отображает текущие установки порта.

```
ubuntu$ setserial -g /dev/ttyS0
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
```

Задавая параметры порта, нужно сначала указать файл устройства, а затем — последовательность названий параметров и их значений. К примеру, команда

```
ubuntu$ sudo setserial /dev/ttyS1 port 0x02f8 irq 3
```

устанавливает адрес ввода-вывода и номер прерывания для порта **ttyS1**. Важно уяснить, что эта команда ни в коей мере не меняет аппаратную конфигурацию системы. Она лишь сообщает конфигурационные параметры драйверу последовательных портов в системе Linux. Физические установки изменяются через системный BIOS.

Изменения, производимые командой **setserial**, не сохраняются при перезагрузке системы. Стандартного способа сделать их постоянными не существует; в каждом дистрибутиве это делается по-разному.

В системе Ubuntu для инициализации последовательных портов используется сценарий **/etc/init.d/setserial**. Он считывает параметры для каждого порта из файла **/var/lib/setserial/autoserial.conf**.

В системе SUSE для инициализации последовательных портов используется сценарий **/etc/init.d/serial**. К сожалению, этот сценарий не имеет файла

конфигурации; для того чтобы отразить команды, которые вы хотите выполнить, его приходится редактировать непосредственно. Бедная система SUSE! Этот сценарий написан на своем собственном метаязыке, позволяющем создавать строки команд **setserial**, но, к счастью, в нем достаточно много хорошо прокомментированных примеров.

В системе Red Hat сценарий **/etc/rc.d/rc.sysinit** проверяет наличие сценария **/etc/rc.serial** и, если он существует, выполняет его на этапе начальной загрузки. Готового примера сценария нет, поэтому его придется создать самостоятельно. Просто перечислите команды **setserial**, которые нужно выполнить, по одной на строку. Для надежности можно добавить в начало файла строку `#!/bin/sh` и сделать сам файл исполняемым, хотя это необязательно.

## 31.7. Псевдотерминалы

Терминалы с электронно-лучевыми трубками сегодня представляют собой не более чем музейные экспонаты, но их дух сохранился в виде псевдотерминалов. Эти пары файлов устройств эмулируют текстовый интерфейс терминалов со стороны служб, например, виртуальных консолей, виртуальных терминалов (например, **xterm**) и служб сетевой регистрации (например, **telnet** и **sh**).

Рассмотрим принцип их работы. Каждая пара файлов устройств имеет доступ к одному и тому же драйверу устройства внутри ядра. Подчиненное устройство имеет имя вроде **/dev/ttyp1**. Процесс, который мог бы нормально взаимодействовать с терминалом, например оболочка, использует подчиненное устройство вместо физического, например **/dev/ttyS0**. Главный процесс, например **sshd** или **telnetd**, открывает соответствующее главное устройство, в нашем примере — **/dev/ttyp1**. Драйвер псевдотерминала переносит коды нажатых клавиш и введенный текст между этими двумя устройствами, скрывая, что физического терминала не существует.

Несмотря на то что псевдотерминалы не имеют скорости двоичной передачи или стратегии управления передачей данных, к ним можно применить большинство других атрибутов и настроек, описанных в главе. Язык сценария **expect** использует псевдотерминал для управления процессами (например, **ftp** или **parted**), которые могут взаимодействовать с пользователем. Он довольно полезен для автоматизации некоторых административных задач.

## 31.8. Конфигурирование терминалов

За последнее десятилетие дешевые компьютеры практически вытеснили с рынка текстовые терминалы. Но даже "терминальные" окна, работающие в графическом режиме, используют те же драйверы и файлы конфигурации, что и реальные терминалы, поэтому системные администраторы должны знать, как они работают.

При настройке терминала нужно решить две важные задачи: закрепить за терминалом процесс, который будет принимать поступающие от него запросы на регистрацию, и обеспечить доступность информации о терминале после входа пользователя в систему. Однако прежде чем приступать к решению этих задач, давайте рассмотрим, как осуществляется регистрация в системе.

## Процедура регистрации в системе

В ходе регистрации задействуется несколько программ, самая важная из них — демон **init**. Одна из его задач заключается в порождении процесса, обобщенно называемого **getty**, на каждом терминальном порту, который определен в файле **/etc/inittab**. Демон **getty** устанавливает начальные характеристики порта (в частности, скорость передачи и режим контроля четности) и выводит на экран приглашение к регистрации.

О демоне **init** рассказывалось в разделе 3.5.

Реальное имя демона **getty** зависит от дистрибутива системы Linux, а в некоторых дистрибутивах существует даже несколько реализаций демона. В системах Red Hat и SuSE используется упрощенная версия, называемая **mingetty**, которая обрабатывает запросы, поступающие от виртуальных консолей. Для работы с терминалами и модемами предназначен демон **mgetty**, разработанный Гертом Дорингом (Gert Doering). В системе Ubuntu используется утилита **getty**, созданная Вице Венема (Wietse Venema). Она есть и в системе SuSE под названием **agetty**. Более старая реализация, называемая **uugetty**, в основном, вытеснена демоном **mgetty**. И наконец, популярный открытый сервер факсов HylaFAX (`hylafax.org`) имеет свою собственную версию демона **getty**, которая называется **faxgetty**.

Для того чтобы сориентироваться в этом многообразии, рассмотрим утилиты с точки зрения сложности. Проще всех демон **mingetty**. Он способен лишь обрабатывать запросы, поступающие от виртуальных консолей системы Linux. На следующем уровне иерархии находится демон **agetty**, который умеет обрабатывать запросы, поступающие от последовательных портов и модемов. Наконец, на вершине иерархии находится демон **mgetty**. Он может обрабатывать не только запросы на регистрацию, но и команды факсов, а также координировать работу модемов, чтобы один и тот же модем мог управлять как входными, так и выходными потоками.

Последовательность событий при регистрации в системе такова.

- Демон **getty** отображает содержимое файла **/etc/issue**, а также приглашение к регистрации.
- Пользователь вводит регистрационное имя в строке приглашения.
- Демон **getty** запускает программу **login**, передавая ей в качестве аргумента введенное имя.
- Программа **login** запрашивает пароль и сверяет его с записями в файле **/etc/shadow** или административной базе данных, например NIS или LDAP.
- Программа **login** выводит на экран "сообщение дня", хранящееся в файле **/etc/motd**, и запускает интерпретатор команд.
- Интерпретатор выполняет соответствующие сценарии конфигурации[2].
- Интерпретатор отображает на экране приглашение командной строки и переходит в режим ожидания команд.

Когда пользователь выходит из системы, управление возвращается демону **init**, который "пробуждается" и порождает новый процесс **getty** для порта терминала.

[2] Файл **.profile** — в случае **sh/ksh**; файлы **.bash_profile** и **.bashrc** — в случае **bash**; файлы **.cshrc** и **.login** — в случае **csh/tcsh**.

Большинство параметров каждого терминального порта сосредоточено в файлах каталога **/etc**. К ним относятся приглашение к регистрации и процесс **getty**, связанный с портом, скорость передачи двоичных данных и тип терминала, соединенного с портом.

К сожалению, среди разработчиков нет согласия по поводу конфигурации терминалов. В табл. 31.4 перечислены конфигурационные файлы, используемые в разных системах.

**Таблица 31.4. Конфигурационные файлы терминалов**

| Система | Вкл/выкл | Тип терминала | Параметры | Монитор |
|---|---|---|---|---|
| Ubuntu[а] | /etc/event.d/tty[б] | /etc/ttytype | /etc/getttydefs | getty |
| SUSE | /etc/inittab | /etc/ttytype | /etc/getttydefs | getty |
| Red Hat | /etc/inittab | /etc/ttytype | /etc/getttydefs | getty |
| Solaris [в] | _sactab | _sactab | zsmon/_pmtab | ttymon |
| HP-UX | /etc/inittab | /etc/ttytype | /etc/getttydefs | getty |
| AIX [г] | /etc/inittab | /etc/security/login.cfg | База данных ODM | getty |

[а] Для управления демоном **TTY/getty** в системе Ubuntu используется не демон **init**, а демон **ipstart**.

[б] Виртуальные консоли определены в файле **/etc/default/console-setup**.

[в] Конфигурационные файлы системы Solaris хранятся в каталоге **/etc/saf** и управляются демоном **sacadm**.

[г] Для обеспечения согласованности следует для модификации параметров TTY использовать систему SMIT.

## Файл /etc/ttytype

Во многих системах информация о типе терминала хранится в файле **/etc/ttytype**. Формат записи в файле **ttytype** имеет следующий вид.

*тип_терминала устройство*

Здесь *устройство* — это короткое имя файла устройства, соответствующего порту, а *termtype* — запись в базе данных **termcap** или **terminfo**. Когда вы регистрируетесь, переменной окружения TERM присваивается значение этого поля.

Рассмотрим пример файла **ttytype**.

```
wyse      console
dialup    ttyi0
dialup    ttyi1
vt320     ttyi2
h19       ttyi3
dialout   ttyi4
```

## Файл /etc/gettytab

Файл **gettytab** ассоциирует символические имена, например `std.9600`, с профилями конфигурации портов, содержащих такие параметры, как скорость, четность и предложение зарегистрироваться. Рассмотрим пример.

```
# Запись, используемая, чтобы задать значения по умолчанию
# для остальных записей, а также в ситуациях, когда демон getty
# вызывается без конкретного имени записи.

default:\
   :ap:lm=\r\n%h login\72 :sp#9600:

# Параметры фиксированной скорости
```

```
2|std.9600|9600-baud:\
   :sp#9600:
h|std.38400|38400-baud:\
   :sp#38400:
```

Этот формат совпадает с форматом файлов **printcap** или **termcap**. Строки с именами, разделенными вертикальными чертами (|), перечисляют имена каждой конфигурации. Другие поля в записях задают параметры, используемые последовательным портом.

## Файл /etc/gettydefs

Как и файл **gettytab**, файл **gettydefs** определяет конфигурации портов, используемые демоном **getty**. Каждая система может использовать только один из этих двух файлов. Файл **gettydefs** выглядит следующим образом.

```
console# B9600 HUPCL # B9600 SANE IXANY #login: #console
19200# B19200 HUPCL # B19200 SANE IXANY #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #9600
```

Формат записи имеет следующий вид.

*метка*# *инит_флаги* # *финал_флаги* # *приглашение* #*следующий*

Демон **getty** пытается сравнить свой второй аргумент с *меткой*. Если он вызывается без второго аргумента, то используется первая запись в файле. Поле *инит_флаги* содержит флаги **ioctl**(2), которые должны быть установлены на порту, пока выполняется регистрация. Поле *финал_флаги* устанавливает флаги, которые должны использоваться в дальнейшем.

В файле должна существовать запись, задающая скорость соединения в наборах флагов *инит_флаги* и *финал_флаги*. Набор флагов зависит от системы; полную информацию о них можно найти на справочных страницах **gettydefs** или **mgettydefs**.

Поле *приглашение* определяет строку приглашения зарегистрироваться, которая может содержать символы табуляции и перехода на новую строку с помощью обратной косой черты. Поле *следующий* задает метку записи **inittab**, которая должна заменяться текущей меткой, если поступила команда на прерывание. Это было полезно десятки лет тому назад, когда модемы не согласовывали скорость автоматически и вы должны были согласовывать ее вручную с помощью серии прерываний. Сегодня это уже анахронизм. Для аппаратного терминала поле *следующий* должно ссылаться на метку текущей записи.

Каждый раз, изменяя файл **gettydefs**, вы должны выполнять команду **getty -c gettydefs**, которая проверяет синтаксис этого файла, чтобы убедиться, что все записи являются правильными.

## Файл /etc/inittab

Демон **init** поддерживает различные "уровни выполнения", которые определяют, какие системные ресурсы задействуются. Существует семь уровней выполнения, пронумерованных от 0 до 6, плюс уровень "s", являющийся синонимом уровня 1 (однопользовательский режим). При выходе из однопользовательского режима демон **init** приглашает пользователя ввести номер уровня выполнения, если только в файле **/etc**

**/inittab** не задан параметр initdefault, как описано ниже. Затем демон просматривает файл **inittab** в поиске всех строк, соответствующих указанному уровню.

🕮 Демон **init** описан в разделе 3.5.

Уровни выполнения обычно устанавливаются таким образом, чтобы у пользователя был один уровень, на котором доступна только консоль, и другой уровень, на котором доступны все терминалы. Можно задавать уровни выполнения так, как того требует конкретная система, но мы рекомендуем не слишком отклоняться от значений по умолчанию.

Записи файла **inittab** имеют следующий формат.

*идентификатор*:*уровни_выполнения*:*действие*:*процесс*

Приведем примеры записей в файле **inittab**.

```
# Обработка команды <Ctrl+Alt+Del>
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# Запуск демонов getty на стандартных уровнях
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
```

Здесь поле *идентификатор* — это одно- или двухсимвольная строка, идентифицирующая запись. Поле идентификатора может быть пустым. Для терминалов в качестве идентификатора обычно указывают номер терминала.

В поле *уровни_выполнения* перечислены номера уровней выполнения, к которым относится запись. Если уровни не заданы (как в первой строке), то запись действительна для всех уровней. В поле *действие* указывается, как трактовать поле *процесс*; наиболее распространенные значения приведены в табл. 31.5.

**Таблица 31.5. Возможные значения поля *действие* файла /etc/inittab**

| Значение | Ждать? | Интерпретация |
|---|---|---|
| initdefault | — | Задает исходный уровень выполнения |
| boot | Нет | Процесс запускается при первом чтении файла **inittab** |
| bootwait | Да | Процесс запускается при первом чтении файла **inittab** |
| ctrlaltdel | Нет | Процесс запускается в ответ на нажатие комбинации клавиш <Ctrl+Alt+Del> |
| once | Нет | Процесс запускается однократно |
| wait | Да | Процесс запускается однократно |
| respawn | Нет | Процесс постоянно поддерживается в рабочем состоянии |
| powerfail | Нет | Процесс запускается при получении демоном **init** сигнала сбоя питания |
| powerwait | Да | Процесс запускается при получении демоном **init** сигнала сбоя питания |
| sysinit | Да | Процесс запускается перед обращением к консоли |
| off | – | Прерывает выполняемый процесс в некоторых системах |

Если одно из значений в поле *уровни_выполнения* совпадает с номером текущего уровня, а значение поля *действие* свидетельствует об актуальности записи, то демон **init** с помощью интерпретатора **sh** выполняет команду, заданную в поле *процесс* (или прекращает ее выполнение). В столбце "Ждать?" табл. 31.5 указано, в каких случаях демон **init** ожидает завершения команды, прежде чем продолжить работу.

В приведенном выше примере в последних двух строках порождаются демоны **mingetty** для первых двух виртуальных консолей (консоли переключаются нажатием

комбинаций клавиш <ALT+F1> и <Alt+F2>). При добавлении в систему аппаратных терминалов или модемов соответствующие записи файла **inittab** будут примерно такими же, только в качестве команды должна быть указана строка вызова демона **mgetty** или **getty** (**agetty** в SuSE), поскольку демон **mingetty** не работает с этими устройствами. Как правило, в качестве значения поля *действие* выбирается `respawn`, а в поле *уровни_выполнения* указывают `2345`.

Команда **telinit -q** заставляет демон **init** повторно прочитать файл **inittab**.

Различные версии демона **getty** конфигурируются по-разному. Версия **getty/agetty**, имеющаяся в SuSE и Ubuntu, немного проще в этом плане, чем **mgetty**, потому что берет свою конфигурационную информацию из командной строки (содержащейся в файле **/etc/inittab**).

Общий формат строки имеет следующий вид.

**/sbin/getty** *порт скорость тип_терминала*

Здесь *порт* — это файл устройства для данного порта (имя файла указано относительно каталога **/dev**); *скорость* — это скорость передачи в бодах (например, 38400), а поле *тип_терминала* определяет стандартный тип терминала, подключаемого к данному порту. Последнее поле является ссылкой на запись базы данных **terminfo**. Большинство эмуляторов имитирует работу терминала VT100 компании DEC, обозначаемого как **vt100**. Существует и много других параметров, в основном связанных с управлением модемами.

Демон **mgetty** обладает более сложными средствами работы с модемами, чем **agetty**, и к тому же понимает входные и выходные команды факсов. Его конфигурация рассредоточена среди большего количества файлов. Помимо обычных флагов командной строки, демон **mgetty** может принимать ссылку на файл **/etc/gettydefs**, в котором указаны детали конфигурации драйвера последовательных портов. Необходимость в этом возникает только при сложной настройке модемов.

Информацию о файле **gettydefs** можно получить с помощью команды **man mgettydefs**. Соответствующая страница названа так во избежание конфликта со старой **man**-страницей **gettydefs**, которая больше не существует ни в одной системе Linux.

Строка вызова демона **mgetty** для аппаратного терминала выглядит примерно так.

**/sbin/mgetty -rs** *скорость устройство*

Параметр *скорость* — это скорость передачи в бодах (например, 38400), а *устройство* — это файл устройства для последовательного порта (указывайте полный путь к нему).

Если требуется задать тип терминала по умолчанию, это придется сделать в отдельном файле, **/etc/ttytype**, а не в командной строке демона **mgetty**. Формат записей этого файла описан выше.

## Демон Upstart в системе Ubuntu

В системе Ubuntu демон **init** заменен его модифицированным вариантом Upstart, который запускает и останавливает службы в ответ на события. Однако выполняемый файл демона Upstart имеет прежнее имя **/sbin/init**.

Демон Upstart использует один файл для каждого активного терминала, указанного в файле **/etc/event.d**. Например, если мы хотим применить демон **getty** к терминалу ttyS0, то файл **/etc/event.d/ttyS0** может иметь следующий вид.

```
# ttyS0 - getty

# Эта служба поддерживает работу демона getty на терминале ttys0
# с момента запуска системы до ее остановки

start on runlevel 2
start on runlevel 3
start on runlevel 4
start on runlevel 5

stop on runlevel 0
stop on runlevel 1
stop on runlevel 6 respawn
exec /sbin/getty 38400 ttyS0
```

Дополнительные комментарии по поводу программы Upstart можно найти в разделе 3.5.

### *Система Solaris и демон sacadm*

Вместо традиционных демонов **getty** из системы UNIX, которые следили за работой каждого порта и выдавали приглашение зарегистрироваться, в системе Solaris существует запутанная иерархия Service Access Facility, управляющая мониторами TTY, мониторами портов и многими другими сложными, но почти бесполезными функциями.

Для того чтобы последовательный порт выдавал приглашение зарегистрироваться, сначала необходимо сконфигурировать "монитор", следящий за состоянием порта (**ttymon**). Затем следует сконфигурировать монитор порта, который следит за монитором TTY. Например, чтобы настроить монитор, работающий со скоростью 9 600 бод, на терминал **ttyb** и заставить его выдавать приглашения зарегистрироваться с помощью терминала типа VT100, необходимо выполнить следующие команды.

```
solaris$ sudo sacadm -a -p myttymon -t ttymon -c /usr/lib/saf/ttymon -v 1
solaris$ sudo pmadm -a -p myttymon -s b -i root -fu -v 1 -m "`ttyadm -d
  /dev/term/b -l 9600 -T vt100 -s /usr/bin/login`"
```

Файл **/etc/ttydefs** используется почти так же, как и файл **gettydefs** в других системах, чтобы задать параметры скорости и четности.

Информацию о демонах **saf**, **sacadm**, **pmadm**, **ttyadm** и **ttymon** можно найти на соответствующих справочных страницах и в главе о терминалах в книге *Solaris AnswerBook*. Развлекайтесь!

## 31.9. Специальные символы и драйвер терминала

Драйвер терминала поддерживает несколько специальных функций, доступ к которым осуществляется нажатием особых комбинаций клавиш (как правило, в эти комбинации входит клавиша <Ctrl>). Привязку функций к клавишам можно задать с помощью команд **tset** и **stty**. Некоторые из этих функций и соответствующие им комбинации клавиш приведены в табл. 31.6.

В зависимости от желания разработчика, значением по умолчанию для функции `erase` может быть либо комбинация клавиш <Ctrl+H>, либо символ удаления. (Реальная клавиша на клавиатуре может быть помечена как "backspace" или "delete" или стрелка, направленная влево.) К сожалению, существование двух различных стандартов порождает многочисленные проблемы.

**Таблица 31.6. Специальные символы и функции драйвера терминала**

| Название | Стандартная комбинация клавиш | Функция |
|---|---|---|
| erase | <Ctrl+?> | Стирает один введенный символ |
| werase | <Ctrl+W> | Стирает одно введенное слово |
| kill | <Ctrl+U> | Стирает всю строку |
| eof | <Ctrl+D> | Посылает терминалу признак конца файла |
| intr | <Ctrl+C> | Прерывает выполнение текущего процесса |
| quit | <Ctrl+\> | Уничтожает текущий процесс, создавая дамп памяти |
| stop | <Ctrl+S> | Останавливает вывод на экран |
| start | <Ctrl+Q> | Возобновляет вывод на экран |
| susp | <Ctrl+Z> | Приостанавливает текущий процесс |
| lnext | <Ctrl+V> | Игнорирует специальное значение следующего символа |

Можно воспользоваться командой **stty erase**, чтобы сообщить драйверу терминала о том, какой символ на самом деле используется. Однако некоторые программы (например, текстовые редакторы и командные интерпретаторы с режимом редактирования командной строки) имеют "собственное представление" о том, каким должен быть символ возврата, поэтому они не всегда обращают внимание на установку драйвера терминала. Часть программ поддерживает оба варианта. Может даже оказаться, что в локальной системе принята одна установка, а в удаленной системе, в которой вы зарегистрировались по сети, — другая.

В целом, не существует простого, универсального, решения подобных конфликтов. С каждой конкретной программой приходится разбираться отдельно. Есть два полезных ресурса на эту тему: документ *Linux Backspace/Delete mini-HOWTO* на узле `tldp.org` и статья Анне Баретта (Anne Baretta) по адресу: `ibb.net/~anne/keyboard.html`.

# 31.10. Команда stty: задание параметров терминала

Команда **stty** позволяет непосредственно изменять и запрашивать значения различных параметров драйвера терминала. Существует множество опций, большинство которых можно проигнорировать. Обычно названия опций команды **stty** совпадают с теми, которые указаны на **man**-странице **termios**, хотя бывают и расхождения.

Вот хорошее сочетание опций для простого терминала.

```
solaris$ stty intr ^C kill ^U erase ^H -tabs
```

Здесь параметр **-tabs** запрещает драйверу терминала задействовать встроенный механизм табуляции, поскольку многие эмуляторы не поддерживают его в полной мере. Остальные параметры назначают специальным символам прерывания (`intr`), удаления строки (`kill`) и удаления символа (`erase`), соответственно, комбинации клавиш <Ctrl+C>, <Ctrl+U> и <Ctrl+H> (возврат с удалением).

Команду **stty** можно использовать для анализа текущих режимов драйвера терминала и их включения. Команда **stty** без аргументов выдает такую информацию.

```
solaris$ stty
speed 38400 baud;
erase = ^H;eol = M-^?; eol2 = M-^? swtch = <undef>;
ixany
tab3
```

Более детальный отчет можно получить с помощью команды **stty -a**.

```
solaris$ stty -a
speed 38400 baud; rows 24; columns 80;
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D; eol = M-^?; eol2 = M-^?;
swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; dsusp = ^Y; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon - ixoff
-iuclc ixany imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab3 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echop rt
echoctl echoke
```

Формат вывода остался прежним, но отображена вся имеющаяся информация. О назначении параметров в большинстве случаев можно догадаться интуитивно.

Команда **stty** работает с файловым дескриптором своего стандартного входного потока, поэтому с помощью оператора переадресации, предусмотренного в интерпретаторе команд (<), можно устанавливать и запрашивать режимы не только текущего, но и других терминалов. Изменять режимы чужих терминалов имеет право только пользователь **root**.

## 31.11. Команда tset: автоматическое задание параметров терминала

Команда **tset** переводит драйвер терминала в режим, соответствующий типу терминала. Тип терминала можно задать в командной строке. Если он не указан, используется значение переменной окружения TERM.

Синтаксис команды **tset** предусматривает возможность изменения значений переменной окружения TERM. Это удобно, если вы часто регистрируетесь в системе через модем или коммутатор данных и хотите, чтобы драйвер был сконфигурирован в расчете на реальный терминал, который используется на другом конце соединения, а не на нечто общее и бесполезное вроде "коммутируемого терминала" (тип "dialup").

Предположим, к примеру, что на домашнем компьютере используется программа **xterm**, а система, в которой вы регистрируетесь по модему, воспринимает тип терминала модема как "dialup". Тогда, если добавить команду

```
tset -m dialup:xterm
```

в файл **.login** или **.profile**, драйвер терминала всегда будет настроен на работу с программой **xterm**.

К сожалению, команда **tset** не так проста, как кажется. Для того чтобы заставить ее не только задавать режим работы терминала, но и менять значения переменных среды, нужно воспользоваться следующим мини-сценарием.

```
set noglob
eval `tset -s -Q -m dialup:xterm`
unset noglob
```

Здесь подавляется обычный вывод команды **tset** (флаг **-Q**), а вместо этого запрашивается вывод команд интерпретатора, устанавливающих значения переменных среды (флаг **-s**). Поскольку присутствуют обратные кавычки, командные строки, отображаемые командой **tset**, перехватываются и подаются на вход интерпретатора благодаря

встроенной команде **eval**. В результате команды будут выполнены так, как если бы их ввел сам пользователь.

Команда **set noglob** запрещает интерпретатору раскрывать метасимволы, такие как '*' и '?', в выводе команды **tset**, а команда **unset noglob** восстанавливает нормальный режим работы интерпретатора. Это не нужно в интерпретаторах **sh/ksh**, поскольку обычно они не раскрывают специальные символы, которые заключены в обратные кавычки. Вид самой команды **tset** не зависит от интерпретатора: проверяя значение переменной среды SHELL, она определяет, какие команды следует генерировать.

## 31.12. Как справиться с "зависшим" терминалом

Некоторые программы (например, редактор **vi**) в процессе работы вносят серьезные изменения в параметры драйвера терминала. Пользователь, как правило, этого не замечает, поскольку при завершении или приостановке программы состояние терминала полностью восстанавливается. Однако существует опасность, что программа завершит работу аварийно или ее выполнение будет прекращено до завершения процедуры восстановления. Если это случится, терминал может начать вести себя очень странно: перестать правильно обрабатывать символы новой строки, отображать вводимые символы и адекватно выполнять команды.

Еще один способ нарушить работу терминала — применить к бинарному файлу команду **cat** или **more**. В большинстве таких файлов содержится причудливая смесь управляющих символов, которая, наверняка, окажется "смертельной" для недостаточно устойчивых эмуляторов.

Для того чтобы решить эту проблему, можно воспользоваться командой **reset** или **stty sane**. Первая из них является всего лишь ссылкой на команду **tset** и может принимать большинство ее аргументов, хотя, как правило, вызывается без аргументов. Обе команды — и **reset**, и **stty sane** — восстанавливают работоспособность драйвера терминала и посылают терминалу соответствующий код сброса, который берется из базы данных **termcap/terminfo**, если таковая имеется.

Во многих случаях сброс необходим потому, что терминал был оставлен в особом состоянии, когда вводимые пользователем символы не обрабатываются. В этом состоянии большинство терминалов при нажатии клавиши <Return> или <Enter> генерирует только символ возврата каретки (<Ctrl+M>), а символ новой строки, при получении которого текущая команда посылается на выполнение, не генерируется. Чтобы ввести символ новой строки непосредственно, вместо клавиши <Return> нажмите комбинацию клавиш <Ctrl+J> или клавишу перевода строки (если таковая имеется).

## 31.13. Отладка последовательной линии

Наладить работу последовательной линии несложно. Приведем перечень типичных ошибок.

- Вы забыли дать демону **init** указание повторно прочитать свои конфигурационные файлы.
- Вы забыли установить режим программного управления потоком при использовании трехпроводных кабелей.
- Вы используете несоответствующий нуль-модемный кабель.
- При пайке или обжиме проводов вы перевернули вверх ногами разъем DB-25.

- Вы подключили устройство не к тому проводу из-за того, что монтажные схемы оказались неправильными или неточными.
- Вы неверно задали параметры терминала.

Коммутационный тестер — незаменимое средство устранения проблем в кабельных системах. Он подключается к последовательной линии и показывает наличие сигналов в каждом проводе кабеля. У качественных коммутационных тестеров на каждой стороне есть и вилки, и розетки, поэтому они могут подключаться самыми разными способами. С каждым из представляющих интерес контактов связаны индикаторы, которые подсвечиваются, когда контакт активен.

Одни коммутационные тестеры позволяют только наблюдать за сигналами, другие дают возможность осуществлять повторную разводку соединений и подтверждают наличие напряжения на контакте. Например, если есть подозрение, что кабель нужно сделать нуль-модемным, то можно с помощью коммутационного тестера изменить имеющуюся разводку.

## 31.14. Соединение с консолями последовательных устройств

Вероятно, самым распространенным и полезным приложением порта RS-232 в настоящее время является соединение с последовательной "консолью" другого устройства. Этим устройством может быть все, что угодно, — от управляемого устройства UPS или коммутатора сети до встроенной системы Linux вроде системы TiVo в вашем телевизоре. Например, вы можете установить последовательное соединение с устройством UPS, которое служит источником электроэнергии для вашего оборудования в удаленном центре обработки данных, чтобы дистанционно отключать его в случае опасности.

Перечислим основные этапы соединения с последовательной консолью.

- Соедините кабелем последовательный порт в вашей системе UNIX и устройство, с которым хотите работать. Типы соединений и разъемов, которые могут вам понадобиться, описаны выше. Вероятно, вам также понадобится нуль-модемный кабель. Его можно купить в любом магазине, торгующем компьютерной техникой.
- Инсталлируйте или укажите программное обеспечение для работы с терминалом, которое вы будете использовать в системе UNIX или Linux. Десятки лет тому назад для этого использовались команды **cu** или **tip**. Вы можете использовать их и сегодня, но лучше применить современные варианты — **minicom** и **picocom**. Дистрибутивы системы Linux обычно содержат эти утилиты; в других системах вам, вероятно, придется самостоятельно инсталлировать это программное обеспечение (см. `freshmeat.net/projects/minicom` и `freshmeat.net/projects/picocom` соответственно).
- Конфигурируйте ваше программное обеспечение для работы с терминалом так, чтобы оно открывало правильный файл устройства (см. выше). Обычно этими файлами являются **/dev/ttya**, **/dev/tty1**, **/dev/ttyS0** или **/dev/S0**.
- Установите скорость передачи двоичных данных, биты остановки и поток управления в соответствии со стандартными установками целевого устройства. Эти параметры обычно описаны в документации по устройству, но вы можете попробовать применить все возможные комбинации. Если вы не знаете правильной скорости передачи двоичных данных, примените старый трюк: установите соединение и на-

берите несколько символов. Если вам приходится набирать несколько символов, чтобы получить один символ мусора, значит, скорость передачи данных слишком высокая. Если набор одного или двух символов создает много символов мусора, значит, скорость передачи данных слишком низкая. Тсс... не говорите никому!

- Если вы успешно установили соединение, то должны иметь возможность вводить команды на удаленной консоли. Если устройство вдруг надолго зависает, значит, вы неправильно настроили поток управления; иногда в этих ситуациях помогает нажатие клавиш <Ctrl+Q>.

Если возникли проблемы с соединением, в первую очередь удалите перекрещивание в кабеле или добавьте еще один кабель, если одного недостаточно. Не забывайте, если вы соединяетесь с удаленным блоком UNIX, необходимо настроить демон **getty** на удаленном конце, чтобы он прослушивал ваше соединение и выдавал приглашение зарегистрироваться.

# 31.15. Упражнения

31.1. Что такое нуль-модемный кабель? Как он используется при подключении DCE- и DTE-устройств?

31.2. Можно ли использовать трехпроводной последовательный кабель для подключения модема? А для последовательного принтера? Почему?

31.3. Как осуществляется аппаратный контроль передачи данных? Что можно сделать, если система не поддерживает аппаратный контроль?

31.4. Что такое псевдотерминал? Какие программы работают с псевдотерминалами?

31.5. Создайте в файле **/etc/inittab** записи, которые осуществляют следующее:

а) запускают программу **server-fallback** и дожидаются ее завершения, а затем немедленно останавливают работу системы, если поступает сигнал сбоя питания;

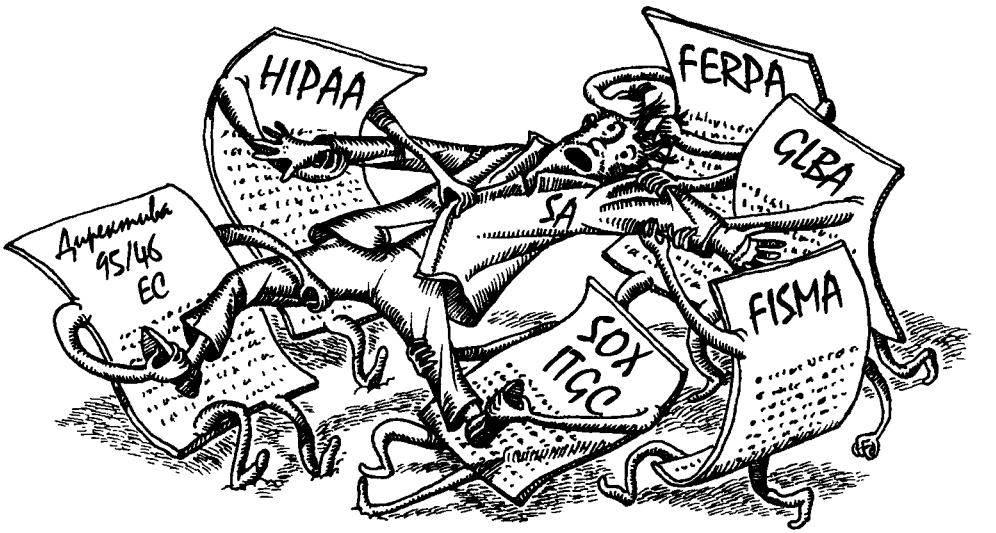
б) перезапускают серверную программу **unstable-srv** в случае сбоя;

в) запускают сценарий **clean-temp**, который удаляет все временные файлы при каждом перезапуске системы.

31.6. ✯ Ваш друг забыл выйти из Linux-системы, оставив ее включенной на ночь в лаборатории. На следующий день он столкнулся со странными проблемами при запуске программ в интерпретаторе команд. Программы внезапно завершались или зависали, а введенные символы исчезали при вводе определенных команд или символов. В то же время некоторые программы работали вполне корректно. Чем могут быть вызваны подобные аномалии? Как это проверить? Как устранить проблему? Кто мог так "пошутить"?

# Глава 32

# Управление, стратегия и политика




Можно иметь самую профессиональную команду администраторов в мире, но без соответствующего технического руководства не обойтись. Эта глава посвящена рассмотрению нетехнических аспектов создания успешного отдела информационных технологий, а также нескольких технических вопросов, затрагивающих связанную с управлением часть системного администрирования.

Большинство представленных в этой главе тем и идей не относится ни к какой определенной области. Они касаются как одного системного администратора, работающего на полставки, так большой группы профессионалов, работающих на полную ставку над каким-нибудь большим проектом. Они подобно зелени подходят каждому, какое бы "блюдо" ни готовилось.

Хорошие администраторы должны одинаково хорошо разбираться как в аппаратном, так и программном обеспечении. Способность организовать группу администраторов и принять меры, чтобы они полностью удовлетворяли потребности организации, отличает просто хорошего администратора от выдающегося.

Кроме советов, касающихся управления, глава содержит разделы по таким темам, как политика в области информационных технологий, эффективные методы работы и соответствие стандартам.

## 32.1. Цель информационных технологий

Отдел информационных технологий — это нечто большее, чем просто группа технических специалистов, умеющих ремонтировать принтеры и компьютеры. Со стратегической точки зрения, это коллектив людей, которые обслуживают организацию, поддер-

живая работу пользователей и системы. Никогда не следует забывать золотое правило администратора: производство должно управлять информационными технологиями, а не наоборот.

Для того чтобы достичь максимальной результативности, отдел информационных технологий должен взаимодействовать с другими подразделениями организации. В частности, необходимо согласовывать свои действия в области расходов, политики, управления и качества оказываемых услуг (service level agreements — SLA).

Во многих организациях — особенно в небольших компаниях и маленьких подразделениях внутри больших компаний — системный администратор совмещает несколько обязанностей, возможно, включая роль начальника отдела или менеджера. Понимание ключевых областей, в которых отдел информационных технологий взаимодействует с остальной частью организации, поможет сделать это взаимодействие намного эффективнее.

Перечислим минимальные обязанности отдела информационных технологий.

- Вести список нерешенных задач.
- Расставлять приоритеты задач и распределять ресурсы для их решения.
- Сообщать о состоянии задачи пользователям и сотрудникам предприятия.
- Взаимодействовать с сотрудниками предприятия, чтобы правильно удовлетворять их потребности.
- Осуществлять мониторинг компьютерного окружения, включая систему безопасности.
- Следить за появляющимися технологиями.
- Развивать навыки своих сотрудников.
- Способствовать соблюдению требований регулирующих органов.
- Разрабатывать инструкции по выполнению повторяющихся процессов и строго им следовать.
- Измерять прогресс в достижении поставленной цели.
- Быть готовым к катастрофам и иметь соответствующий план действий.
- Быть достаточно гибким, чтобы удовлетворять потребности пользователей, и достаточно дисциплинированным, чтобы удовлетворять потребности руководства.

## Составление сметы и расходование средств

Расходование средств отделом информационных технологий должно соответствовать целям всей организации. Бюджет отдела информационных технологий оказывает огромное влияние на диапазон и качество услуг, которые он может оказывать другим подразделениям организации, поэтому крайне важно, чтобы сотрудники этого отдела помогали всем понять эту связь и достигали необходимых компромиссов.

Доля расходов на информационные технологии в общем бюджете организации относительно невелика, хотя и представляет собой нетривиальный компонент сметы. Среднестатистическая организация тратит на них от 2 до 9% своего бюджета. Эта величина зависит от конкретной отрасти промышленности, но в среднем составляет примерно 4-5%.

Этот общий бюджет впоследствии подразделяется на капитальные затраты и эксплуатационные расходы. Капитальные затраты обычно связаны с приобретением оборудования. Эксплуатационные расходы включают стоимость труда и услуг, например

поддержание работы глобальной сети. Существует много способов конвертировать затраты одного типа в затраты другого. Например, аренда оборудования преобразует капитальные затраты в эксплуатационные расходы, а предварительно оплаченные контракты на техническое обслуживание позволяют капитализировать расходы по обслуживанию. Вероятно, вас не интересуют различия между этими видами расходов, а вот ваших бухгалтеров они очень даже интересуют, так что вам придется в этом разбираться.

Системные администраторы должны понимать, как составляется бюджет, потому что от этого зависит их способность планировать свою работу на год. Например, если администратор хочет реализовать и систему централизованной регистрации, и систему мониторинга безопасности, то его желания будут ограничены бюджетом. Если бюджетных средств хватает только на один сервер, то администратор должен либо изменить приоритеты проектов, либо принять решение, позволяющее запустить обе системы на одном сервере. (Отличным вариантом в этом случае станет виртуализация, но в других ситуациях совместное использование сервера осуществить не так легко и стоит дороже.) Если администратор может влиять на составление бюджета, он может способствовать выделению больших средств, если сможет обосновать, как информационная инфраструктура сможет повысить доходы компании.

## Стратегия в области информационных технологий

Стратегии в области информационных технологий влияют на работу всей организации, поэтому они являются важными компонентами общей стратегии компании. Основной вклад в разработку и реализацию стратегий в области информационных технологий вносят системные администраторы. Администраторы иногда несут прямую ответственность за стратегию развития; в других случаях их могут попросить дать отзыв на стратегии, разработанные другими членами организации.

В любом случае системные администраторы вносят ценный вклад в разработку общей стратегии организации. Многие организации имеют один набор стратегий для конечных пользователей и другой — для администраторов. Администраторы должны быть знакомы с обоими наборами стратегий и обязаны разрабатывать организационные мероприятия, направленные на их поддержку.

Ведение документации, тесно связанное с разработкой стратегии, иногда игнорируется или считается маловажным по сравнению с "реальной работой". Большинство системных администраторов не любят писать документы, хотя это важно для нормального функционирования информационной системы. Настройте систему wiki или используйте другие инструменты, с помощью которых администраторы могут делать короткие заметки, и помогите другим размещать релевантную информацию для последующего разбора или использования.

Для этой цели хорошо подходят пакеты MediaWiki и Confluence. Программа MediaWiki, основанная на Википедии, — это свободно распространяемый пакет, написанный на языке PHP (`mediawiki.org`). Программа Confluence — это промышленное коммерческое приложение, разработанное для средних и крупных предприятий. Вы можете инсталлировать его на своем сервере или взять в аренду (hosted solution), если не хотите управлять им локально (`atlassian.com`). "Список вики-приложений" на странице Википедии содержит много пунктов, а сделать правильный выбор на основе сравнительного анализа вам поможет сайт `wikimatrix.org`.

Конкретные стратегии и способы их согласованной реализации обсуждаются в разделе 32.7.

## Соглашения о качестве оказываемых услуг

Системное администрирование — это услуга, а получателями этой услуги являются люди и компьютеры. Для того чтобы отдел информационных технологий успешно справлялся со своими обязанностями, стараясь угождать пользователям и удовлетворяя потребности предприятия, все детали необходимо согласовать и зафиксировать в соглашении о качестве оказываемых услуг. Хорошее соглашение предусматривает соответствующий уровень обслуживания и является документом, на который можно ссылаться в проблемных ситуациях. (Однако помните, что отдел информационных услуг помогает, а не мешает пользователям!)

Пользователи довольны, если выполняются следующие условия.

- Их компьютеры в порядке, и они могут на них зарегистрироваться.
- Их другие ресурсы, такие как принтеры и файловые серверы, доступны.
- Их файлы с данными сохраняются, когда они их оставляют на время.
- Их прикладное программное обеспечение установлено и работает, как положено.
- При необходимости они получают дружественную и компетентную помощь.

Пользователи хотят, чтобы эти условия выполнялись 24 часа в сутки, 7 дней в неделю. Желательно бесплатно. Пользователи недовольны в следующих случаях.

- У них возникает простой, намеченный или незапланированный.
- Из-за модернизации происходят внезапные, несовместимые изменения.
- Они получают непонятные сообщения от системы или системных администраторов.
- Они получают долгие объяснения того, почему все это не работает.

Когда что-то выходит из строя, пользователи хотят знать, когда это будет исправлено. Их не интересует, какой жесткий диск или генератор сломались или почему; оставьте эту информацию для своих административных отчетов.

С точки зрения пользователя никакие новости не являются хорошими. Система или работает, или нет, и в последнем случае не имеет значения почему. Максимальное удовольствие наши клиенты получают, если они даже не замечают, что мы существуем! Грустно, но факт.

Не менее важно, чтобы были довольны и ваши сотрудники. Хороших администраторов трудно найти, и их потребности нужно учитывать при настройке системы управления вашей организацией. Системные администраторы и другой технический штат довольны, если выполняются следующие условия.

- Их компьютеры и системы поддержки пребывают в работоспособном состоянии.
- У них есть необходимые ресурсы, чтобы выполнять свою работу (двойные мониторы!).
- Они имеют новейшее и лучшее программное и аппаратное обеспечение.
- Они имеют стимул для работы или, по крайней мере, интересную работу (минимум тяжелой работы).
- Их никто постоянно не дергает.
- Они могут проявлять творческий подход без оглядки на босса, постоянно во все вмешивающегося и подсказывающего.
- Их часы работы и уровень стресса не выходят за пределы разумного.

Для того чтобы двигаться вперед, технические сотрудники нуждаются не только в зарплате в конце месяца. Они должны чувствовать, что у них есть степень творческого контроля над своей работой и что их ценят коллеги, босс и их пользователи.

Некоторые условия, при которых довольны и пользователи, и технический персонал, совпадают. Однако некоторые факторы кажутся несовместимыми или даже находятся в прямом противоречии. Босс должен удостовериться, что все эти противоречия можно сгладить или устранить.

Соглашение о качестве оказываемых услуг помогает помирить конечных пользователей и технический персонал. Хорошо написанное соглашение учитывает каждую проблему, упомянутую в следующих разделах.

### *Спектр услуг и их описание*

В соглашении о качестве оказываемых услуг описывается то, что организация может ожидать от отдела информационных технологий. Он должен быть написан в терминах, которые могут быть поняты нетехническими сотрудниками. Перечислим в качестве примера некоторые из этих услуг.

- Электронная почта.
- Интернет и веб-доступ.
- Файловые серверы.
- Бизнес-приложения.
- Печать.

В соглашении также должны быть определены стандарты, которых будет придерживаться отдел информационных технологий. Например, раздел о доступности услуг должен определять часы работы, согласованные окна обслуживания и ожидаемое время, в течение которого будет доступен технический персонал, чтобы обеспечить интерактивную поддержку. Одна организация могла бы решить, что регулярная поддержка должна быть доступна с 8:00 до 18:00 в будние дни, а чрезвычайная поддержка — круглосуточно. Другая организация могла бы решить, что нуждается в стандартной интерактивной поддержке, доступной всегда.

Представим список проблем, которые следует рассмотреть, документируя ваши стандарты.

- Время отклика.
- Обслуживание (и время отклика) в течение выходных и неурочных часов.
- Посещения на дому (поддержка для домашних компьютеров).
- Специальное (уникальное или патентованное) аппаратное обеспечение.
- Политика модернизации (устаревшие аппаратные средства, программное обеспечение и т.д.).
- Поддерживаемые операционные системы.
- Стандартные конфигурации.
- Истечение сроков действия резервных магнитных лент.
- Программное обеспечение специального назначения.
- Вспомогательная работа (чистка экранов и клавиатур, прочистка вентиляционных решеток).

Рассматривая стандарты услуг, имейте в виду, что многие пользователи захотят самостоятельно настроить свою среду (или даже свои системы), если не будет установлено программное обеспечение, чтобы это предотвратить. Стереотипный ответ на эти попытки — запретить всем пользователям осуществлять любые модификации. Это упрощает работу отдела информационных технологий, но не всегда является лучшей стратегией для всей организации.

Укажите эту проблему в своем соглашении о качестве оказываемых услуг и попытайтесь стандартизировать ее решение на нескольких определенных конфигурациях. Иначе ваше стремление к легкому обслуживанию и быстрому росту в рамках всей организации встретят серьезные препятствия. Поощряйте своих творческих сотрудников предлагать модификации, в которых они нуждаются, и будьте заботливы и щедры, учитывая их предложения в своих стандартных конфигурациях. Если вы не сделаете этого, то ваши пользователи будут упорно нарушать ваши правила.

### *Стратегии управления очередями*

Пользователи должны знать не только, какие услуги им будут оказаны, но и схему приоритетов, используемую для управления очередью заданий. Схемы приоритетов всегда оставляют возможность для маневра, но все же попытайтесь спроектировать такую схему, которая охватывала бы большинство ситуаций, почти или вовсе не прибегая к исключениям. Некоторые факторы, связанные с приоритетом, перечислены ниже.

- Важность услуги для всей организации.
- Влияние на безопасность (там было нарушение?).
- Оплаченный или оговоренный уровень сервисного обслуживания.
- Количество задействованных пользователей.
- Важность любого релевантного крайнего срока.
- Скандальность задействованных пользователей ("писклявые колеса").
- Важность задействованных пользователей (это дело тонкое, но будем честными: у некоторых людей в вашей организации есть больше авторитета, чем у других).

Хотя на ваше ранжирование будут влиять все эти факторы, мы рекомендуем подходить к исключениям с простым сводом правил и долей здравого смысла. В основном, мы используем следующие приоритеты.

- Много людей не могут работать успешно.
- Один человек не может работать успешно.
- Запросы на усовершенствования.

Если два или больше запросов имеют высший приоритет и они не могут выполняться параллельно, мы делаем выбор, основываясь на серьезности проблемы (например, неработающая электронная почта доставляет неудобства почти всем, тогда как временное отсутствие веб-службы могло бы помешать только нескольким людям). Очереди с более низкими приоритетами обычно обрабатываются по принципу FIFO.

Пользователи предполагают, что все их важные данные хранятся на резервных магнитных лентах, которые будут заархивированы навсегда. Однако резервные носители не хранятся неопределенно долго; магнитные носители имеют конечный срок годности, после наступления которого чтение данных затрудняется. (Вы должны периодически перезаписывать свои данные, возможно, на более новые носители, если хотите хранить их в течение долгого времени.) Резервные магнитные ленты могут также быть затребова-

ны в суд. Таким образом, ваша организация, возможно, не захочет, чтобы старые данные остались доступными навсегда. В этом случае лучше всего совместно с руководителями, отвечающими за такие решения, составить письменное соглашение, которое определяет, сколько времени резервные копии должны храниться, следует ли их обновлять (обязательно? возможно? никогда?) и где эти копии должны храниться.

Эти решения должны быть приняты в контексте политики хранения данных всей организации. Такой тип стратегий описан в этой главе позже, но, вообще, вы должны классифицировать свои данные и разработать график хранения для каждого класса.

Доведите стратегию резервирования и хранения данных до ваших пользователей. Эта мера будет способствовать реалистическим ожиданиям и относительно резервных копий, и относительно возможности их восстановления. Она также позволит уведомить пользователей, что они должны принять собственные меры предосторожности, если считают, что нуждаются в лучшей защите данных, чем предусмотрено в их соглашении о качестве оказываемых услуг.

В частности, пользователи должны понять, будут ли сохраняться файлы на их местных автоматизированных рабочих местах. Многие организации поддерживают свои централизованные файловые серверы, но не отдельные автоматизированные рабочие места. Обычно автоматизированные рабочие места клонируются на основе системных образов и считаются доступными. Пользователи должны знать об этом, чтобы правильно хранить важную информацию.

### Роли и обязанности

Вы должны зафиксировать в документе, кто и за что отвечает. Организации, которые не делят обязанности, становятся непроизводительными и неэффективными. Задачи остаются "беспризорными", потому что не ясно, где заканчивается одна область компетенции и начинается другая. Задачи могут также стать жертвой группового мышления, где одну задачу решают два или три администратора. Перечислим примеры определенных ролей.

- Администратор резервного копирования
- Знаток сетей хранения и файловых серверов
- Прикладной спорщик
- Царь безопасности и заплат
- Парень, который никогда не уверен в том, что он делает[1]

В качестве альтернативы вы могли бы распределять роли и обязанности согласно описаниям услуг, которые вы уже определили. Этот подход может подразумевать, что вы должны очертить обязанности с точки зрения администрации, а не с точки зрения пользователя.

Не забывайте включать обязанности "дублера" в свою таксономию. Сотрудники не будут присутствовать в офисе каждый день, и вы должны знать, кто куда идет, когда основной администратор домена отсутствует.

### Показатели соответствия

Соглашение о качестве оказываемых услуг должно определить, как организация измеряет ваш успех при выполнении условий соглашения. Цели и ориентиры позволяют

[1] Ну хорошо, возможно, в вашем коллективе эта роль не нужна. Но она предусмотрена промышленным стандартом.

работникам совместно стремиться к общему результату и могут заложить основу для сотрудничества во всей организации. Конечно, вы должны удостовериться, что у вас есть инструменты, позволяющие измерить согласованные показатели.

Как минимум, вы должны отследить следующие показатели вашей информационной инфраструктуры.

- Процент или количество проектов, законченных вовремя и в пределах бюджета.
- Процент или количество выполненных пунктов соглашения о качестве оказываемых услуг.
- Процент продолжительности работы системы (например, электронная почта доступна через службу Q1 в течение 99,92% времени).
- Процент или число билетов, которые были удовлетворительно распознаны.
- Среднее время распознания билета.
- Процент или количество нарушений техники безопасности, зарегистрированных согласно установленной процедуре.

## 32.2. Информационная структура организации

Теперь, когда мы рассмотрели общую схему функционирования отдела информационных технологий, остановимся на структуре организации. По мере развития организации становится очевидным, что не каждый член группы может или должен знать все о ее инфраструктуре. Вместо этого необходимо найти баланс между эффективностью и разделением обязанностей.

Ролевое разделение добавляет уровень сдержек и противовесов к структуре организации. Со временем эта особенность становится более важной, а стандарты и инструкции внедряются в даже самую маленькую организацию.

Рассмотрим, например, американскую компанию с 20 сотрудниками, которая разработала дистанционное приложение для медицинских учреждений. Если это приложение хранит какую-либо защищенную медицинскую информацию (protected health information — PHI), то все системы организации должны соответствовать суровому Закону об отчетности и безопасности медицинского страхования (Health Insurance Portability and Accountability — HIPAA). Между прочим, это законодательство требует, чтобы вы определили роли, чтобы защитить доступ к конфиденциальным данным. Например, определением уровня доступа, который должен иметь пользователь, и фактическим обеспечением этого доступа обязаны заниматься два разных человека.

В центре типичной информационной структуры организации лежит система отслеживания запросов (ticketing system), а также сервисная служба, группа архитектуры предприятия, оперативная группа и уровень управления. Как показано на рис. 32.1, каждая часть организации взаимодействует с системой отслеживания запросов.

### Основы: система отслеживания запросов и управления задачами

Система отслеживания запросов и управления задачами лежит в основе функционирования каждого отдела информационных технологий. Наличие хорошей системы поможет вашим сотрудникам избежать двух наиболее распространенных ловушек технологического процесса, перечисленных ниже.


*Рис. 32.1. Типичная информационная структура организации*

- "Беспризорные" задачи, о которых все думают, что о них заботится кто-то другой.
- Ресурсы, растраченные впустую из-за дублирования усилий, когда несколько людей или групп работают над той же самой проблемой без координации.

## Общие функции систем отслеживания запросов

Система отслеживания запросов на устранение неисправностей принимает заявки через различные интерфейсы (электронная почта, веб-формы и командные строки, являющиеся наиболее распространенными инструментами) и отслеживает их от момента подачи до решения проблемы. Менеджеры могут пересылать заявки специальным группам или отдельным сотрудникам. Штат может запросить у системы, чтобы видеть очередь поданных заявок и, возможно, удовлетворить некоторых из них. Пользователи могут узнать статус запроса и видеть, кто работает над ним. Менеджеры могут извлечь следующую информацию высокого уровня.

- Количество поданных заявок
- Среднее время удовлетворения заявки
- Производительность системных администраторов
- Процент неудовлетворенных заявок
- Распределение рабочей нагрузки во время поиска решения

История заявки, сохраненная в системе отслеживания запросов, становится историей проблем, существующих в вашей информационной инфраструктуре, и их решений. Если эта история легко доступна для поиска, то это становится неоценимым ресурсом для системного администратора.

Сообщения о решенных проблемах можно послать начинающему системному администратору и стажерам, поместить в систему часто задаваемых вопросов или просто зарегистрировать. Новые сотрудники могут извлечь выгоду из получения копий удовлетворенных запросов, потому что эти запросы содержат не только техническую информацию, но и примеры тона и коммуникационного стиля, которые являются подходящими для общения с клиентами.

Как и все документы, исторические данные системы отслеживания запросов могут потенциально использоваться против вашей организации в суде. Следуйте инструкциям по хранению документов, установленным вашим юридическим отделом.

Большинство систем, отслеживающих запросы, автоматически подтверждают получение новых запросов и назначают им номер, который сотрудник, подавший запрос, может использовать, чтобы отследить его или его статус. В ответе, который отсылается автоматически, должно быть ясно указано, что это — только подтверждение. Оно должно сопровождаться немедленным сообщением от живого человека, который объяснит план устранения проблемы или обработки запроса.

## Владение запросом

Работу можно разделить, но, как показывает опыт, ответственность меньше поддается распределению. Каждая задача должна иметь одного владельца. Этот человек не должен быть наблюдателем или менеджером, желающим действовать только как координатор. Этот кто-то должен быть готов сказать: "Я беру на себя ответственность за решение задачи".

Важный побочный эффект этого подхода состоит в том, что он позволяет выяснить, кто что сделал и кто какие изменения внес. Эта прозрачность становится важной, если вы хотите выяснить, почему что-то было сделано именно так или почему что-то работает не так или не работает вообще.

Быть "ответственным" за задачу — не значит быть козлом отпущения, если возникают проблемы. Если ваша организация определяет ответственность как "наказуемость", то вы можете обнаружить, что количество доступных владельцев проектов быстро истощилось. Ваша цель в назначении ответственности состоит в том, чтобы просто устранить двусмысленность в отношении того, кто должен решать каждую проблему. Не наказывайте сотрудников за то, что они попросили помощи.

С точки зрения клиента, хорошая система назначения — та, в которой проблемы попадают к компетентному человеку, способному решить проблемы быстро и полностью. Но с организаторской точки зрения, назначения должны быть стимулом для сотрудников, чтобы они продолжали совершенствоваться. Ваша работа состоит в том, чтобы уравновесить возможности сотрудников и стимулировать их, одновременно удовлетворяя все желания клиентов.

Большие задачи могут быть разными, включая полноценные проекты по разработке программного обеспечения. Эти задачи могут потребовать использования методов формального управления проектом и инструментов программирования. Мы не описываем эти инструменты здесь; однако они важны и не должны быть пропущены.

Иногда системные администраторы знают, что конкретная задача должна быть сделана, но они не делают этого, потому что эта задача неприятна. Системный администратор, указавший на заброшенную, беспризорную или непопулярную задачу, вероятно, и получит ее "в награду". Эта ситуация создает конфликт интересов, потому что заставляет системных администраторов замалчивать такие ситуации. Не позволяйте этому происходить в вашей организации; дайте вашему системному администратору свободу безбоязненно указывать на проблемы. Вы можете позволить им поднимать вопросы, не назначая владельца или привязывая их непосредственно к проблеме. Кроме того, вы можете создать почтовый псевдоним, на который можно послать сообщения о проблемах.

## Система отслеживания запросов с точки зрения пользователей

Получение быстрого ответа от человека является критерием удовлетворенности потребителя, даже если личный ответ содержит не больше информации, чем автоматизированный. При возникновении большинства проблем человеку, подавшему заявку, более важно знать, что она была рассмотрена человеком и что это поможет немедленно решить проблему. Пользователи понимают, что администраторы получают много запросов, и они готовы ждать вашего внимания разумно долгое время. Но они не желают, чтобы их игнорировали.

Механизм, посредством которого пользователи представляют запросы, затрагивает их восприятие системы. Удостоверьтесь, что вы понимаете культуру своей организации и предпочтения своих пользователей. Они хотят использовать веб-интерфейс? Прикладную программу? Почтовый псевдоним? Возможно, они хотят делать только телефонные звонки!

Также важно, чтобы администраторы не торопились и удостоверились, что они понимают то, о чем фактически просят пользователи. Это кажется очевидным, но вспомните последние пять писем, которые вы послали по электронной почте псевдониму технической поддержки или обслуживания клиентов. Держим пари, что было, по крайней мере, несколько случаев, когда ответ не имел никакого отношения к вопросу — не потому, что те компании не были компетентны, а потому, что точный анализ заявок на устранение неисправности более труден, чем это кажется.

Как только вы прочитали первую часть запроса и поняли, о чем спрашивает клиент, остальная часть запроса часто перестает вас интересовать и заменяется фразой "и тому подобное". Боритесь с этим! Клиенты очень не любят, когда их запрос был неправильно понят и должен быть повторно представлен или вновь заявлен. Перечитайте его еще раз.

Запросы часто бывают неопределенными или неточными, потому что сотрудники, которые их подают, не обязаны иметь техническую подготовку, чтобы описывать проблему так, как хочет системный администратор. Как бы то ни было, это не мешает пользователям высказывать их собственные предположения относительно того, что случилось. Иногда эти предположения совершенно правильные, а иногда вы должны сначала расшифровать запрос, чтобы определить то, что пользователь думает о проблеме, а затем проследить ход его мыслей, чтобы интуитивно постигнуть основную проблему.

## Типичные диагностические системы

В табл. 32.1 и 32.2 перечислены характеристики нескольких известных диагностических систем. В табл. 32.1 содержатся характеристики общедоступных систем, а в табл. 32.2 представлены коммерческие системы.

**Таблица 32.1. Открытые диагностические системы**

| Название | Ввод* | Язык | Прикладная часть* | Веб-сайт |
|---|---|---|---|---|
| Double Choco Latte | W | PHP | PM | dcl/sourceforge.net |
| Mantis | WE | PHP | M | mantist.org |
| OTRS | WE | Perl | PMOD | otrs.org |
| RT Request Tracker | WE | Perl | M | bestpractical.com |

*Окончание табл. 32.1*

| Название | Ввод[а] | Язык | Прикладная часть[б] | Веб-сайт |
|---|---|---|---|---|
| Scarab | WE | Java | M | scarab.tigris.org |
| Trouble Ticket Express | WE[в] | Perl | FM[в] | troubleticketexpress.com |

[а] Типы ввода: W — веб, E — электронная почта.

[б] Прикладная часть: M — MySQL, P — PostgreSQL, O — Oracle, D — DB2, F — простые файлы.

[в] Для использования электронной почты и MySQL необходимо купить модули надстроек (правда, они недорогие).

**Таблица 32.2. Коммерческие диагностические системы**

| Название | Масштаб | Веб-сайт |
|---|---|---|
| EMC Ionix (Infra) | Огромная | infra-corp.com/solutions |
| HEAT | Средняя | frontrange.com |
| Remedy (сейчас BMC) | Огромная | remedy.com |
| ServiceDesk | Огромная | ca.com/us/service-desk.aspx |
| Track-It! | Средняя | numarasoftware.com |

Нам очень нравится система Mantis. Первоначально она была разработана для отслеживания неполадок в программном обеспечении видеоигр. Она запускается на Linux, Solaris, Windows, Mac OS и даже OS/2, занимает мало места, проста, легко изменяется и конфигурируется. Требует наличия PHP, MySQL и веб-сервера. Но ее самым важным качеством является хорошая документация!

OTRS (Open Ticket Request System) — тоже неплохая система. В ней доступны интерфейсы как для клиентов, так и для системных администраторов, а также интерфейс для электронной почты. OTRS позволяет настраивать очень много параметров (например, она позволяет по очереди настраивать приветственные сообщения) и умеет даже фиксировать время, затраченное на обработку сообщений о неполадках.

В табл. 31.2 представлено несколько коммерческих программ для управления запросами. Их язык реализации и базы данных здесь не указаны, потому что эта информация в полном объеме доступна на посвященных им веб-сайтах.

Некоторые из этих коммерческих систем настолько сложны, что для их сопровождения, настройки и поддержания в рабочем состоянии нужно нанимать отдельного человека или даже двух (для тех, кто не знает, сообщаем, что в данном случае речь идет о системах Remedy и ServiceDesk). Такие системы подходят для организаций с большим штатом IT-сотрудников и являются неудачным выбором при наличии типичной, небольшой команды IT-специалистов, которые и так постоянно завалены работой.

## Распределение сообщений о неполадках

В большой организации, даже в той, где установлена замечательная диагностическая система, все равно необходимо решить еще одну проблему. Не очень хорошо, когда системным администраторам приходится разрываться между задачей, над которой они сейчас работают, и очередью запросов, особенно если запросы поступают по электронной почте прямо в их личный почтовый ящик. Мы применили два подхода для решения этой проблемы.

Вначале мы организовали для наших системных администраторов смены по полдня. Каждый во время своей смены должен был стараться ответить на как можно большее

количество поступивших запросов. Проблемой этого подхода было то, что не у всех системных администраторов были навыки, необходимые для ответа на все вопросы и устранения всех проблем. Иногда ответы были некорректными из-за того, что заступивший на смену человек был новичком и еще плохо знал клиентов, их среды или предусмотренные в их контракте услуги поддержки. В результате старшим администраторам все равно приходилось присматривать за всеми остальными, из-за чего они, естественно, не могли полностью сконцентрироваться на своей работе. В конечном итоге качество обслуживания только ухудшилось, и никакой пользы этот подход не принес.

После такого неудачного опыта мы придумали роль "диспетчера", которая ежемесячно переходит от одного старшего администратора к другому. Диспетчер отвечает за проверку системы отслеживания запросов на наличие новых записей и распределяет задачи между конкретными сотрудниками. При необходимости диспетчер связывается с пользователями и узнает у них дополнительные сведения, необходимые для определения степени важности запросов. Диспетчер пользуется специальной базой данных о навыках сотрудников, созданной в организации своими силами, для принятия решений о том, у кого в группе поддержке имеются необходимые навыки и время. Кроме того, диспетчер также следит за тем, чтобы запросы обрабатывались своевременно.

## Перечень навыков

Никто не раздражает опытного пользователя больше, чем человек из команды поддержки, который, пока на самом деле просто пытается отыскать в базе данных информацию о клиенте, спрашивает: "А вы подключили кабель питания?". С другой стороны, заставлять опытного администратора объяснять пользователю-новичку, как отыскать клавишу <Del> в какой-нибудь системе обработки текстов, — это тоже просто неэффективное использование ресурсов.

У нас каждый сотрудник обязательно должен отработать в службе поддержки определенное количество часов в неделю. Сотрудники отдела поддержки включают это время в свой еженедельный план работ и затем вызывают человека, когда для него появляется задание. Задания распределяются в соответствии с навыками, требующимися для устранения проблемы, и количеством времени, оставшимся у каждого в еженедельном резерве отдела поддержки.

Для того чтобы эта схема работала успешно, необходим сбалансированный перечень навыков сотрудников. На протяжении определенного времени каждый сотрудник должен отработать свое количество часов в службе поддержки. В целом, сотрудник, имеющий множество записей в списке навыков, является более "ценным". Однако в сотруднике с меньшим количеством навыков тоже нет ничего плохого, если для всех хватает работы.

Точный список навыков помогает определить, достаточно ли в организации сотрудников с определенными навыками, чтобы можно было не волноваться, если кто-то из них уйдет на больничный или в отпуск. Список навыков можно составлять постепенно, по мере появления проблем и их решения различными сотрудниками, включая в него наименование проблемы, имя занимавшегося ею сотрудника и продемонстрированный им при ее решении уровень мастерства.

Навыки следует описывать с определенной степенью точности, т.е. и не слишком конкретно и не слишком обще. Приведенный ниже в качестве примера список демонстрирует наиболее оптимальный вариант перечисления навыков.

- Создание и удаление пользователей, установка паролей, изменение квот.
- Создание учетных записей CVS или SVN.

- Интегрирование новых драйверов оборудования в RIS (Windows).
- Упаковка Windows-приложения в формате MSI.
- Создание и установка пакетов программ в Linux.
- Анализ журнальных файлов.
- Устранение неполадок, связанных с почтовым сервером.
- Отладка проблем, связанных с печатью.
- Устранение общих неполадок с оборудованием.
- Внесение записей в службу DNS.
- Решение вопросов, касающихся лицензий на программное обеспечение.
- Умение отвечать на вопросы, касающиеся безопасности в системах Windows.
- Умение отвечать на вопросы, касающиеся безопасности в системах UNIX.
- Разрешение запросов по пакету Samba.
- Конфигурирование сервера DHCP.
- Настройка сервера LDAP.
- Добавление и удаление веб-сайтов (конфигурирование сервера Apache).

## Управление временем

Системное администрирование подразумевает такое количество переключений с одного контекста на другой за день, которое нигде не встретишь и за целый год, и, в основном, с этим хаосом приходится справляться именно сотрудникам отдела поддержки пользователей. Каждый администратор должен обладать навыками распределения времени. Без них он не сможет справляться со своими повседневными обязанностями и, следовательно, станет раздражительным или вообще впадет в депрессию. (Если же он уже находится в раздраженном или депрессивном состоянии, ему станет еще хуже.)

Системные администраторы "изнашиваются" очень быстро. Многих из них хватает всего лишь на несколько лет. Никто не хочет, чтобы его все время вызывали и постоянно вычитывали. Умение эффективно распределять время и успевать обслуживать пользователей так, чтобы они оставались довольны, является очень выигрышным качеством.

# 32.3. Служба поддержки

Служба поддержки — главная составляющая структуры отдела информационных технологий, представленной на рис. 32.1. Задача службы поддержки — взаимодействие с людьми, которые используют компьютерные системы, которые вы обслуживаете.

## Диапазон услуг

Сотрудники службы поддержки выполняют те разделы соглашения о качестве оказываемых услуг, в которых определено, какие виды прямой помощи человек в пределах организации может получить. Проблемы, решаемые службой поддержки, включают вопросы эксплуатации настольных систем, сопровождение приложений и такие первоочередные проблемы системного администрирования, как отключение электропитания сервера, сбой в сети и восстановление файлов.

Кроме обычных услуг, предоставляемых диагностической системой или системой экстренной поддержки, это подразделение может также предложить вспомогательные

услуги, такие как учебные семинары. Эти меры помогают увеличить самостоятельность клиентов и сокращают количество запросов поддержки.

Кроме того, важно сформулировать правила эскалации. Служащие должны знать, что сделать, когда их потребности не удовлетворяются или когда они хотят выразить благодарность за хорошо сделанную работу.

### Доступность службы поддержки

Хороший отдел информационных технологий состоит из квалифицированных сотрудников, всегда готовых прийти на помощь клиентам.

Большинство проблем незначительны и могут безопасно включаться в очередь на обслуживание. Другие проблемы приводят к простою и требуют пристального внимания. Автоматизированные ответы от системы отслеживания запросов и записанные телефонные сообщения, объявляющие о рабочем расписании, только вызывают раздражение. Обеспечьте пользователям доступ к последней инстанции, если возникает такая необходимость. Как правило, для этого достаточно мобильного телефона, которым обмениваются системные администраторы во внеурочное время.

### Зависимость от службы поддержки

К сожалению, превосходная поддержка иногда порождает зависимость. Пользователи могут легко привыкнуть консультироваться со службой поддержки, даже когда это не является необходимым. Если вы понимаете, что кто-то использует систему поддержки для ответов, которые они легко могли найти в справочнике или в системе Google, то можете попытаться ответить на такие вопросы, указывая соответствующую справочную страницу или идентификатор URL. Однако будьте осторожны: эта тактика может действительно возмутить пользователей, если не проявить при этом предельное уважение.

## 32.4. Архитектура предприятия

Еще одна информационная составляющая (рис. 32.1) — архитектура предприятия — состоит из администраторов, имеющих полное техническое представление об организации. Эта роль почти всегда подразумевает некоторое количество администраторов систем UNIX или Linux. Эти люди рассматривают как непосредственное, так и долгосрочное влияние новых систем на общую инфраструктуру. Они понимают, как организация будет развиваться в ближайшие годы и как сегодняшние требования станут основой для будущих модификаций.

Архитекторы предприятия также ответственны за понимание принципов взаимодействия систем. Например, в организации, которая хранит конфиденциальную информацию о клиентах, архитекторы должны понимать, как возможность шифрования базы данных будет воздействовать на конечных пользователей, и определять, является ли это воздействие приемлемым.

Следующие разделы представляют собой перечень наилучших архитектурных решений, которыми следует пользоваться при планировании структуры отдела информационных технологий в своей организации. Эти принципы особенно важны, когда конфигурация, которая будет поддерживаться, является новой или необычной, потому что примеры таких ситуаций очень сложно отыскать в реальном мире. Все хорошо спроектированные процессы подразумевают соблюдение этих или близких к ним принципов.

## Процессы должны быть воспроизводимыми

Системное администрирование — это не показательное выступление. Все, что делается, должно делаться последовательно и многократно. Обычно это означает, что низкоуровневые изменения должны вноситься сценариями или конфигурационными программами, а не самими системными администраторами. Расхождения в конфигурации должны перехватываться и фиксироваться в конфигурационных файлах соответствующего, предназначенного для администрирования программного обеспечения.

Например, сценарий, настраивающий новый компьютер, не должен задавать вопросов о номерах IP-адресов и пакетах, которые нужно установить. Вместо этого, он должен проверять каталог конфигурации системы и извлекать всю необходимую информацию оттуда. Он может отображать эту информацию для подтверждения, но выбираемые параметры должны быть предопределены. Чем меньше будет участвовать в этом процессе пользователь, тем меньше шансов, что он допустит какую-нибудь ошибку.

Но давайте уточним: мы не описываем здесь организацию, где все стратегические решения принимаются администраторами "высшего ранга" и беспрекословно выполняются глупыми трутнями. Возможность воспроизведения не менее важна и тогда, когда вы являетесь единственным администратором в своей организации. Незапланированных конфигурационных решений, не оставляющих никаких данных для аудита, лучше не принимать. Если требуется что-то изменить, изменяйте центральную конфигурационную информацию и распространяйте изменения уже оттуда.

## Сохранение отладочных данных

Кто сделал, что сделал и зачем? В случае появления проблем в системе, наличие возможности вернуться к последнему рабочему состоянию или, по крайней мере, выяснить, что изменилось с тех пор, позволит устранить их быстрее. Важно знать не только "что" изменилось, но и "кто" внес это изменение и "почему". Беседа с человеком, реализовавшим приведшее к возникновению проблем изменение, часто позволяет увидеть суть. Возможно, изменение удастся быстро отменить, но иногда бывает так, что изменение было внесено в благих целях и его отмена может только ухудшить дело.

Системы контроля изменений являются одним из наиболее удобных способов их отслеживания. Они предоставляют как хронологический перечень фактических данных, так и информацию о том, каким из системных администраторов было внесено то или иное изменение. При правильном использовании каждое изменение сопровождается комментарием, в котором объясняется причина его внесения.

Автоматизированные средства могут проверять конфигурационные файлы, которые они изменяют, и указывать на себя в этих комментариях. Благодаря этому выявить неправильно работающий сценарий и отменить внесенные им изменения совсем не трудно.

🕮 Системы контроля изменений описаны в разделе 12.9.

Если ваша организация использует систему отслеживания запросов, в ней также удобно хранить изменения. Для каждого изменения можно создать запрос, в который включается информация о том, кто, что, когда, где и почему сделал. Кроме того, не менее важно то, что этот запрос может содержать план возврата. Иначе говоря, если однажды ночью что-то пойдет не так, то дежурному администратору будет не обязательно будить системного администратора.

И вы, и ваши сотрудники должны дисциплинированно обращаться с каждым запросом на изменение. Системы отслеживания запросов приносят пользу только при условии, что их используют все администраторы при каждом изменении.

## Осознание важности документации

На самом деле документация настолько важна для масштабируемой инфраструктуры, что мы посвятили ей целый подраздел в разделе 32.5.

## Настройка и кодирование

Использовать существующие средства — это хороший подход, и так следует поступать всегда, когда это возможно. Но двух абсолютно одинаковых организаций не бывает, и каждая организация обязательно имеет какие-нибудь уникальные требования. Информационная инфраструктура, которая в точности удовлетворяет требования организации, увеличивает конкурентоспособность этой организации в целом и продуктивность ее сотрудников в частности.

Благодаря гибкости в плане написания сценариев и обилию средств с открытым исходным кодом, система UNIX является идеальной базой для создания хорошо отрегулированной инфраструктуры. С нашей точки зрения, группа системных администраторов, не умеющая выполнять такую функцию, как разработка программного обеспечения, является слабой.

## Содержание системы в чистоте

Системное управление заключается не только в установке, добавлении и конфигурировании программного и аппаратного обеспечения; оно также предполагает наличие знаний о том, что следует сохранить, выкинуть и обновить. Мы называем эту концепцию "устойчивым управлением". Иметь возможность добавить новый компьютер в среду за пять минут и создать новую учетную запись пользователя за десять секунд — это замечательно. Но если заглянуть в будущее, станет ясно, что наличие возможности отыскивать и удалять старые учетные записи и компьютеры организованным образом является не менее важным. Устойчивость в системном управлении означает, что у вас есть концепции и средства, необходимые для того, чтобы вы могли выполнять свою работу организованным образом в течение длительного срока.

# 32.5. Операции

В заключение рассмотрим последнюю роль в этой главе — операции. С деловой точки зрения, под операциями подразумеваются "ежедневные рутинные действия, которые и представляют собой цель бизнеса". С технической точки зрения, операции — это источник проблем, которые должен решать системный администратор. В качестве примера операций назовем возврат, мониторинг, установку заплат, обновление, инсталляцию нового программного обеспечения и отладку.

Оперативная группа отвечает за установку и сопровождение информационной инфраструктуры. Как правило, оперативная группа имеет дело с компьютерами и проводами, а справочная служба — с людьми.

Оперативная группа фокусируется на создании стабильной и заслуживающей доверия среды для клиентов. Доступность и надежность являются ее ключевыми задачами. Сотрудники этой группы не должны ни проводить никаких экспериментов, ни делать никаких неожиданных исправлений или улучшений по пятницам. Просто вероятность того, что это закончится появлением проблем (заметить которые на выходных смогут только клиенты), слишком высока.

## Время простоев должно быть минимальным

Многие люди зависят от предоставляемой нами вычислительной инфраструктуры. Внутреннее подразделение, пожалуй, сможет какое-то время обходится без своего вебсайта, а вот компания, принимающая заказы по почте через Интернет, такая как компания Amazon.com, — нет. Некоторые даже не заметят, что сервер печати не работает, но сотрудника, которого поджимают сроки сдачи документа, это действительно очень расстроит. В большинстве организаций потеря доступа к электронной почте обычно всех выводит из себя. Центральные файловые серверы тоже являются потенциальным источником неприятностей.

В некоторых организациях придется обеспечивать наличие "аварийной службы". В коммерческой среде это может означать круглосуточное присутствие на месте опытных системных администраторов.

Даже если бюджет не позволяет явно обеспечить круглосуточное дежурство, нужно быть готовым к вызову любых администраторов, которые окажутся доступны в позднее время суток или в выходной день. Дежурный мобильный телефон или другая система уведомления часто являются "довольно хорошим" средством. Проверьте, чтобы пользователи могли получать доступ к этому средству каким-нибудь простым и хорошо известным способом; например, создайте адрес-псевдоним и сделайте так, чтобы все поступающие на этот адрес SMS-сообщения автоматически и немедленно перенаправлялись на дежурный мобильный телефон.

## Документирование зависимостей

Для того чтобы делать точные предположения о доступности оборудования или периоде безотказной работы, нужно знать не только о своих сильных и слабых местах (степени надежности устанавливаемого оборудования), но и о зависимостях информационных систем от другого оборудования, программного обеспечения и персонала.

- **Питание**. Независимые источники и схемы питания, защита от перепадов напряжения, резервные системы питания, такие как генераторы и источники бесперебойного питания, разводка кабелей электропитания в здании, карты электропитания определенных компонентов оборудования.
- **Сеть**. Проводка в здании, резервные линии связи, служба работы с клиентами для интернет-провайдеров, топология сети, контактная информация для других отделов в организации, выполняющих свою собственную роль в сетевом управлении.
- **Оборудование**. Системы высокой степени готовности и процедуры для их использования, горячие и холодные резервы, запасные детали, договоры на техническое обслуживание оборудования.

## Переналадка или списывание старого оборудования

Для того чтобы поддерживать свою инфраструктуру, нужно покупать новые компьютеры, переналаживать устаревшие и выкидывать совсем старые. О закупке оборудования мы расскажем позднее, но отметим, что многие сотрудники не желают расставаться со своими "любимцами".

Поскольку пользователям и менеджерам часто не очень хочется обновлять устаревшее оборудование, вы иногда должны брать инициативу в свои руки. Финансовая информация является самым убедительным доказательством. Если вам удастся на бумаге

продемонстрировать, что стоимость обслуживания старого оборудования превышает стоимость средств, требующихся на его замену, многие возражения по поводу обновления отпадут сразу. Иногда разнородное оборудование полезно заменить хотя бы для того, чтобы сэкономить время и усилия, требующиеся на поддержание различных версий операционных систем и программного обеспечения на уровне современных требований.

Недорогое оборудование Intel/PC является стандартной архитектурной базой настольных систем, особенно теперь, когда продукция компании Apple поставляется на оборудовании компании Intel. Распространенность персональных компьютеров за последние годы сместила затраты со стороны оборудования в сторону программного обеспечения и его поддержки.

Для того чтобы поддерживать актуальную инфраструктуру, необходимо действовать на опережение. Разработайте правила, которые регламентируют срок использования разных систем. Например, ноутбуки следует использовать не более трех лет, настольные системы — четыре года, а серверы — пять лет. Эти показатели могут зависеть от производителей и контракта на поддержку.

Планирование замены старого оборудования экономит время и снижает вероятность сбоев. Если ноутбуки меняются каждые три года, то вряд ли вы посреди ночи получите сообщение, что ваш исполнительный директор, находящийся в командировке, не может получить электронные сообщения, потому что его ноутбук вышел из строя.

## Поддержка локальной документации

Точно так же как большинство людей верят в то, что физические упражнения и свежие овощи благотворно влияют на организм, каждый ценит хорошую документацию и знает о ее важности. К сожалению, это вовсе не означает, что она будет составляться и обновляться ими без напоминаний.

Действительно, почему мы должны волноваться о документации?

- Документация снижает вероятность одиночных сбоев. Замечательно, когда есть программы, которые позволяют мгновенно разворачивать рабочие станции и распространять заплаты и обновления при помощи одной-единственной команды, но они становятся практически бесполезными, если по ним нет никакой документации, а умеющий с ними обращаться специалист ушел в отпуск или вообще уволился.
- Документация способствует воспроизводимости. Если правила и процедуры не хранятся в документах организации, ими вряд ли будут пользоваться регулярно. Когда администраторы не могут отыскать информацию о том, как следует поступать в том или ином случае, им приходится импровизировать.
- Документация экономит время. Когда пишешь документацию, вовсе не кажется, что ты экономишь время, но, потратив несколько дней на решение проблемы, которая когда-то раньше решалась, но как именно уже никто не помнит, большинство администраторов соглашаются, что написание документации более чем оправдывает себя.
- И, наконец, самое главное: документация делает систему более понятной и позволяет вносить последующие изменения с учетом архитектуры системы. Когда изменения вносятся на основе только частичных знаний, они часто не совсем вписываются в имеющуюся архитектуру. Со временем степень несоответствия увеличивается, и система даже работающим с ней администраторам начинается казаться беспорядочной коллекцией компонентов. В конечном итоге часто возникает желание удалить все компоненты и начать все заново.

Локальная документация удобна во многих ситуациях. Доводилось ли вам заходить в машинный зал для выполнения перезагрузки одного сервера и сталкиваться с рядами стоек с похожей и в то же время разной аппаратурой, никак не помеченной? Приходилось ли вам устанавливать устройство, с которым вы уже имели дело когда-то давно, но сейчас помните только то, что настраивать его было очень трудно?

Локальная документация должна храниться в четко установленном месте. В зависимости от объема выполняемых операций, это может быть либо каталог на файловом сервере, смонтированный на всех компьютерах, либо даже домашний каталог, принадлежащий какой-нибудь специальной учетной записи в системе.

Убедив системных администраторов описывать в документах конфигурации и действия, важно защитить саму документацию. Злонамеренный пользователь может принести много вреда, исказив документацию организации. Примите меры, чтобы сотрудники, которым нужна документация, могли найти и прочитать ее (организуйте ее поиск) и чтобы каждый, кто поддерживает эту документацию, мог изменить ее. Однако следует уравновешивать доступ и защиту.

Документация, основанная на системе Wiki, особенно хороша, если вам необходимо легко устранить злонамеренные искажения. Другие системы можно защитить аналогичным образом с помощью системы контроля вариантов.

### *Стандартизированная документация*

Как показывает наш опыт, самым простым и эффективным способом ведения документации является стандартизация небольших, простых документов. Вместо того чтобы писать целое руководство по системному управлению для своей организации, лучше написать ряд одностраничных документов, посвятив каждый из них какой-нибудь одной конкретной теме. Начните с общей темы, а затем разбейте ее на подтемы, содержащие дополнительную информацию. Если где-то необходимо добавить больше деталей, напишите дополнительный документ и перечислите в нем шаги, которые являются сложными или запутанными.

Такой подход имеет несколько преимуществ.

- Начальника, наверняка, интересует только общая настройка компьютерной среды в организации. Этого будет вполне достаточно, чтобы ответить на все вопросы как начальства, так и менеджеров. В подробности лучше не вдаваться, а то начальник еще решит разобраться во всем.
- То же верно и для клиентов.
- Любому сотруднику, переведенному на новую должность, необходимо быстро ознакомиться с инфраструктурой, чтобы нормально включиться в производственный процесс. "Загружать" таких сотрудников деталями не имеет никакого смысла, это только замедлит их адаптацию на новом месте.
- Намного эффективнее и быстрее выбрать один подходящий короткий документ, чем искать нужную информацию в длинном руководстве.
- Можно индексировать страницы, чтобы их было легко найти. Чем меньше времени администратор проводит в поиске, тем лучше.
- Поддерживать документацию в "текущем" состоянии намного проще, если можно обновлять по одной странице.

Последнее особенно важно. Обновление документации — это непростая задача; ее часто откладывают, когда не хватает времени. Имеется несколько подходов, которые позволяют поддерживать документацию в порядке.

Во-первых, нужно установить правило, что документация должна быть краткой, важной и вовсе не обязательно "отшлифованной". Главное — передать суть. Ничто так не отбивает желание составлять документацию, как осознание того, что придется писать целую "диссертацию" на тему проектирования. Предъявление высоких требований к составлению документации приведет только к тому, что никто вообще не будет ее составлять. Разработайте простую форму или шаблон для системных администраторов. Стандартная структура помогает системным администраторам регистрировать важную информацию, не раздувая документацию.

Во-вторых, нужно внедрить документацию в процессы. Комментарии в конфигурационных файлах являются чуть ли не самой лучшей документацией. Они всегда там, где нужны, и их сопровождение (обновление) занимает совсем немного времени. Во многие стандартные конфигурационные файлы разрешается добавлять комментарии, и даже в те из них, которые не очень "дружат" с комментариями, часто все равно можно вставить какую-нибудь дополнительную информацию.

Создаваемые локально средства могут требовать документацию в качестве части их стандартной конфигурационной информации. Например, программе, выполняющей установку нового компьютера, может быть нужна информация о владельце компьютера, размещении, уровне поддержки и оплате, даже если эти факты не влияют непосредственно на конфигурацию программных средств компьютера.

Документация не должна приводить к информационной избыточности. Например, если в организации имеется главный конфигурационный файл, в котором перечисляются все компьютеры и их IP-адреса, больше нигде эта же информация не должна обновляться вручную. Это чревато не только тратой дополнительного времени на обновление многочисленных источников, но и неизбежным появлением в них через какое-то время определенных несоответствий. При необходимости использовать эту информацию в другом контексте или другом конфигурационном файле, следует просто написать сценарий, который будет извлекать ее из главного конфигурационного файла (или обновлять ее там). Если устранить все избыточные источники не представляется возможным, нужно хотя бы определиться с тем, какой из них будет главным, а также написать программу для перехвата несоответствий и сделать так, чтобы она запускалась регулярно, например, из программы **cron**.

Появление таких инструментов, как Wiki, блоги и другие простые системы управления знаниями, намного упростило отслеживание документации по информационным технологиям. Выберите одно место, в котором можно найти и отредактировать все ваши документы. Одну страницу Wiki с 200 дочерними страницами в одном списке очень трудно использовать. Не забудьте включить функцию поиска в вашу систему.

### *Маркировка оборудования*

Иногда документация оказывается наиболее удобной, когда она написана на бумаге, приклеенной к устройству. Например, в случае отказа всей системы или сети от перечня действий мало толку, если он хранится на сломавшемся или недоступном компьютере.

Нужно, чтобы каждый компьютер можно было идентифицировать без включения и прохождения регистрации, потому что эти операции не всегда будут возможны. Промаркируйте все рабочие станции каким-нибудь уникальным образом (например, путем указания имени узла и его IP-адреса) и наклейте на них бумажки с контактной информацией службы поддержки.

В серверном зале все системы и их внешние устройства должны иметь уникальные метки с обеих сторон компьютеров (особенно тех, что установлены в узких стойках), чтобы, при необходимости отключить и снова включить какой-нибудь компьютер, можно было быстро отыскать выключатель, отключающий именно его питание.

При наличии множества систем разных типов полезно добавить дополнительную информацию, такую как сведения об архитектуре, инструкции по загрузке, специальные последовательности нажатия клавиш, ссылки на дополнительную документацию, телефон "горячей линии" производителя или номер телефона человека, который за все это отвечает. Записывание последовательностей нажатия клавиш может показаться немного глупым занятием, но серверы часто подключены к устаревшему терминалу или консоли, а не к специальному монитору.

Копию информации со всех наклеек следует обязательно сохранить также и в центральном хранилище записей или инвентаризационных данных. Она пригодится, если вы решите управлять своими компьютерами через TCP/IP-соединение с сервером консоли, вместо того чтобы проводить рабочий день в шумном машинном зале.

Если в организации есть много специфичных компьютерных данных, необходимо подумать о внедрении системы штрихкодов, которая позволит хранить всю релевантную документацию на ноутбуке. (Разумеется, такая мобильная система сама не должна зависеть от работы сети или сервера баз данных.)

### Сетевая документация

Сетевая проводка должна быть тщательно документирована. Подпишите все кабели, обозначьте коммутационные панели и сетевые розетки и пометьте сетевые устройства. Всегда делайте так, чтобы электрику было удобно обновлять документацию; повесьте на стенку в коммутационном шкафу ящик с карандашом и формами, для того чтобы он мог всегда быстро записать, что такой-то кабель был отключен от одного устройства и подключен к другому. И не забывайте регулярно переносить эти данные в электронное хранилище.

Большинство сетевых устройств (например, маршрутизаторы и коммутаторы) могут быть переконфигурированы по сети. Хотя это и позволяет управлять компьютерами в разных подсетях, не выходя из своего удобного кабинета, документация становится еще более важной. В таком случае следует соблюдать еще большую осторожность, потому что можно быстро и безнадежно испортить гораздо большую часть инфраструктуры.

Следует подумать об использовании специального программного обеспечения, такого как Rancid, чтобы отслеживать конфигурации устройств. Эти инструменты могут перехватывать случайные и забытые изменения, что поможет резко сократить время простоев.

### Документация для пользователей

Неплохой идеей является подготовка печатного документа, который можно будет выдавать новым пользователям. В этом документе следует перечислить местные "обычаи", правила уведомления о проблемах, имена и местонахождение принтеров, расписания резервного копирования и простоев и т.д. Такой документ может сэкономить системным администраторам и пользователям огромное количество времени. Эту информацию также следует сделать доступной в сети веб. Печатный документ дает большую уверенность в том, что новые пользователи его все-таки прочитают, зато на веб-страницу удобно ссылаться в случае возникновения вопросов. Поэтому лучше сделать и печатную, и веб-версию документа и не забывать регулярно их обновлять. Ничто не раздражает так, как устаревшие электронная документация или раздел часто задаваемых вопросов (FAQ).

Помимо документа с информацией о локальной компьютерной среде, можно также подготовить какой-нибудь вводный материал по системе UNIX. Мы предоставляем нашим пользователем одностраничные "шпаргалки", в которых перечислены наиболее часто используемые в нашей среде команды и приложения.

## Разделение окружающей среды

Организации, которые пишут и развертывают свое собственное программное обеспечение, должны разделить разработку, тестирование и производство, чтобы выпуски могли быть организованы с помощью структурированного процесса. Разделяйте окружающую среду, но примите меры, чтобы после обновления системы изменения учитывались в средах для тестирования и производства. Конечно, обновления самой конфигурации должны подвергнуться тому же самому виду структурированного контроля версий, что и код. "Изменения конфигурации" включают все: от заплат операционных систем до прикладных обновлений и административных изменений.

Очень важно защитить вашу производственную среду путем ролевого разделения в рамках процесса продвижения кода. Например, разработчики, имеющие административные привилегии в среде проектирования, не должны иметь административных прав и привилегий продвижения кода в другой окружающей среде. Раздраженный разработчик с правами продвижения кода может специально вставить вредоносный код на стадии разработки и затем продвинуть его на этап производства. Если вы распределяете обязанности одобрения и продвижения среди разных людей, то проблемы могут попасть на этап производства, только если много людей тайно сговорятся или сделают ошибки.

Зафиксируйте в документах процесс продвижения кода и следуйте ему неукоснительно. Не делайте исключений! Если вы находите, что регулярный процесс не достаточно эффективен для экстренных изменений, опишите в документации этот процесс экстренных изменений и затем удостоверьтесь, что он выполняется. Вы должны также перепроверить процесс продвижения кода и при необходимости внести корректировки.

Разработчиков иногда расстраивает уровень документации, который требуется в этом типе системы. Проведите несколько встреч за обеденным столом и помогите им понять побудительные мотивы ваших требований. Разработчики, которые стали вашими единомышленниками, более вероятно, будут следовать стандартным процедурам.

## Автоматизируйте, автоматизируйте, автоматизируйте!

Ваша система управления организацией должна содержать следующие главные элементы.

- **Автоматизированная установка новых компьютеров.** Это относится не только к инсталляции операционных систем, но и ко всему дополнительному программному обеспечению и локальным конфигурациям, необходимым для того, чтобы включить компьютер в производственный процесс. Ваш сайт неизбежно должен будет поддерживать несколько типов конфигурации, поэтому следует сразу включить разные типы компьютеров в свои планы.
- **Систематическое внесение исправлений и обновление существующих машин.** Когда вы выявляете проблему со своей установкой, нуждаетесь в стандартизированном и легком способе развертывания обновлений на всех затронутых машинах. Отметьте, что, поскольку компьютеры не включены все время (даже если это предполагается), ваша схема обновления должна правильно обращаться с выключенными компьютерами во время инициирования обновлений. Поиск обновлений можно

выполнять на этапе загрузки или по расписанию. Дополнительную информацию можно найти в главе 12.

- **Система мониторинга.** Вашим пользователям не придется звонить вам, чтобы сказать, что сервер "упал". Мало того, что это непрофессионально, но и вы понятия не имеете, надолго ли "упала" система. Вы должны обнаружить проблему еще до того, как вам позвонят первые пострадавшие. Вам нужна определенная система мониторинга, которая поднимет тревогу, как только проблемы станут очевидными. Однако сигнализация — тонкая вещь. Если сигналов слишком много, системные администраторы начинают их игнорировать, а если слишком мало, то важные проблемы остаются незамеченными.
- **Система связи.** Следите за потребностями ваших пользователей; их поддержка является конечной целью всего, что вы делаете как системный администратор. Система отслеживания запросов — это необходимость (см. раздел 32.2). Кроме того, полезно организовать централизованное место, где пользователи могут найти информацию о состоянии системы и контактную информацию (как правило, в сети веб).

## 32.6. Управление

Главной задачей руководства отдела информационных технологий является определение общей информационной стратегии и управление работой отдела. На плечи менеджера ложится много обязанностей.

- Выполнение роли руководителя группы, определение цели и предоставление необходимых ресурсов.
- Набор, увольнение, оценка и развитие навыков персонала.
- Распределение заданий и контроль за ходом их выполнения.
- Проверка и оценка качества работы.
- Согласование изменений и дополнений к соглашению о качестве оказываемых услуг.
- Взаимодействие с менеджерами в рамках всей организации.
- Решение проблем: конфликты между сотрудниками, недовольные пользователи, старое оборудование и т.д.
- Выполнение роли "высшей инстанции", к которой пользователи могут обращаться со своими проблемами.
- Контроль над разработкой масштабируемой инфраструктуры.
- Составление планов на случай аварии и других чрезвычайных обстоятельств.
- Извлечение документации из "замороченных" голов системных администраторов.
- Разработка и реализация стратегии обеспечения безопасности (как пользователей, так и администраторов).

Может показаться, что в этом списке не хватает такой обязанности, как общение с клиентами. Однако мы считаем, что эта роль больше подходит техническому персоналу. Менеджерам обычно не хватает технических знаний для того, чтобы правильно оценить степень сложности и выполнимости требований клиента. Сюрпризов будет меньше для обеих сторон, если в определении сроков доставки и составлении графиков выполнения будут принимать участие именно те, кто и будет этим непосредственно заниматься.

### Руководство

Да, это трудно описать. Но когда оно отсутствует или плохо осуществляется, это сразу же становится заметно. В некотором роде руководство — это "системное администрирование" организаций: оно тоже подразумевает выбор направления, проверку компонентов "на совместимость" и поддержание всей "системы" в рабочем состоянии с наименьшим возможным количеством "сообщений об ошибках".

К сожалению, технические навыки, позволяющие человеку стать замечательным администратором компьютерных систем, вовсе не означают, что он сможет хорошо справляться с ролью руководителя, которая требует, скорее, умения работать с людьми. Совладать с людьми гораздо сложнее, чем с языком Perl, например.

Новым менеджерам с хорошими техническими навыками может быть особенно трудно сфокусироваться на выполнении роли руководителя и избежать соблазна заняться разработкой. Намного проще и интереснее погрузиться в решение технической проблемы, чем вести длительные беседы с "проблематичным" сотрудником. Но что важнее для организации?

Проще всего определить свой уровень как руководителя так: составить список задач, над которыми сейчас работает организация, одним цветом выделить области, в которых всем руководишь ты, а другим — области, в которых всем руководит кто-то другой, а ты лишь ему подчиняешься, и потом посмотреть, какой цвет преобладает.

### Управление персоналом

Это особенно трудная задача. Выполняя руководящую роль, приходится иметь дело как с техническими, так и с личностными качествами сотрудников. Согласно стереотипу, системные администраторы, гениальные с технической точки зрения, как правило, совсем не умеют общаться и порой лучше ладят с машинами, чем с людьми. Менеджер должен следить за кривой роста у сотрудников в обеих этих областях.

Стимулировать и оценивать технический рост довольно легко, но личностный рост не менее важен. Ниже приведено несколько важных вопросов, которые следует задавать при оценке "пользовательского интерфейса" сотрудника.

- Вписывается ли поведение этого человека в вашу рабочую среду?
- Как этот человек общается с начальством, клиентами и поставщиками?
- Ладит ли этот человек с другими членами команды?
- Имеются ли у этого человека задатки руководителя, которые следовало бы развить?
- Как этот человек реагирует на критику и технические споры?
- Старается ли этот человек устранить пробелы в своих знаниях?
- Каковы коммуникативные навыки этого человека?
- Может ли этот человек спланировать, реализовать и продемонстрировать проект клиента?
- Чувствует ли этот человек ответственность за выполнение поставленной задачи?
- Стремится ли этот человек решать задачи или видит одни препятствия?

### Набор персонала

Оценивать подобным образом следует не только уже работающих сотрудников, но и тех, кто претендует на работу в данной организации. Личностные качества кандидатов часто не учитываются или недооцениваются. Будьте внимательными, чтобы потом не пришлось пожалеть!

Существует два подхода к формированию штата системных администраторов.

- Нанимать опытных специалистов.
- Растить своих собственных.

Опытные специалисты обычно быстрее включаются в работу, но всегда есть что-то, от чего их нужно отучать. Неопытных специалистов, наоборот, придется интенсивно учить. Как бы то ни было, желательно иметь точно сформулированные и зафиксированные в документе правила и процедуры. Если существующий штат сотрудников имеет ясную цель и понимает ваши правила, они могут сами стать лидерами и помочь акклиматизироваться новичкам.

Некоторые качества хорошего системного администратора противоречат друг другу. Администратор должен быть достаточно дерзким для того, чтобы не бояться принимать неординарные решения при возникновении сложной проблемы, но при этом и достаточно осторожным для того, чтобы не нанести вред системе. Навыки в межличностных отношениях и решении проблем одинаково важны, хотя, по нашим наблюдениям, у многих администраторов они являются взаимоисключающими. И хотя системные администраторы не обязаны блестяще владеть навыками коммуникации, некоторым из них удается пройти долгий путь навстречу клиентам.

Нанимая системных администраторов, вы должны решить, какие личностные качества являются самыми важными для конкретной роли. Например, если вы нанимаете администратора сервера, который будет сосредоточен на прикладной части и будет редко взаимодействовать с клиентами, то коммуникативные способности менее важны, чем технические навыки. Если же этот человек будет частью многочисленной команды, вы не можете полностью проигнорировать его коммуникабельность.

Для того чтобы оценить его техническую подготовку, поставьте перед ним ряд соответствующих технических задач. Вы могли бы даже подбросить ему коварный вопрос, чтобы измерить его интеллектуальный уровень.

Мы полагаем, что личная беседа имеет большое значение. За первые 15 минут личной беседы вы узнаете больше, чем в более длинном телефонном разговоре.

Мы также верим в рекомендательные письма. Во время проверки рекомендательных писем, мы любим задавать открытые вопросы, которые дают респонденту шанс сделать тонкий намек о соискателе. Слушайте внимательно! Людям вообще не нравится говорить отрицательные вещи во время проверки рекомендаций, но вы можете уловить тонкий намек, если будете внимательны.

После интервью с кандидатом у вас могут появиться еще кое-какие вопросы. Например, если в ходе беседы возникнет вопрос о том, обращает ли претендент внимание на детали, вы могли бы спросить человека, давшего ему рекомендацию, что-нибудь вроде следующего: "Считаете ли вы претендента человеком, ориентированным на детали, или он любит мыслить крупными категориями?"

### Увольнение персонала

Допустив ошибку при подборе сотрудника, лучше уволить его как можно раньше. Промахи с подбором персонала случаются, но держать на работе людей, не желающих тянуть общую лямку, — значит настраивать против себя других сотрудников, потому что именно им придется взваливать на свои плечи обязанности этих лентяев и подчищать за ними их работу. Клиенты тоже быстро заметят, что кое-кто не делает то, чего от него просят, и начнут требовать, чтобы их задания выполнял какой-нибудь конкретный си-

стемный администратор. Мало кому из менеджеров понравится давление со стороны клиентов и их подсказки, кому какое задание следует дать.

Во многих организациях сложно уволить сотрудника, особенно по истечении испытательного срока. По этой причине к испытательному сроку следует относиться серьезно.

Не исключено, что позже все-таки придется собирать факты, подтверждающие некомпетентность сотрудника, делать ему официальные выговоры, устанавливать рабочие нормы и т.д.

### Тонкости управления персоналом

Для интеграции нового сотрудника в вашу инфраструктуру необходимо нечто большее, чем простое письмо с приглашением на работу. Вы должны понимать и соблюдать правила вашей организации относительно размещения объявлений о вакансиях, испытательных сроках, собеседовании и т.д.

Еще один набор правил должен определять процедуру выделения новому сотруднику стола, компьютера, ключей, учетных записей и доступа **sudo**. Ваши процедуры должны гарантировать, что найм системным администратором сотрудника для управления конкретным набором серверов не означает карт-бланш на набор администраторов для любой системы, существующей в компании.

Не менее важно выполнять процедуры и правила, когда системный администратор покидает компанию. Вы должны составить обходной лист, чтобы гарантировать, что ничего не забыли. Ваш обходной лист должен включать следующие пункты.

- Удаление доменной учетной записи пользователя (LDAP или Active Directory).
- Удаление учетной записи пользователя в системах UNIX и Linux.
- Удаление пользователя из всех файлов **sudoers** вашей организации.
- Возвращение ключей и карт доступа (регистрируйте все собранные ключи и карты).
- Возвращение сотового телефона компании.

Некоторые организации могут напечатать точный список прав доступа и аппаратных средств, выданных каждому служащему. Это отличный способ удостовериться, что вы ничего не забыли.

В Соединенных Штатах Америки служащим принято высылать уведомление за две недели до увольнения. Некоторые организации не желают ждать две недели и выставляют сотрудника за дверь, немедленно отменяя любой физический доступ к рабочему месту и в сеть. Это разумно!

### Тщательное тестирование и контроль качества выполняемой работы

Менеджеры задают тон для контроля качества. У каждой задачи должны быть ясные критерии ее выполнения. Кроме действий, характерных для каждой задачи, критерии ее выполнения могли бы включать следующие факторы.

- Тестирование решения.
- Обновление локальной документации.
- Рассылка решения на все задействованные компьютеры.
- Выполнение заявки на устранение неисправности с подробным указанием предпринятых мер.
- Просьба лицу, пославшему заявку, поставить отметку о том, насколько он удовлетворен.

Даже простая задача, такая как создание расписания `cron` для выполнения ежедневных административных процедур, должна включать фазу тестирования, чтобы гарантировать, что изменение работает как надо. Более сложные задачи должны иметь утвержденный план испытаний.

В идеале отдел информационных технологий должен внедрить культуру, в которой системные администраторы гарантируют, что каждая работа выполняется качественно и полностью. Однако для этого вам, вероятно, придется установить жесткий контроль; это тот случай, когда можно применить принцип "доверяй, но проверяй".

### Управление, а не вмешательство

Технически компетентный менеджер может советовать сотрудникам, как они должны выполнять свою работу. Но таким образом он может лишить их шанса приобрести собственные навыки и стать полностью ответственным за их работу.

Мы думаем о развитии сотрудника в некотором роде как о его воспитании. Если сотрудник может совершить серьезную ошибку, вызвать проблему, которую трудно устранить, попросите, чтобы сотрудник показал свой план действий, и удостоверьтесь, что он понимает вероятные последствия своего плана. В противном случае вы, вероятно, должны будете вмешаться.

С другой стороны, если кто-то может сделать ошибку, которая послужит хорошей возможностью для обучения без нанесения серьезного ущерба, то стоит устраниться. Уроки, извлеченные из непосредственного опыта, усваиваются лучше.

Конечно, оба варианта не идеальны. Вы не хотите, чтобы вас воспринимали как мелочного менеджера, но вы также не хотите выглядеть руководителем, не желающим ничего знать или позволяющим сотрудникам терпеть неудачу, которой можно было избежать. Поддержите своих сотрудников, даже если они совершили ошибки, и помогите им. Никогда не позволяйте ошибкам стать постоянным источником затруднений.

### Связи с сотрудниками организации

Системное администрирование — забавное занятие. Если вы делаете свою работу хорошо, пользователи считают вашу незаметную вычислительную среду само собой разумеющимся, и никто не замечает то, что вы делаете. Однако в сегодняшнем мире вирусов, спама, раздутых приложений и тотальной зависимости от Интернета, сотрудники отдела информационных технологий — обязательная часть организации.

Ваши удовлетворенные клиенты — ваш лучший маркетинговый индикатор. Однако есть другие способы стать заметным в вашей организации и даже в пределах более широкого сообщества. Основываясь на нашем опыте "саморекламы", мы предлагаем следующие методы, так как считаем их особенно эффективными.

- Проводите общие собрания, на которых пользователи могут формулировать свои проблемы и задавать вопросы о вычислительной инфраструктуре. Проанализируйте просьбы пользователей о помощи, чтобы кратко описать самые неприятные темы, которые вы обнаружили. Обеспечьте закуски, чтобы гарантировать хорошую явку.
- Оставьте достаточно много времени для вопросов и удостоверьтесь, что вы имеете хорошо осведомленный штат, чтобы ответить на них. Не пытайтесь блефовать, отвечая на неожиданные вопросы. Если вы не знаете ответа в данный момент, лучше признаться в этом и отложить решение вопроса.
- Запланируйте ряд семинаров, предназначенных для конечных пользователей в вашей организации. Проводите их один раз в два-три месяца и заранее оглашайте темы, которые будут на них освещаться.

- Посещайте конференции по системному администрированию, делайте доклады или пишите статьи об инструментах, которые вы разрабатываете. Такие презентации не только способствуют взаимодействию с вашими коллегами, но и демонстрируют вашим клиентам (и вашему боссу), что вы делаете свою работу хорошо.

Системное администрирование, в конечном счете, сводится к общению с людьми и удовлетворению их потребностей. Личные отношения столь же важны, как и в любом другом деле. Разговаривайте со своими клиентами и коллегами и уделите время личным обсуждениям и обмену мнениями.

Если вы поддерживаете работу нескольких групп людей, подумайте о том, чтобы поручить определенному сотруднику действовать как менеджер по работе с заказчиками в рамках группы. Этот сотрудник должен взять на себя ответственность за общее удовлетворение клиентов и регулярно общаться с конечными пользователями. Канал новостей и передача информации об изменениях в вычислительной среде через посредника позволят создать дополнительные возможности для контакта.

### Манипулирование вышестоящим руководством

Для того чтобы эффективно выполнять свои обязанности руководителя (особенно лидера), нужно уважать и поддерживать свое собственное руководство. Нужно умело подбирать штат своих сотрудников и распределять, кто будет принимать решение о приеме и увольнении людей. Нужно контролировать процесс распределения заданий и определить, кто отвечает за их выполнение и назначение новых задач сотрудникам. И, наконец, нужно нести ответственность за представление своей команды как внутри собственной организации, так и вне ее.

Вышестоящее руководство часто не имеет ни малейшего представления о том, чем занимаются системные администраторы. Предоставляйте им такую информацию, пользуясь возможностями своей системы отслеживания запросов, когда хотите убедить начальника в необходимости найма дополнительных сотрудников или приобретении нового оборудования. Примите меры, чтобы ваше руководство понимало разницу между службой поддержки и оперативной группой. Они должны понимать, что сотрудник, отвечающий по телефону службы поддержки, — это не тот специалист, который конфигурирует маршрутизаторы и серверы. Это позволит избежать многих недоразумений.

Фиксировать информацию о выполняемой системными администраторами работе может быть полезно даже при отсутствии конкретной цели. Руководители, особенно менеджеры нетехнического звена, часто недооценивают сложность и трудоемкость стоящих перед системными администраторами задач. Особенно это касается задач, связанных с поиском и устранением неисправностей.

Старайтесь подходить к планированию реалистично. Если не уверены, увеличьте время, которое, на ваш взгляд, необходимо для решения масштабных и очень важных задач. Если обновление системы будет выполнено за два дня, вместо трех, пользователи только поблагодарят вас вместо того, чтобы ругать, как они бы сделали, если бы вы решили, что оно может быть выполнено за один день.

Когда наступает время вносить изменения в производственные системы, необходимо строго придерживаться утвержденной процедуры. Она должна предусматривать получение одобрения от комиссии экспертов по изменениям. Участие руководства в работе этой комиссии позволит в будущем уменьшить количество жалоб от пользователей. Пользователи, увидевшие в составе этой комиссии исполнительного директора, при переходе на новую систему электронной почты вряд ли станут настаивать на сохранении старой.

Труднее всего обычно получить поддержку руководства в вопросах ужесточения стратегии безопасности. Чем жестче правила безопасности, тем больше неудобств испытывают пользователи, а поскольку их больше и ноют они громче, их протесты быстро находят отклик у начальства. Увеличение степени безопасности может снижать продуктивность пользователей, поэтому прежде чем вносить то или иное изменение в систему защиты, следует проводить анализ степени риска, дабы быть уверенными в том, что руководство и пользователи понимают, почему оно предлагается.

Также нужно следить за тем, чтобы пользователи были заранее оповещены обо всех планирующихся изменениях в стратегии безопасности, которые могут повлиять на их работу (например, о переходе с паролей на ключи RSA/DSA для удаленной регистрации). Все эти изменения должны быть хорошо документированы и сопровождаться поддержкой во время внедрения. Документация должна быть понятной и содержать пошаговые инструкции по работе с новой системой. При переходе на новую систему лучше выделить дополнительное время на обслуживание пользователей, которые могут начать паниковать из-за того, что не успели прочитать свою электронную почту.

### *Приобретение оборудования*

Во многих организациях системные администраторы не принимают решения о закупках. Иногда это оправданно, но при покупке компьютерной техники системные администраторы должны иметь возможность высказывать свое мнение и даже выбирать ту или иную систему.

Системные администраторы могут сообщить много полезной информации об имеющихся в локальной среде требованиях к совместимости, о компетентности поставщиков (особенно сторонних посредников) и надежности определенных типов оборудования. Информация о надежности играет особенно важную роль при покупке системы, от которой зависит работа всей организации.

Еще одна важная информация, которую может сообщить системный администратор, — воздействие новой системы на информационную безопасность и ее соответствие законодательству. Хороший пример — клиническое отделение больницы, которое заказывает новую систему обработки изображений, не консультируясь с группой системного администрирования. К сожалению, в результате оказывается, что новое оборудование не способно взаимодействовать с существующей системой аутентификации, работающей в больнице. Фактически новое оборудование не требует регистрации пользователей вообще! Больница потратила тысячи долларов на систему, которая не соответствует закону HIPAA, и теперь она вынуждена покупать новую систему или просить разработчика установить индивидуальную систему аутентификации. Ни одно из этих решений не является удачным. Для больницы было бы лучше, если бы системные администраторы были с самого начала привлечены к решению этой проблемы и могли рекомендовать другого поставщика.

Системные администраторы должны уведомляться о предстоящих заказах любого нового оборудования для того, чтобы они могли определить, как его можно будет интегрировать в текущую инфраструктуру и какие проекты и ресурсы будут необходимы для его поддержки.

Несмотря на то что системные администраторы могут давать рекомендации, окончательное решение о приобретении оборудования принимает руководство. Если организация купила нечто, что вам не нравится, вы обязаны с этим смириться — игнорировать системы, которые вам не нравятся, нельзя.

Участие системного администратора в разработке спецификаций приобретаемых систем особенно важно в тех организациях, где все покупается по минимальным ценам. В большинстве случаев при закупке разрешается указывать критерии оценки. Не стоит забывать включать и такие пункты, как "должна быть совместима с существующей средой" или "должна позволять нормально работать с пакетом программ XYZ".

Показатель воздействия и стоимости дополнительного компонента оборудования (или иногда программного обеспечения) не постоянен. Является ли он шестидесятым в этой архитектуре или первым? Достаточно ли у него места на жестком диске для системных файлов? Хватит ли у него памяти для выполнения современных объемных приложений? Имеется ли свободный сетевой порт, к которому его можно подключить? Установлена ли на нем совершенно новая операционная система? Соответствует ли он текущему законодательству? Как он согласуется с долговременными планами предприятия? Одобрили ли его приобретение архитекторы всей системы предприятия?

### *Разрешение конфликтов*

Некоторые из ложащихся на плечи менеджера обязанностей связаны непосредственно с умением ладить с людьми (как правило, клиентами или сотрудниками) в напряженных ситуациях. Мы сначала рассмотрим, каким должно быть поведение менеджера в целом, а затем поговорим о таком особом случае, как работа с "норовистыми" клиентами, которых иногда еще также называют ковбоями.

В мире системного администрирования конфликты чаще всего возникают между системными администраторами и их клиентами, коллегами или поставщиками. Например, клиент может быть не доволен качеством предоставленных ему услуг, поставщик мог не предоставить обещанный материал вовремя, коллега мог сделать не то, что его просили, а конструкторский отдел мог требовать полного контроля над установленными на их компьютерах конфигурациями операционной системы.

### *Посредничество*

Большинство людей не любят говорить о конфликтах или даже признавать, что они существуют. Когда эмоции уже бушуют, это обычно означает, что конфликт распознан слишком поздно и что негатив накапливался на протяжении длительного периода времени. За это время стороны могли уже затаить серьезную обиду и не раз прокрутить в своей голове "подлые" намерения друг друга.

Встреча лицом к лицу под контролем нейтрального посредника иногда позволяет разрядить обстановку. Попробуйте ограничить спор одним вопросом, а выделенное на его обсуждение время — одним часом. Такие меры снижают вероятность превращения встречи в бесконечную "схватку". Цель такой встречи — выработка взаимовыгодного решения. Формальную подготовку в этой области можно получить в разных организациях, но мы сформулируем основные принципы.

- Дайте каждой стороне шанс выразить свое представление о желаемом результате. Запишите его нейтральным способом на доске, что обе стороны могли его видеть.
- Ваша цель как посредника состоит в том, чтобы выявить точки соприкосновения, рассматривая оба варианта желаемых результатов.
- Вы не сможете достигнуть соглашения за одну встречу. Но если вы сделаете шаги к обнаружению точек соприкосновения, считайте встречу успехом.
- Основывайтесь на любых точках соприкосновения, которые вы выявили на последующих встречах. После нескольких встреч вы сможете найти достаточно много точек соприкосновения, чтобы обе стороны были удовлетворены результатом.

### Норовистые пользователи и конфликтные отделы

Процесс внедрения строго контролируемых систем часто приводит к возникновению конфликтов. Технические пользователи (а иногда и целые отделы) могут посчитать, что централизованное системное администрирование не может адекватно удовлетворять их конфигурационные потребности или что они нуждаются в автономном контроле над используемыми ими компьютерами.

Первым импульсом менеджера может быть просто взять и заставить этих норовистых пользователей принять стандартную конфигурацию для того, чтобы свести к минимуму затраты и время, требующиеся на их поддержку. Однако такой жесткий подход обычно заканчивается недовольством как среди пользователей, так и среди системных администраторов. Не забывайте, что желания норовистых пользователей зачастую бывают вполне законными и что именно системным администраторам приходится поддерживать их или, по крайней мере, воздерживаться от усложнения им жизни.

Лучше всего в такой ситуации поступить следующим образом: выяснить причины, по которым норовистые пользователи не хотят принимать управляемые системы. В большинстве случаев их потребности можно удовлетворить и тем самым поставить их на место.

В качестве альтернативного варианта можно согласиться на автономию. Разрешите своим норовистым пользователям или отделам делать все, что им хочется, при условии что они сами будут отвечать за поддержание своих систем в рабочем состоянии. Установите брандмауэр, чтобы защитить контролируемые вами системы от возможного взлома или вирусов со стороны сети этих норовистых пользователей.

Обязательно заставьте всех представителей автономной сети подписать соответствующий стратегический документ, оговаривающий определенные правила безопасности: например, если их системы начнут мешать работе организации, они будут отключены от общей сети до тех пор, пока на них не будут установлены необходимые заплаты и обновления и они не перестанут негативно влиять на производственный процесс.

Во всех организациях есть так называемые "пользователи-передовики", которые любят устанавливать все самые последние новинки сразу. Такие пользователи не чураются ни бета-версий, ни нестабильных предварительных выпусков программных продуктов; им главное, чтобы у них было самое новое программное обеспечение. К таким пользователям следует научиться относиться как к полезным ресурсам, а не как к "бельму на глазу". Они являются просто идеальными кандидатами для тестирования нового программного обеспечения и зачастую с удовольствием составляют отчеты об обнаруженных в нем дефектах, тем самым предоставляя возможность вовремя устранить проблемы.

Творческое системное администрирование также необходимо, чтобы иметь дело с увеличивающимся числом мобильных устройств, принесенных на работу. Вы должны найти способы оказывать услуги и для этих (вообще-то, не вызывающих доверия) устройств, чтобы не подвергать опасности целостность ваших систем. В этой ситуации хорошей идеей могла бы быть отдельная сеть. Другой выбор — запускать ноутбуки через сеть VPN, которая производит "оценку положения".

Оценка положения гарантирует, что ноутбуки придерживаются вашей самой важной политики безопасности. Например, вы могли бы потребовать, чтобы у всех машин, соединяющихся через сеть VPN, были установлены критически важные заплаты. Для ноутбуков, работающих под управлением операционной системы Windows, вы могли бы также потребовать наличия антивирусного программного обеспечения.

# 32.7. Инструкции и процедуры

Исчерпывающие инструкции и процедуры, касающиеся информационных технологий, служат основой для работы современных организаций. Инструкции устанавливают нормы для пользователей и администраторов и способствуют согласованной работе всех вовлеченных в нее сотрудников. Все больше и больше инструкции требуют подтверждения в форме подписи или другого доказательства, что пользователь согласился их соблюдать. Хотя это может показаться чрезмерным формализмом, но такое подтверждение представляет собой действительно отличный способ защитить администраторов.

Хорошим основанием для разработки инструкций является стандарт ISO/IEC 27001. Он сочетает общую стратегию в области информационных технологий с другими важными элементами, такими как информационная безопасность и роль отдела кадров. В следующих разделах мы обсудим стандарт ISO/IEC 27001 и выдвинем на первый план некоторые из его самых важных и полезных элементов.

## Различие между инструкциями и процедурами

Инструкции и процедуры — это разные категории, но их часто путают, и иногда они даже используются как синонимы, что вызывает путаницу. Для того чтобы правильно понимать их сущность, рассмотрим следующие аспекты.

- Инструкции — это документы, устанавливающие требования или правила. Требования обычно определяются на относительно высоком уровне. Примером инструкции можно считать требование, чтобы добавочное копирование выполнялось ежедневно, а резервное копирование с копиями уровня 0 — каждую неделю.
- Процедуры — это документы, описывающие процесс выполнения требований или правил. Так, процедура, связанная с описанной выше инструкцией, могла бы быть сформулирована примерно так: "Добавочное копирование выполняется с помощью программы Backup Exec, инсталлированной на сервере backups01 ..."

Это различие важно, потому что инструкция не должна изменяться очень часто. Вы могли бы пересматривать их ежегодно и, возможно, изменять один или два раздела. С другой стороны, процедуры уточняются непрерывно, поскольку постоянно изменяется архитектура, системы и конфигурации.

Некоторые стратегические решения диктуются программным обеспечением, которым вы управляете, или инструкциями внешних групп, например интернет-провайдерами. Если необходимо защитить конфиденциальные данные пользователей, то некоторые инструкции носят категорический характер. Мы называем эти инструкции "не подлежащими обсуждению".

В частности, мы полагаем, что IP-адресами, именами узлов, идентификаторами пользователей, идентификаторами групп и именами пользователей необходимо управлять централизованно. Некоторые организации (транснациональные корпорации, например) являются слишком большими, чтобы осуществить эту политику, но если вы можете реализовать ее, то централизованное управление сделает все намного проще. Мы знаем компанию, которая реализует политику централизованного управления для 35 тысяч пользователей и 100 тысяч компьютеров. Таким образом, порог, после которого организация становится слишком крупной для централизованного управления, должен быть довольно высоким.

Другие важные проблемы имеют более широкую сферу влияния, чем ваша локальная группа системных администраторов.

- Борьба со взломами системы безопасности.
- Контроль над экспортом файловой системы.
- Критерии выбора паролей.
- Удаление регистрационных записей.
- Материал, защищенный авторским правом (например, MP3 и DVD).
- Программное пиратство.

## Эффективные инструкции

Существует несколько стратегических инструкций, охватывающих примерно такую же область. Ниже перечислены примеры инструкций, которые, как правило, включаются в стратегический набор инструкций, относящихся к информационным технологиям.

- Информационная политика безопасности.
- Соглашения о возможности соединения с посторонними организациями.
- Политика управления активами.
- Информационная система классификации.
- Политика безопасности людей.
- Физическая политика безопасности.
- Политика управления доступом.
- Стандарты безопасности для разработки и обслуживания новых систем.
- Политика управления инцидентами.
- Управление непрерывностью бизнеса (аварийное восстановление).
- Политика соответствия установленным требованиям закона.

## Процедуры

Процедуры в форме контрольных списков или рецептов могут формализовать существующую практику. Они полезны и для новичков, и для опытных системных администраторов. Еще лучше процедуры, которые содержат выполняемые сценарии.

У стандартных процедур есть несколько преимуществ.

- Рутинные операции всегда выполняются единообразно.
- Контрольные списки уменьшают вероятность ошибок или забытых операций.
- По рецепту системный администратор работает быстрее.
- Изменения самодокументируются.
- Письменные процедуры обеспечивают измеримый стандарт корректности.

Вот некоторые общие задачи, для которых вы могли бы захотеть сформулировать процедуры.

- Подключение компьютера.
- Добавление пользователя.
- Конфигурирование локального компьютера.

- Настройка процедур резервного копирования для нового компьютера.
- Защита нового компьютера.
- Повторный запуск сложных компонентов программного обеспечения.
- Перезагрузка веб-серверов, которые не отвечают на запросы или "зависли".
- Очистка очереди на печать и перезагрузка принтера.
- Модернизация операционной системы.
- Установка заплат.
- Установка пакета прикладных программ.
- Установка программного обеспечения по сети.
- Обновление наиболее важных программ (**sendmail**, **gcc**, **named**, OpenSSL и т.д.).
- Резервное копирование и восстановление файлов.
- Уничтожение резервных лент.
- Аварийный останов системы (всех компьютеров; кроме самых важных; и т.д.).

Такие вопросы следует строго регламентировать, чтобы не стать жертвой известной уловки четырехлетних детей: "Мама не разрешила, надо спросить у папы!"

# 32.8. Восстановление после аварий

Работа организации зависит от функционирования информационной среды. Системный администратор отвечает не только за повседневные операции, но и за наличие плана действий на случай возникновения различных экстренных ситуаций, по крайней мере тех, которые можно предвидеть. Подготовка к таким масштабным проблемам влияет как на общий план работы, так и на способ выполнения повседневных операций.

## Оценка рисков

Прежде чем завершить разработку плана восстановления после аварий, целесообразно оценить степень риска, чтобы понять, какими активами вы располагаете, каким рискам подвергаетесь и как можно смягчить последствия аварии. Специальный документ NIST 800-30 детализирует обширный процесс оценки степени риска. Вы можете загрузить его с адреса

```
http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf
```

В ходе оценки степени риска необходимо составить список потенциальных угроз, от которых вы хотите защититься. Угрозы не являются одинаковыми, и вам, возможно, понадобится несколько различных планов, чтобы покрыть полный спектр возможностей. Например, перечислим угрозы общего характера.

- Наводнения.
- Пожары.
- Землетрясения.
- Ураганы и торнадо.
- Магнитные бури и всплески электропитания.
- Перебои в питании, короткие и долгосрочные.

- Чрезвычайно высокая температура или отказ охлаждающего оборудования.
- Отказы аппаратных средств ("упавшие" серверы, "сгоревшие" жесткие диски).
- Отказы сетевых устройств (маршрутизаторов, выключателей, кабелей).
- Злонамеренные пользователи, внешние и внутренние[2].
- Случайные пользовательские ошибки (удаленные или поврежденные файлы и базы данных, потерянная информация о конфигурации, потерянные пароли и т.д.).

Для каждой потенциальной угрозы рассмотрите и запишите все возможные последствия.

Как только вы поймете угрозу, расставьте приоритеты служб в пределах своей информационной среды. Составьте таблицу, в которой перечисляются службы и их приоритеты. Например, компания, предлагающая "программное обеспечение как службу", может оценить свой внешний веб-сайт как службу первостепенной важности, в то время как офис с простым, информационным внешним веб-сайтом не мог бы волноваться о судьбе организации во время бедствия.

## Борьба со стихийными бедствиями

Все больше организаций проектируют свои критические системы так, чтобы автоматически переключаться на вспомогательные серверы в случае проблем. Это прекрасная идея, если вы не допускаете отключение службы. Однако не становитесь жертвой веры в то, что отражение ваших данных отменяет необходимость в резервном копировании. Даже если ваши информационные центры расположены далеко, вы вполне могли потерять оба сервера. Не забудьте включить резервное копирование данных в свой план борьбы с бедствиями.

🕮 Облачные вычисления описаны в разделе 24.1.

Облачные вычисления — еще один ресурс для борьбы со стихийными бедствиями, который становится все более популярным. Благодаря таким службам, как EC2 компании Amazon, вы можете настроить удаленный сайт и запустить его за несколько минут, без необходимости платить за специализированные аппаратные средства. Вы платите только за то, что используете. Это прекрасная и недорогая альтернатива специализированному резервному сайту, хотя и требует значительного технического планирования.

План восстановления после аварии должен включать следующие разделы (на основе стандарта аварийного восстановления NIST 800-34).

- **Введение** — цель и предмет документа.
- **Понятие операций** — описание системы, цели восстановления, классификация информации, порядок преемственности, обязанности.
- **Уведомление и активация** — процедуры уведомления, процедуры оценки повреждений, активация плана.
- **Восстановление** — последовательность событий и процедур, требуемых для восстановления потерянных систем.
- **Возврат к нормальному функционированию** — параллельная обработка, воссозданное тестирование системы, возвращение к нормальному функционированию, деактивация плана.

[2] По данным 2005 года половина взломов системы безопасности осуществлялась инсайдерами.

Мы привыкли использовать сеть для общения и доступа к документам. Однако эти средства могут оказаться недоступными или оказаться под угрозой после инцидента. Сохраняйте автономно все соответствующие контакты и процедуры. Вы должны знать, где можно получить недавние буферные ленты и как использовать их независимо от сетевых данных.

Во всех сценариях бедствий вам понадобится доступ и к автономным, и к интерактивным копиям существенной информации. Интерактивные копии по возможности должны быть сохранены на самостоятельном компьютере, на котором есть богатый набор инструментов, среды ключевых системных администраторов, управление собственным сервером, полноценный локальный файл `/etc/hosts`, соединение с принтером и нет никаких зависимостей от обмена файлами и т.д. Не используйте старую рухлядь, это бесполезно; компьютер для хранения информации, необходимой для аварийного восстановления, должен быть быстродействующим и должен иметь много памяти и дискового пространства, которое вы можете использовать для восстановления. Компьютер должен иметь полноценную среду проектирования, чтобы можно было исправить и повторно собрать любое поврежденное программное обеспечение. Хорошо, если у этого компьютера есть интерфейсы для всех типов дисководов, используемых на вашем сайте (IDE, SATA, SCSI, FC-AL и т.д.).

Вот список данных для хранения на резервном компьютере в форме печатного буклета или оптического диска.

- Схема процедуры восстановления: кому звонить, что сказать.
- Номера телефонов сервисного центра и клиентов.
- Ключевые местные номера телефона: полиция, пожарные, сотрудники, босс.
- Запас резервных лент и резервного графика их создания.
- Карты сети.
- Регистрационные номера программного обеспечения, лицензионные данные и пароли.
- Копии инсталляционных пакетов программного обеспечения (могут храниться в формате ISO).
- Копия инструкций по эксплуатации ваших систем.
- Контактная информация продавца поврежденного диска, в которой вы нуждаетесь.
- Административные пароли.
- Данные о конфигурациях аппаратного и программного обеспечения: версии операционных систем, заплаты, таблицы разделов, параметры настройки аппаратных средств IRQ, DMA и т.п.
- Инструкции по запуску систем, которые должны быть восстановлены в интерактивном режиме.

## Подбор персонала на случай аварии

Нужно заблаговременно решить вопрос о том, кто будет справляться с ситуацией в случае, если произойдет авария. Следует составить план действий и написать номера телефонов тех, кому надлежит звонить в такой ситуации. Может оказаться, что на роль главного больше подходит кто-нибудь из системных администраторов, а не начальник отдела информационных технологий (начальники отделов обычно не подходят для этой роли).

Ответственным в случае аварии должен быть кто-то, кто пользуется авторитетом и не боится принимать трудные решения при наличии минимального количества информации (наподобие решения отключить от сети весь отдел целиком). Способность принимать такие решения, уведомлять о них понятным образом и фактически руководить персоналом во время кризиса, пожалуй, важнее теоретических знаний о системном и сетевом управлении. Мы пользуемся небольшой ламинированной карточкой, на которой мелким шрифтом напечатаны все важные имена и номера телефонов: она очень удобна, потому что легко помещается в бумажник.

При составлении плана аварийных мероприятий обычно предполагается, что административный персонал будет на месте, когда произойдет авария, и сумеет справиться с ситуацией. К сожалению, люди иногда болеют, уходят на курсы повышения квалификации, уезжают в отпуск, а в тяжелые времена вообще могут даже становиться крайне недружелюбными. Поэтому стоит заранее продумать, где можно будет быстро найти дополнительную помощь. (Если система не слишком устойчива, а персонал неопытен, то недостаточное количество администраторов уже само по себе является аварийной ситуацией.)

Одним из решений может быть договоренность с какой-нибудь местной консультационной компанией или университетом, где всегда есть талантливые и готовые помочь администраторы. Конечно, если у них когда-нибудь возникнут проблемы, тогда вы тоже должны будете оказать им необходимую помощь. Но самое главное — это не работать на пределе: наймите достаточное количество администраторов и не требуйте, чтобы они работали по 12 часов в сутки.

## Электропитание и кондиционирование

План аварийных мероприятий лучше проверить до того, как он понадобится. Непроверенный план вообще не является планом! Тестировать и обновлять план следует ежегодно.

🕮 Информацию об источниках бесперебойного питания можно найти в разделе 27.3.

Генераторы и источники бесперебойного питания следует проверять раз в месяц или в квартал, в зависимости от того, на какую степень риска готово согласиться руководство. Нужно убедиться в том, что все важные устройства подключены к источнику бесперебойного питания, их батареи в порядке и механизм включения питания работает. Для того чтобы проверить какой-нибудь отдельный источник бесперебойного питания, нужно просто вынуть вилку из розетки, а чтобы проверить все источники, т.е. все ли важное оборудование к ним подключено, возможно, придется отключить питание во всем здании или в комнате. В любом случае следует обязательно знать обо всех зависимостях и слабых местах своей системы электропитания.

Источники бесперебойного питания тоже нуждаются в обслуживании. Эта задача может и не входит в обязанности системного администратора, но именно он должен следить за тем, чтобы она выполнялась.

Если у вас есть генератор, заключите с местной компанией контракт на поставку топлива. Храните достаточное количество топлива, чтобы обеспечивать свои системы электричеством во время простоя, но помните, что топливо в конце концов портится. Бензин начинает портиться примерно через месяц. Даже если добавить в него стабилизатор, бензин не может храниться более года. Дизельное топливо является более химически стабильным, но может способствовать росту водорослей, поэтому в дизельное топливо, которое будет храниться долго, следует добавлять средства, подавляющие рост водорослей.

Чаще всего электричество отключается ненадолго, но на всякий случай батареи должны обеспечивать два часа работы, чтобы было время правильно выключить технику. Большинство источников бесперебойного питания оборудованы портами USB или интерфейсом Ethernet, которые можно использовать для корректного выключения не являющихся критически важными компьютеров через определенное время после отключения питания.

Периоды отключения электропитания можно использовать для выполнения любых пятиминутных процедур обновления, которые уже были протестированы, но еще не проводились. Работать все равно невозможно, так что людям будет все равно. В некоторых организациях пользователи спокойнее воспринимают дополнительную пятиминутную задержку после отключения электричества, чем пятиминутное плановое отключение системы, о котором их оповестили за неделю. Если есть старые компьютеры, которыми вроде бы никто уже не пользуется, можно оставить их отключенными до тех пор, пока кто-нибудь не пожалуется на их отсутствие. Иногда отсутствие таких компьютеров остается незамеченным в течение нескольких недель, а иногда о них вообще никто так никогда и не вспоминает.

🕮 Влияние на окружающую среду рассматривается в главе 28.

Системы охлаждения часто оборудованы датчиками температуры со средствами оповещения о ее повышении. Задайте такую верхнюю границу температуры, чтобы после сигнала у вас оставалось время выключить технику, прежде чем она перегреется и выйдет из строя; например, мы пользуемся отметкой в 24 градуса по Цельсию вместо 32, но живем в горах в сорока пяти минутах езды от работы (это летом, а зимой на то, чтобы добраться до места, может уходить и гораздо больше времени). В машинном зале лучше еще держать парочку обычных или работающих от батареи термометров, ведь при отключении электропитания все электронные индикаторы станут бесполезными.

Если какая-то часть оборудования устанавливается в удаленном месте, попросите отвечающие за обслуживание этого помещения службы показать резервные средства питания, прежде чем подписывать с ними договор. Удостоверьтесь в том, что генератор находится в рабочем состоянии и проверяется регулярно. Попросите разрешения присутствовать при следующей проверке генератора; независимо от того, разрешат вам это или нет, таким путем, скорее всего, удастся получить очень полезную информацию.

## Сетевая избыточность

Интернет-провайдеры часто поглощаются более крупными компаниями в результате слияния. Это сводит на нет их тщательно продуманные планы по поддержанию избыточных соединений с Интернетом. Поглощаемые интернет-провайдеры часто консолидируют электрические цепи, принадлежавшие независимым компаниям. В результате клиенты, у которых раньше были отдельные соединения с Интернетом, после этого могут получать их все по одному-единственному кабелю, но и, соответственно, быть отключенными от всех них сразу.

Интернет-провайдеры также нередко рекомендуют устанавливать "избыточные цепочки" или "резервные каналы", преимущества которых являются очень сомнительными. При более тщательном изучении может оказаться, что кабелей действительно два, но они находятся в одной и той же цепи или что по резервному каналу уже передается объемный ATM-трафик. Для уверенности в том, что избыточные каналы оправдывают себя, проводите со своими интернет-провайдерами ежегодную проверку.

## Проблемы с безопасностью

Об обеспечении безопасности системы подробно рассказывалось в главе 22. Однако здесь тоже стоит коснуться этой темы, потому что вопросы безопасности влияют на многие из выполняемых системным администратором задач. Ни один аспект стратегии управления организации не должен разрабатываться без учета безопасности. В главе 22, по большей части, рассказывалось о способах предотвращения проблем с безопасностью. Однако продумывание способов восстановления системы после происшедшего из-за связанных с безопасностью проблем инцидента является не менее важным.

Кража данных с веб-сайта относится к числу наиболее серьезных нарушений системы безопасности. Для системного администратора, работающего в компании, которая занимается предоставлением услуг веб-хостинга, такой инцидент может превратиться в настоящую катастрофу, особенно если речь идет о сайтах, использующих данные о кредитных карточках пользователей. Поступает поток телефонных звонков от клиентов, от средств массовой информации, от VIP-клиентов компании, которые только что услышали о происшедшей краже в новостях. Кто будет отвечать на эти звонки? Что он должен говорить? Кто будет главным? Кто что будет делать? Тем системным администраторам, которые работают в очень популярных компаниях, обязательно следует продумать такой сценарий, подготовить подходящие ответы и план действий и, возможно, даже провести тренировочную проверку, чтобы отработать детали.

Для сайтов, которые имеют дело с данными о кредитных карточках, всегда предусматриваются правила поведения в случае атаки с целью кражи информации. Поэтому системный администратор обязательно должен привлечь к планированию мероприятий на случай проблем с безопасностью сотрудников юридического отдела своей организации и позаботиться о наличии номеров телефонов и имен людей, которым следует звонить во время кризиса.

Когда в новостях объявляют, что такой-то веб-сайт был атакован хакерами, возникает ситуация, напоминающая аварию на дороге: все бросаются посмотреть, что же произошло, и в результате интернет-трафик сильно возрастает, нередко настолько сильно, что все, что администраторам наконец-то с таким трудом удалось исправить, снова может оказаться под угрозой. Поэтому, если веб-сайт не рассчитан на 25-процентное (или более высокое) увеличение трафика, следует позаботиться о наличии программ выравнивания нагрузки, которые будут просто направлять превышающие норму вызовы на сервер, а тот будет возвращать страницу со следующим сообщением: "Извините, сервер слишком загружен и поэтому в данный момент не может обработать ваш запрос".

# 32.9. Соответствие законам и стандартам

IT-аудит и управление в настоящее время являются очень популярными. Инструкции и квазистандарты для спецификации, измерений и сертификации соответствия законам породили множество аббревиатур: SOX, ITIL, COBIT и ISO 27002, и это только некоторые из них. К сожалению, эта мешанина букв оставляет у системных администраторов неприятный осадок, поэтому сейчас ощущается недостаток программного обеспечения, предназначенного для реализации всех средств управления, необходимых для обеспечения соответствия с действующим законодательством.

Некоторые из основных рекомендательных стандартов, руководящих принципов, промышленных концепций и законодательных требований, которые могут касаться системных администраторов, перечислены ниже. Законодательные требования являются в

значительной степени специфичными для Соединенных Штатов Америки. Однако эти стандарты действительно содержат полезную информацию даже для организаций, которые не обязаны их придерживаться. Некоторые из них стоит знать только для того, чтобы овладеть передовым опытом. (Стандарты перечислены в алфавитном порядке.)

- **CJIS (Информационные системы уголовного судопроизводства)** —стандарт, относящийся к организациям, которые отслеживают криминальную информацию и интегрируют ее с базами данных ФБР. Его требования могут быть найдены на странице `fbi.gov/hq/cjisd/cjis.htm`.
- **COBIT** — рекомендации для информационного управления, основанного на передовом промышленном опыте. Они разработаны совместно Ассоциацией аудита и управления информационными системами (Systems Audit and Control Association — ISACA) и Институтом информационного управления (IT Governance Instutute — ITGI); см. детали на сайте `isaca.org`. Задача рекомендаций COBIT состоит в том, чтобы "исследовать, развивать, предавать гласности и продвигать авторитетный, современный, международный набор общепринятых целей управления информационными технологиями для ежедневного использования менеджерами и аудиторами."

  Первая редакция этих рекомендаций была выпущена в 1996 году, сейчас действует версия 4.0, изданная в 2005 году. Последняя версия разрабатывалась под сильным влиянием требований закона Сарбейнза–Оксли (Sarbanes-Oxley). Она включает 34 цели высокого уровня, которые охватывают 215 "целей управления", классифицированных по четырем областям: планирование и организация, приобретение и реализация, поставка и поддержка, мониторинг и оценка.
- **COPPA (Children's Online Privacy Protection Act** — Закон о защите конфиденциальной информации о детях в Интернете); регулирует работу организаций, которые собирают или хранят информацию о детях, не достигших 13 лет. Для сбора определенной информации необходимо разрешение родителей; см. детали на сайте `coppa.org`.
- **FERPA (Family Educational Rights and Privacy Act** — Закон о правах семьи на образование и неприкосновенность частной жизни); относится ко всем учреждениям, являющимся получателями федеральной помощи, которой управляет министерство образования. Этот закон защищает информацию о студентах и предоставляет студентам определенные права относительно их данных; см. детали на сайте `ed.gov/policy/gen/guid/fpco/ferpa/index.html`.
- **FISMA (Federal Information Security Management Act** — Закон об управлении информационной безопасностью в федеральном правительстве); относится ко всем правительственным учреждениям и подрядчикам правительственных учреждений. Это большой и довольно неопределенный набор требований, которые нацелены на согласование со множеством публикаций об информационной безопасности, выпущенных институтом NIST, Национальным институтом по стандартизации и технологии (National Institute of Standards and Technology). Независимо от того, подпадает ли ваша организация под мандат FISMA или нет, документы NIST заслуживают внимания; см. `csrc.nist.gov/publications/PubsTC.html`.
- **Концепция Safe Harbor (правило безопасной гавани) Федеральной комиссии по торговле США (FTC)** представляет собой мост между подходами США и Европейского Союза к законодательству, защищающему частную жизнь, и определяет способ, с

помощью которого американские организации могут взаимодействовать с европейскими компаниями, чтобы продемонстрировать защиту своей информации; см. `export.gov/safeharbor/eg_main_018236.asp`.

- **Закон Грэма–Лич–Блайли (Gramm-Leach-Bliley Act — GLBA)** регулирует использование финансовыми учреждениями конфиденциальной информации потребителей. Если вы задавались вопросом, почему банки, выпускающие кредитные карточки, брокеры и страховщики забрасывали вас уведомлениями о конфиденциальности, то это следствие закона Грэма–Лич–Блайли; см. `ftc.gov/privacy/privacyinitiatives/glbact.html`.
- **Закон HIPAA (Health Insurance Portability and Accountability Act** — Закон об отчетности и безопасности медицинского страхования) относится к организациям, которые передают или хранят защищенную медицинскую информацию (иначе PHI). Это широкий стандарт, который был первоначально предназначен для борьбы с растратами, мошенничеством и злоупотреблениями в области здравоохранения и медицинского страхования, но теперь он используется для того, чтобы измерить качество и улучшить безопасность информации о здоровье; см. `hhs.gov/ocr/privacy/index.html`.
- **ISO 27001 и ISO 27002** — это рекомендательная (и информативная) коллекция передового опыта, связанного с безопасностью организаций, использующих информационные технологии; см. `iso.org`.
- **Библиотека ITIL (IT Insfrustructure Library)** — коллекция руководств, первоначально разработанных британским правительством, которые обрисовывают в общих чертах структуру управления информационными услугами. Это всего лишь рекомендации, но они широко используются; см. сайт `itil.org` и раздел, помещенный далее.
- **CIP (Critical Infrastructure Protection)** — семейство стандартов, разработанных корпорацией North American Electric Reliability Corporation (NERC), которые способствуют защите инфраструктурных систем, таких как электроснабжение, телефонные линии и финансовые сети, от стихийных бедствий и терроризма.

  В полном соответствии с учебной иллюстрацией ницшеанского понятия "жажды власти" оказывается, что большая часть экономики попадает в один из 17 секторов "критических инфраструктур и ключевых ресурсов" (CI/KR) и поэтому очень нуждается в стандартах CIP. Организации, попадающие в эти сектора, должны оценить свои системы и защищать их соответствующим образом; см. `cip.gmu.edu/cip`.
- **Стандарт PCI DSS** (Payment Card Industry data Security Standard — Стандарт защиты информации в индустрии платежных карточек) был создан консорциумом платежных брендов, включая American Express, Discover, MasterCard и Visa. Он охватывает вопросы управления данными о платежной карточке и относится к любой организации, которая принимает платежи по кредитной карточке. Стандарт имеет два варианта: самооценку для небольших организаций и внешний аудит для организаций, которые обрабатывают много сделок; см. `pcisecuritystandards.org`.
- **Правила Red Flags Rules** Федеральной Комиссии по торговле США требуют, чтобы любой, кто расширяет кредит на потребителей (т.е. любая организация, которая отсылает счета), осуществлял формальную программу, предотвращающую и обнаруживающую "хищение персональных данных". Правила требуют, чтобы эмитенты кредитов разработали эвристические правила для того, чтобы идентифициро-

вать подозрительные манипуляции со счетами. Для выяснения деталей наберите фразу "red flag" в поисковой строке на сайте `ftc.gov`.

- **Раздел IT General Controls (ITGC) в законе Сарбейнза-Оксли** (SOX), последний по расположению, но не по важности, относится ко всем акционерным обществам и разработан, чтобы защитить акционеров от бухгалтерских ошибок и мошеннических методов; см. `sec.gov/rules/final/33-8124.htm`.

## Библиотека ITIL: Information Technology Infrastructure Library

Среди этих стандартов библиотека Information Technology Infrastructure Library (ITIL) стала фактическим стандартом для организаций, ищущих полноценное решение для управления информационными услугами. Процессы ITIL разделены на шесть групп.

- **Служба поддержки** – информационные услуги для клиентов, а также прием запросов и сообщений о проблемах; также включает условия для отслеживания и эскалации проблем.
- **Управление инцидентами** — восстановление обслуживания после инцидента, вызвавшего сбой.
- **Управление проблемами** — идентифицирует причины инцидентов, чтобы предотвратить будущие сбои.
- **Управление конфигурацией** — инкапсулирует информацию о компонентах инфраструктуры и их взаимозависимостях.
- **Управление изменениями** — процессы управления изменениями в пределах инфраструктуры.
- **Управление выпусками** — процесс, аналогичный процессу управления изменениями, но используемый для крупномасштабных изменений в пределах организации.

У крупных организаций может быть формальная программа ITIL, дополненная системой отслеживания запросов, которая точно отражает концепции ITIL и ее определения. Однако даже небольшая организация, использующая общедоступную систему отслеживания запросов, может принять политику, которая поощряет внимательное управление изменениями. Все изменения должны сопровождаться подачей заявки на изменение, одобрения комитета по изменениям и отслеживаться с помощью диагностической системы. Все инциденты должны обрабатываться в соответствии с установленным процессом реагирования, который включает анализ состояния после устранения последствий инцидента, чтобы определить, была ли проблема решена самым лучшим способом.

В общем, следует интерпретировать рекомендации в свете определенных потребностей вашей организации. Главная цель состоит в том, чтобы понять концепции, воплощенные в стандартах, и принять их философию. Некоторые из упомянутых выше стандартов состоят из сотен страниц, поэтому их даже трудно просто прочитать; не стесняйтесь резюме и сжатых версий.

## NIST: Национальный Институт стандартов и технологии

Институт NIST издает множество стандартов, полезных для администраторов и технологов. Некоторые из популярных стандартов упомянуты ниже, но если вам скучно и вы ищете стандарты, можете зайти на его веб-сайт. Вы не будете разочарованы.

*Стандарт NIST 800-53* (Рекомендуемые средства управления системами безопасности для федеральных информационных систем и организаций) описывает, как оценить безопасность информационных систем. Если ваша организация разработала внутреннее приложение, которое хранит конфиденциальную информацию, этот стандарт может помочь вам удостовериться, что вы действительно защитили ее. Однако остерегайтесь: процедура согласования со стандартом NIST 800-53 не для слабонервных. Вы, вероятно, столкнетесь с документом объемом около 100 страниц со множеством мучительных деталей.[3]

*Стандарт NIST 800-34* (Принципы планирования на случай непредвиденных ситуаций для информационных систем) является библией аварийного восстановления, разработанной организацией NIST. Он предназначен для правительственных учреждений, но любая организация может извлечь из него выгоду. Следование стандарту планирования NIST 800-34 требует времени, но это вынуждает вас ответить на такие важные вопросы, как: "Какие системы являются самыми важными?", "Сколько времени мы можем обойтись без этих систем?" и "Как мы собираемся восстанавливаться, если наш основной информационный центр потерян?"

# 32.10. Правовые вопросы

Правительство США и некоторые штаты издали законы о преступлениях в области вычислительной техники. На федеральном уровне с начала 1990-х годов существовало два закона, а теперь их четыре.

- Федеральный Закон о конфиденциальности связи.
- Закон о компьютерном мошенничестве и компьютерных злоупотреблениях.
- Закон о компьютерных кражах.
- Закон об авторских правах "Digital Millenium Copyright Act".

Основными правовыми проблемами в настоящее время являются ответственность системных администраторов, сетевых операторов и провайдеров, предоставляющих услуги веб-хостинга; сети для обмена файлами между пользователями, защита авторских прав и конфиденциальность. В разделе рассматриваются эти и другие юридические вопросы, связанные с системным администрированием.

## Конфиденциальность

Частную жизнь всегда было трудно скрыть, но с развитием Интернета она подвергается еще большей опасности, чем когда-либо. Медицинская документация неоднократно раскрывалась с помощью плохо защищенных систем, украденных ноутбуков и резервных магнитных лент, которые находились не в положенном месте. Базы данных, номера кредитных карточек всегда подвергались нападениям. Веб-сайты, предлагающие антивирусное программное обеспечение, фактически сами устанавливают программы-шпионы. Поддельная электронная почта поступает почти ежедневно в виде фиктивных писем от вашего банка, в которых утверждается, что у вас возникли проблемы с вашим счетом и требуется, чтобы вы проверили свои учетные данные. Обычно тщательное изучение электронной почты показывает, что эти данные попали бы к хакеру в Восточной Европе или Азии, а не в ваш банк. Этот тип нападения называют фишингом.

[3] Если вы планируете сотрудничество с правительственными учреждениями США, от вас могут потребовать соответствия стандарту NIST 800-53, хотите вы этого или нет...

Технические меры никогда не смогут защитить от этих нападений, потому что они нацелены на самый уязвимый компонент — ваших пользователей. Ваша лучшая защита — образованные пользователи. Никакая легальная служба электронной почты или веб-сайт никогда не будут сообщать о следующем:

- что вы выиграли приз;
- чтобы вы "проверили" информацию о счете или пароли;
- чтобы вы отправили часть электронной почты;
- чтобы вы установили программное обеспечение, которое вы не искали;
- о вирусе или другой проблеме безопасности.

Пользователи, осведомленные о таких опасностях, с большей вероятностью сделают правильный выбор, когда всплывающее окно будет утверждать, что они выиграли бесплатный MacBook.

## Реализация стратегии

Файлы системного журнала могут легко доказать вам, что человек X сделал плохо человеку Y, но для суда это всего лишь слова. Защитите себя письменными предписаниями. Файлы системного журнала иногда содержат метки времени, которые полезны, но не обязательно допустимы как доказательства, если ваш компьютер не управляет протоколом NTP (Network Time Protocol), чтобы синхронизировать свои часы со стандартными.

Вам, возможно, понадобится инструкция по безопасности, чтобы преследовать кого-то по суду за неправильный поступок. Она должна содержать утверждение: "Несанкционированное использование вычислительных систем может повлечь не только нарушение организационной политики, но и нарушение законов штата и федеральных законов. Несанкционированное использование — это преступление, которое может повлечь уголовное наказание и гражданско-правовые санкции; оно будет преследоваться по суду в полном объеме закона."

Мы советуем показывать заставку, которая сообщает пользователям о ваших правилах слежения. Вы могли бы написать что-то вроде следующего: "Действие может отслеживаться в случае реального или подозреваемого нарушения правил безопасности".

Вы можете гарантировать, что пользователи видят уведомление, по крайней мере, один раз включив его в файлы запуска, которые вы предоставляете новым пользователям. Если вы требуете, чтобы использование протокола SSH регистрировалось (а вы должны это требовать), можете сформировать файл `/etc/ssh/sshd_config` так, чтобы протокол SSH всегда показывал заставку.

Убедитесь, что, пользуясь своими учетными записями, пользователи признают вашу письменную инструкцию. Объясните, где пользователи могут получить дополнительные копии стратегических документов и куда отправлять ключевые документы о соответствующей веб-странице. Также включайте определенный штраф за несоблюдение правил (удаление учетной записи и т.д.). Более важно, чтобы вы добросовестно уведомили пользователей об их обязанностях, а не просто составили юридически точное уведомление.

Кроме заставки, целесообразно сделать так, чтобы пользователи подписали стратегическое соглашение, прежде чем получить доступ к вашим системам. Это приемлемое пользовательское соглашение должно быть разработано в сотрудничестве с вашим юридическим отделом. Если у вас нет подписанных соглашений со служащими, соберите их. Сделайте подписание соглашения стандартной частью процесса приема на работу новых сотрудников.

Вы могли бы также периодически проводить семинары по вопросам безопасности. Это удобная возможность рассказать пользователям о важных проблемах, таких как фишинг, научить их правильно устанавливать программное обеспечение и пароли, а также кое-чему другому, что относится к вашей компетенции.

## Контроль — это ответственность

Поставщики интернет-услуг обычно следуют правилам пользования сетью (appropriate use policy — AUP), которые диктуют стоящие над ними поставщики и которые являются обязательными для их клиентов. Таким образом, вся ответственность за действия клиентов ложится на самих клиентов, а не на провайдеров. Целью такой стратегии является защита провайдеров от ответственности за рассылку спама и прочие незаконные действия, например хранение пользователями на своих узлах незаконного или защищенного авторскими правами материала. В каждом штате и каждой стране имеются свои законы на этот счет.

Ваши правила должны явно запрещать пользователям использовать ресурсы компании для незаконной деятельности. Однако на самом деле этого недостаточно — вы также должны дисциплинировать пользователей, обнаружив, что они осуществляют незаконные действия. Организации, которые знают о незаконных действиях, но не принимают меры для их пресечения, считаются соучастниками и могут преследоваться по закону. Нет ничего хуже несоблюдаемых или противоречащих друг другу правил, как с практической, так и юридической точки зрения.

Из-за риска стать соучастником незаконных действий пользователя, некоторые организации ограничивают данные, которые они регистрируют, отрезок времени, в течение которого сохраняются файлы системного журнала, и объем информации о файлах системного журнала, хранящейся на резервных лентах. Некоторые пакеты программ помогают выполнять эти правила, устанавливая уровни регистрации. Это помогает системным администраторам устранять проблемы, не вторгаясь в частную жизнь пользователей. Тем не менее всегда следует знать, какой вид регистрации требуется по местным законам или по любым регулирующим стандартам, которые распространяются на вашу организацию.

## Лицензии на программное обеспечение

Многие компании оплачивают меньшее количество копий программ, чем используют на самом деле. Если об этом становится известно, компания теряет гораздо больше, чем сэкономила на приобретении недостающего числа лицензий. Другие компании получают демоверсию дорогого пакета и взламывают ее (меняют дату на компьютере, определяют лицензионный ключ и так далее), чтобы пакет продолжал работать по истечении демонстрационного срока. Как системный администратор должен реагировать на предложения нарушить лицензионное соглашение и установить нелицензионные копии программы на дополнительные компьютеры? Что он должен делать, если обнаруживается, что на обслуживаемых им компьютерах работает пиратское программное обеспечение? И как быть с условно-бесплатными программами, за которые так никогда и не заплатили?

Это очень сложный вопрос. К сожалению, начальство не всегда поддерживает администратора, предлагающего удалить нелицензионные копии программ либо оплатить их. А ведь часто именно системный администратор подписывает лицензионное соглашение, требующее удалить демонстрационные копии после определенной даты, тогда как решение их не удалять принимает руководитель.

Нам известно несколько случаев, когда непосредственный начальник системного администратора предлагал ему "не раскачивать лодку". Тогда администраторы написали докладную вышестоящему руководству, в которой указали количество лицензированных и используемых копий, а также процитировали лицензионное соглашение. В одном случае подход сработал, и уволен был непосредственный начальник системного администратора. Во втором случае уволиться пришлось системному администратору, потому что даже вышестоящее руководство отказалось действовать по закону. Чтобы вы ни делали в такой ситуации, обязательно делайте это в письменном виде. Требуйте, чтобы ответы предоставлялись в письменном виде, а если не получится, документируйте полученные инструкции в виде коротких служебных записок и отправляйте их ответственному лицу.

## 32.11. Организации, конференции и другие ресурсы

Многие группы поддержки систем UNIX и Linux — и общего характера, и конкретных поставщиков — помогают устанавливать контакты с другими людьми, использующими то же программное обеспечение. В табл. 32.3 приведен короткий список организаций, но большинство национальных и региональных групп в этой таблице не упомянуто.

Организация FSF (Free Software Foundation — Фонд бесплатного программного обеспечения) является спонсором проекта GNU (GNU — аббревиатура от "GNU is Not Unix"; так называется проект по свободному распространению программного обеспечения). Под словосочетанием "бесплатное программное обеспечение" в названии этой организации подразумевается программное обеспечение, которое не имеет почти никаких ограничений в использовании, а не бесплатное программное обеспечение. Также организация FSF является автором лицензии GPL, которая распространяется на большую часть систем UNIX и Linux.

Организация USENIX, представляющая собой союз пользователей Linux, UNIX и других операционных систем с открытым исходным кодом, каждый год проводит одну общую конференцию и несколько специализированных (небольших). На общей конференции обычно речь идет об открытых системах и новых разработках, которые проводятся в рамках Linux и BSD.

**Таблица 32.3. Организации, ориентированные на системных администраторов систем UNIX и Linux**

| Название | URL | Описание |
|---|---|---|
| FSF | `fsf.org` | Free Software Foundation, спонсор GNU |
| USENIX | `usenix.org` | Группа пользователей UNIX, обсуждающих технические вопросы |
| SAGE | `sage.org` | Гильдия System Administrators Guild, ассоциированная с организацией USENIX; проводит ежегодные конференции LISA |
| LOPSA | `lopsa.org` | League of Professional System Administrators — лига профессиональных системных администраторов, "отколовшаяся" от организации USENIX/SAGE |
| SANS | `sans.org` | Проводит конференции по системному администрированию и безопасности; обсуждает менее технические вопросы по сравнению с гильдией SAGE, делая упор на обучении |
| The Linux Foundation | `linuxfoundation.org` | Некоммерческий консорциум, ориентированный на стимулирование роста системы Linux |

*Окончание табл. 32.3*

| Название | URL | Описание |
|---|---|---|
| AUUG | `auug.org.au` | Australian UNIX Users Group; охватывает технические и управленческие аспекты вычислений |
| SAGE-AU | `sage-au.org.au` | Australian SAGE; проводит ежегодные конференции в Австралии |
| SANE | `sane.nl` | Группа System Administration and Network Engineering; проводит ежегодные конференции в Европе |

Важным событием для системных администраторов является конференция LISA (Large Installation System Administration — системное администрирование в крупных организациях), которая проводится поздней осенью. Помимо всех этих конференций, часто проводятся соответствующие торговые выставки.

Кроме того, в течение нескольких последних лет организация USENIX специально для сообщества Linux проводит отдельную конференцию по вопросам разработки ядра Linux. Попасть на это двухдневное мероприятие можно только по приглашению.

SAGE (USENIX System Administration Guild — гильдия системных администраторов USENIX) является первой международной организацией для системных администраторов. Она рекламирует системное администрирование как профессию путем финансирования различных конференций и неофициальных программ. Подробнее о ней можно узнать на сайте `www.sage.org`.

Гильдия SAGE совместно со своей головной организацией USENIX проводит конференции, посвященные управлению системами и сетями, учебные и технические семинары, лекции и презентации. Иногда она также проводит однодневные рабочие семинары, посвященные специальным темам. См. информацию на сайте `usenix.org`.

Бюллетень организаций USENIX и SAGE —;login: — выпускается обеими организациями; он содержит административные новости, советы, обзоры и объявления, представляющие интерес для системных администраторов. Гильдия SAGE поддерживает список ресурсов для системных администраторов. Свежую информацию можно найти на сайте `sage.org`.

В 2005 году раскол организаций USENIX и SAGE поставил будущее гильдии SAGE под сомнение. В результате этого раскола появилась новая отдельная организация под названием LOPSA (League of Professional System Administrators — Лига профессиональных системных администраторов), основанная бывшими сотрудниками организации SAGE. До начала 2010 года организация LOPSA еще не проводила никаких конференций. У SAGE была программа по сертификации системных администраторов, но она от нее отказалась; будем надеяться, что LOPSA вернется к ней.

Институт SANS проводит много курсов обучения и семинаров по вопросам безопасности и поддерживает программу по сертификации. Экзамен проводится в течение ограниченного времени в форме многовариантного теста, вопросы которого известны заранее. Прежде чем сдать реальный экзамен, испытуемые могут попробовать сдать два тренировочных экзамена. Индивидуальные сертификаты относятся к узкоспециализированным вопросам; соискатели могут также получить общий сертификат по безопасности (GSEC). Сертификат действует только 2-3 года, поэтому вы должны следить за новинками и повторно проходить сертификацию (за дополнительную плату). Детали см. на сайте `giac.org`.

Существует множество групп пользователей UNIX, Linux и других открытых систем. Некоторые из них связаны с организацией USENIX, а другие нет. Локальные группы обычно проводят регулярные встречи и рабочие семинары с местными или приезжими

лекторами и часто организовывают банкеты до или после мероприятия. Это хороший способ поддерживать контакты с системными администраторами в своем регионе.

Главной торговой выставкой в сфере сетевой индустрии является выставка Interop. Проводимые в ее рамках обучающие семинары известны своим высоким качеством. Когда-то выставка Interop проходила только раз в год; этого события с нетерпением ждали как технические специалисты, так и производители. Сейчас она проводится несколько раз в год, можно сказать, представляя собой "бродячий цирк сетевых технологий". Зарплата преподавателей семинаров сократилась вполовину, но качество семинаров вроде бы от этого не пострадало.

## 32.12. Рекомендуемая литература


- Limoncelli T.A. *Time management for system administrators*. Sebastopol, CA: O'reilly Media, 2005.
- Machiavelli Niccolo. *The prince*. 1513. Available on-line from `gutenberg.org/etext/1232`.
- Brooks F.P., Jr. *The mythical man-month: essays on software engineering*. Reading, MA: Addison-Wesley, 1995.
- Senft S., Gallegos G. *Information technology control and audit (3rd edition)*. Boca Raton, FL: Auerbach Publications, 2008.


- Сайт `itil-toolkit.com` удобен для начала поисков, если вы хотите овладеть терминологией и жаргоном, связанными с процессами и стандартами ITIL.
- Сайт `itl.nist.gov` — домашняя страница организации NIST Information Technology Laboratory. Она содержит много информации о стандартах. Зайдите на страницы с публикациями.
- Веб-сайт организации Electronic Frontier Foundation, `eff.org`, — прекрасное место для поиска комментариев, касающихся новейших достижений в области защиты конфиденциальности, криптографии и законодательства. Эти материалы всегда интересны.
- Страница `sans.org/resources/policies` содержит проект инструкции SANS о безопасности. На этом сайте можно найти хорошие примеры инструкций по информационным технологиям.
- Большое количество отличных ресурсов для системных администраторов можно найти на сайте гильдии SAGE: `sage.org/field/field.html`.

## 32.13. Упражнения

32.1. Назовите повторяющиеся процедуры, выполняемые в вашей организации? Какие из них повторяются редко и каждый раз изменяются? Какие из них связаны с риском?

32.2. Как вы зависите от внешних поставщиков? Нуждаетесь ли вы в плане Б и есть ли он у вас? Объясните, почему. Опишите план Б, если он существует.

32.3. Проведите краткие интервью с несколькими внутренними клиентами, чтобы определить их ожидания относительно доступности вычислительной инфраструктуры.

Согласованы ли эти ожидания? Действительно ли они разумны? Действительно ли они совместимы с установленными целями группы системного администрирования?

32.4. Какая организованная инфраструктура для системного управления уже установлена в вашей организации? Идентифицируйте части, которые все еще отсутствуют.

32.5. Один из ваших сотрудников собирается уехать завтра в обед и никогда возвращаться, но вы еще не знаете кто. (Нет, не пытайтесь угадать.) На какие критические процедуры это могло бы повлиять и насколько готова ваша организация восполнить недостаток сотрудника? Какая документация могла бы помочь избежать разрушения службы?

32.6. Что произошло бы, если бы вы не ходили на работу в течение следующих трех месяцев? Насколько ваши коллеги ненавидели бы вас, когда вы наконец вернулись, и почему? Что вы можете сделать за следующие две недели, чтобы уменьшить ущерб от своего отсутствия?

32.7. ★ Босс приказывает вам сократить бюджетные ассигнования на системное администрирование на 30% до конца текущего года. Вы можете оценить последствия этого сокращения? Представьте резюме, которое позволит боссу принять обоснованное решение относительно того, какие службы сократить или ликвидировать.

32.8. ★ Кто в вашей корпорации поддерживает систему Linux? Каковы их интересы и побуждения? Какие вклады они делают?

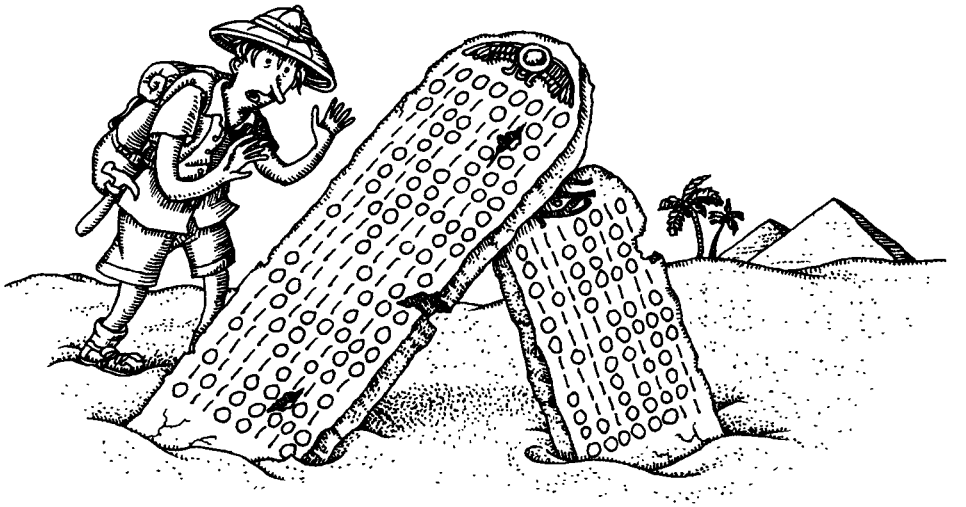
32.9. ★ Вы наводите порядок после сбоя диска и замечаете файлы в каталоге **lost+found**. Проводя дальнейшее расследование, вы обнаруживаете, что некоторые файлы представляют собой почтовые сообщения, которые послали друг другу два студента, настраивающие черный ход вокруг брандмауэра отдела, чтобы заархивировать файлы MP3 на отдаленном файловом сервере. Что вы должны сделать? Есть ли правила или инструкции в вашей организации, которые касаются таких инцидентов?

32.10. ★★ Оцените локальную документацию вашей организации, предназначенную для новых пользователей, системных администраторов, стандартных процедур и чрезвычайных ситуаций.

32.11. ★★ Опишите будущее различных коммерческих и свободных вариантов UNIX и разновидностей Linux на следующие пять лет. Как будут развиваться текущие модели разработки и распределения в течение долгого времени? Каковы будут долгосрочные последствия принятия системы Linux продавцами аппаратных средств? Проведите различие между рынками серверов и настольных систем.

# Краткая история системного администрирования

*Из записок доктора Питера Салуса (Peter H. Salus), историка, занимающегося вопросами развития технологий*



В современную эпоху большинство людей имеют хотя бы минимальное представление о задачах, решаемых системными администраторами: они должны неустанно заботиться об удовлетворении потребностей пользователей и организаций, а также заниматься проектированием и реализацией устойчивой к ошибкам вычислительной среды, стараясь при этом поражать своих клиентов эффективными решениями. Несмотря на то что системные администраторы часто считаются низкооплачиваемыми и недооцененными, большинство пользователей могут хотя бы назвать имя своего местного "сисадмина" — причем во многих случаях гораздо быстрее, чем имя начальника их начальника.

Но так было не всегда. За последние 40 лет (и в течение 20-летней истории этой книги) роль системного администратора постепенно эволюционировала вместе с операционными системами (ОС) UNIX и Linux. Полное постижение предназначения системного администратора потребует понимания того, каким путем мы пришли к нынешнему состоянию, а также понимания некоторых исторических факторов, которые сформировали наш ОС-ландшафт. Итак, окинем взглядом несколько чудесных десятилетий. Присоединяйтесь!

## Рассвет компьютеризации: системные операторы (1952–1960)

Первый коммерческий компьютер, IBM 701, был выпущен в 1952 году. До него все компьютеры выпускались в единичных экземплярах. В 1954 году модернизированная версия IBM 701 была анонсирована как IBM 704. Она имела оперативную память на магнитных сердечниках объемом 4 096 слов и включала три индексных регистра. Этот компьютер использовал 36-разрядные слова (в отличие от 18-разрядных слов у IBM 701)

и выполнял арифметические операции с плавающей запятой. Он работал со скоростью 40 000 инструкций в секунду.

Но компьютер IBM 704 был несовместим с IBM 701. Несмотря на то что его поставки должны были начаться к концу 1955 года, операторы (предшественники современных системных администраторов) восемнадцати существующих компьютеров IBM 701 уже нервничали: как им пережить эту "модернизацию" и на какие еще подводные камни они могут наткнуться?

Что касается самой компании IBM, то ее специалисты не имели решения проблемы модернизации и совместимости. В августе 1952 года для пользователей IBM 701 компания провела "курсы повышения квалификации", но учебников не выпустила. Некоторые слушатели этих "курсов" продолжали неформально встречаться и обсуждать свой опыт использования системы. Компания IBM поощряла встречи операторов, поскольку совместное рассмотрение проблем помогало общими усилиями найти их решение. IBM финансировала встречи операторов и предоставила их участникам доступ к библиотеке, включающей 300 компьютерных программ. Эта группа по обмену информацией, именуемая SHARE, все еще существует (вот уже 50 с лишним лет)[1].

## От узкого назначения к работе с разделением времени (1961–1969)

Первые образцы компьютерного оборудования были достаточно громоздкими и чрезвычайно дорогими. Это наводило покупателей на мысль о том, что их компьютерные системы предназначены для решения одной-единственной конкретной задачи: только достаточно крупная и достаточно конкретная задача могла оправдать дороговизну и громоздкость такого компьютера.

Если компьютер был инструментом специального назначения (таким, как, например, пила), то персонал, который обслуживал компьютер, можно назвать операторами такой "пилы". К первым системным операторам относились скорее как к "тем, кто пилит древесину", чем как к "тем, кто обеспечивает все необходимое для строительства здания". Переход от системного оператора к системному администратору начался тогда, когда компьютеры превратились в универсальные (многоцелевые) инструменты. Основной причиной изменения точки зрения на компьютер стало то, что его начали использовать в режиме разделения времени.

Джон Мак-Карти (John McCarthy) начал подумывать о режиме разделения времени в средине 1950-х годов. Но это было только в Массачусетсском технологическом институте (MIT) в 1961-1962 годах, когда Джон Мак-Карти, Джек Дэннис (Jack Dennis) и Фернандо Корбато (Fernando Corbato) серьезно говорили о том, чтобы разрешить "каждому пользователю компьютера вести себя так, будто он — единственный его пользователь".

В 1964 году MIT, General Electric и Bell Labs объединили свои усилия по созданию проекта по построению претенциозной системы, работающей в режиме разделения времени. Эта система получила название Multics (Multiplexed Information и Computing Service). Пять лет спустя проект Multics превысил бюджет и безнадежно отстал от графика работ. В результате компания Bell Labs вышла из проекта.

[1] Несмотря на то что группа SHARE изначально спонсировалась производителями вычислительной техники, в настоящее время она является независимой организацией.

## Рождение UNIX (1969–1973)

Отказ компании Bell Labs от участия в проекте Multics оставил нескольких научных работников в Мюрей Хилл (штат Нью-Джерси) без работы. Трое из них — Кен Томпсон (Ken Thompson), Руд Кенедей (Rudd Canaday) и Дэннис Ричи (Dennis Ritchie) — заинтересовались некоторыми аспектами проекта Multics, но не были в восторге от размера и сложности этой системы. Они не раз собирались вместе обсудить принципы проектирования компьютерных систем. Компания Labs запустила систему Multics на своем компьютере GE-645, и Кен Томпсон продолжил работу над этим проектом "шутки ради." Дуг МакИлрой (Doug McIlroy), руководитель этой группы, сказал: "Система Multics впервые заработала именно здесь. Вдохнуть в нее жизнь удалось трем нашим сотрудникам".

Летом 1969 года Томпсон на месяц стал холостяком, пока его жена, Бонни, взяв их годовалого сына, уехала повидать родственников на западное побережье. Томпсон вспоминал: "Я выделил по неделе на операционную систему, интерпретатор команд, редактор и ассемблер...и все эти компоненты были полностью переписаны в форме, которая придавала им вид операционной системы (с набором известных утилит, ассемблером, редактором, интерпретатором команд), которая, если не поддерживала себя сама, была практически на грани такой поддержки, т.е. больше совершенно не нуждалась в GECOS[2]".

Стив Борн (Steve Bourne), пришедший работать в Bell Labs в следующем году, так описывал "заброшенный" компьютер PDP-7, которым воспользовались Ричи и Томпсон: "В PDP-7 были только ассемблер и загрузчик. На этом компьютере мог работать лишь один пользователь. Среда еще "отдавала сыростью", хотя части однопользовательской системы UNIX уже были "на подходе"... И вот были написаны ассемблер и зачаточное ядро операционной системы, которые еще требовали использования кросс-ассемблера для трансляции программы на машине PDP-7 с системой GECOS. Название UNICS (UNiplexed Information and Computing Service) для "новорожденной" операционной системы придумал заядлый остряк Питер Нейман (Peter Neumann) в 1970 году". Первоначально UNIX была однопользовательской системой, отсюда, повидимому, и намек на "урезанный вариант Multics". И хотя некоторыми аспектами система UNICS/UNIX все же обязана своей предшественнице Multics, у них, по словам Дэнниса Ричи, были "серьезные различия".

"Мы были немного подавлены большим менталитетом системы, — сказал он. — Кен хотел создать что-то простое. Вероятно, из-за того, что наши возможности были тогда довольно невелики. Мы могли достать только маленькие машины без каких-либо аппаратных средств Multics. Поэтому UNIX нельзя назвать ответным ударом, направленным против Multics... Операционная система Multics больше не существовала для нас, но нам нравилось ощущение интерактивной работы на компьютере, которое она предлагала пользователю. У Кена были некоторые идеи по созданию новой системы... И хотя Multics повлияла на подходы к реализации UNIX, это влияние не было определяющим."

"Игрушечная" система Кена и Дэнниса не долго оставалась "простой". К 1971 году в число команд пользователя входили следующие: `as` (ассемблер), `cal` (простая утилита-календарь), `cat` (конкатенация и вывод), `chdir` (изменение рабочего каталога), `chmod` (изменение режима), `chown` (изменение владельца), `cmp` (сравнение двух файлов), `cp`

[2] GECOS (General Electric Comprehensive Operating System) — операционная система, разработанная компанией General Electric в 1962 г.

(копирование файла), `date`, `dc` (калькулятор), `du` (отчет об использовании дискового пространства), `ed` (редактор) и еще два десятка других команд. Большинство из этих команд все еще находятся в употреблении.

К февралю 1973 года можно было говорить о существовании 16 инсталляций UNIX, а также о двух больших нововведениях. Первое связано с "новым" языком программирования C, основанным на языке B, который сам представлял собой "урезанную" версию языка BCPL (Basic Combined Programming Language), созданного Мартином Ричардсоном (Martin Richards). Второе нововведение — понятие канала.

Канал, попросту говоря, — это стандартный способ связи выходных данных одной программы с входными данными другой. Среда Dartmouth Time-Sharing System имела файлы связи, которые предвосхитили каналы, но их использование было гораздо более узким. Идея каналов (как обобщенного средства передачи данных) была предложена Дугом Макилроем (Doug McIlroy), а реализована — Кеном Томпсоном благодаря настойчивости МакИлроя. ("Это был один из немногочисленных случаев, когда я, по сути, осуществлял административное управление над UNIX", — сказал Дуг.)

"Легко сказать: '`cat` — в `grep` — в...' или '`who` — в `cat` — в `grep`' и так далее, — както заметил МакИлрой. — Это легко сказать, и было ясно с самого начала, что именно это вы и хотели бы сказать. Но еще существуют все эти параметры... И время от времени я говорил: 'Как бы нам сделать что-то вроде этого?' И вот в один прекрасный день я пришел с синтаксисом для командного языка, который развивался вместе конвейерной пересылкой данных, и Кен сказал: 'Я готов сделать это!' "

Применив множество вариантов, Томпсон обновил все UNIX-программы за одну ночь. Это было настоящее начало власти UNIX, возникшее не из отдельных программ, а из взаимосвязей между ними. Теперь у UNIX были собственные язык и основополагающие принципы.

- пишем программы, которые бы выполняли одну задачу, но выполняли ее хорошо;
- пишем программы, которые бы работали вместе;
- пишем программы, которые обрабатывали бы текстовые потоки как универсальный интерфейс.

Итак, можно было говорить о "рождении" операционной системы общего назначения с разделением времени, но пока лишь в "недрах" фирмы Bell Labs. ОС UNIX обещала простое и незаметное разделение компьютерных ресурсов между проектами, группами и организациями. Но прежде чем этот универсальный инструмент стал достоянием всего мира, он должен был вырваться из собственной колыбели и "размножиться".

## UNIX становится знаменитой (1974–1990)

В октябре 1973 года международная научно-образовательная Ассоциация по вычислительной технике (Association for Computing Machinery — ACM) провела симпозиум на тему "Принципы построения операционных систем" (Symposium on Operating Systems Principles — SOSP) в аудитории Центра исследований (T.J. Watson Research Center) компании IBM в Йорктаун Хайтс (Yorktown Heights), штат Нью-Йорк. Кен и Дэннис представили на рассмотрение свой доклад и в один прекрасный осенний день поехали в Долину Гудзона (Hudson Valley), чтобы представить его. (Томпсон сделал настоящую презентацию.) В зале было около 200 человек, и доклад произвел фурор.

Шесть месяцев спустя количество инсталляций UNIX утроилось. Когда этот доклад был опубликован в июльском (1974) выпуске журнала "Communications of the ACM", реакция читателей была просто ошеломляющей. Научно-исследовательские лаборатории

и университеты увидели в разделяемых UNIX-системах потенциальное решение проблемы их постоянно возрастающих потребностей в компьютерных ресурсах.

В соответствии с положениями антитрестовского соглашения 1958 года, подписанного корпорацией AT&T (из недр которой выделилась фирма Bell Labs) с федеральным правительством, ее деятельность была весьма ограничена: она не имела право заниматься рекламой, продажей и сопровождением компьютерных продуктов. Компания Bell Labs должна была давать разрешение на использование другими своей технологии. Тем не менее система UNIX завоевала популярность в мире, особенно среди телефонных компаний. Отвечая на многочисленные просьбы, Кен Томпсон стал распространять копии исходного кода UNIX, и, по легенде, каждый пакет включал его личную записку, подписанную "love, ken" (с любовью, Кен).

Одним из тех, кто получил копию системы от Кена, был профессор Роберт Фабри (Robert Fabry) Калифорнийского университета в Беркли. Так, к январю 1974 года зерно Berkeley UNIX попало в плодородную почву.

Другие ученые, работающие в области компьютерных наук, также проявили интерес к UNIX. В 1976 году Джон Лайонс (John Lions), преподаватель факультета компьютерных наук в Университете Нового Южного Уэльса в Австралии, опубликовал подробные комментарии относительно ядра версии V6. Эта работа стала первым серьезным пакетом документации по системе UNIX и помогла другим понять и развить работу Кена и Дэнниса.

Студенты Калифорнийского университета в Беркли поработали над версией UNIX, полученной из Bell Labs, и изменили ее так, чтобы она отвечала их требованиям. Первый ленточный дистрибутив программы Беркли (1st Berkeley Software Distribution — 1BSD) содержал систему Pascal для Unix и текстовый редактор ex для компьютера PDP-11. Этот дистрибутив подготовил аспирант Билл Джой (Bill Joy). Второй выпуск (2BSD) вышел в следующем году, а третий (3BSD) — в качестве первого дистрибутива программы Беркли для мини-компьютера VAX, выпускаемого корпорацией DEC, увидел свет в конце 1979 года.

В 1980 году профессор Роберт Фабри заключил контракт с управлением перспективных исследовательских программ в области обороны (Defense Advanced Research Project Agency — DARPA) на продолжение разработки системы UNIX. Результатом этого соглашения стало образование в Беркли исследовательской группы по компьютерным системам (Computer Systems Research Group — CSRG). В конце следующего года вышла четвертая версия системы — 4BSD. Она приобрела довольно широкую популярность, в основном, потому, что это была единственная версия UNIX, которая работала на DEC VAX 11/750, весьма распространенной в то время компьютерной платформе. Еще одно крупное усовершенствование, отличавшее выпуск 4BSD, состояло в использовании сокетов TCP/IP, обобщенной сетевой абстракции, которая в будущем породила Интернет и сейчас используется многими современными операционными системами. К средине 1980-х годов в большинстве крупных университетов и научно-исследовательских институтов была установлена хотя бы одна система UNIX.

В феврале 1982 года в городе Санта-Клара, что расположен в Кремниевой Долине, была основана компания Sun Microsystems. Так вышло, что Билл Джой присоединился к SUN (сейчас это часть компании Oracle America) практически в момент зарождения, причем вместе с усовершенствованной им версией Unix (4.2BSD). Там он начал работать над операционной системой SunOS (версия UNIX для рабочих станций и серверов). В 1983 году по решению суда началась ликвидация корпорации AT&T, и одним непредвиденным побочным эффектом этой ликвидации было то, что AT&T отныне могла сво-

бодно продавать систему UNIX как программный продукт. В результате увидела свет версия AT&T UNIX System V — общеизвестная, хотя и несколько неудобная коммерческая реализация системы UNIX.

Теперь, когда Berkeley, AT&T, Sun и другие дистрибутивы UNIX стали доступны для широкого круга организаций, был заложен фундамент для общей компьютерной инфраструктуры, построенной на технологии UNIX. Систему, которую использовали в области астрономии для вычисления звездных расстояний, можно было успешно применять в математике для вычисления множеств Мандельброта. И та же система одновременно обеспечивала доставку электронной почты для всего университета.

## Системные администраторы, ваш выход!

Управление компьютерными системами общего назначения требовало другого уровня знаний, чем двадцать лет назад. Ушли в прошлое дни, когда системный оператор обслуживал единственную компьютерную систему, предназначенную для выполнения узкоспециализированной задачи. Системный администратор в начале 1980-х годов стал настоящим "хозяином положения", который знает, как настроить систему UNIX, чтобы она удовлетворяла требованиям самых разных приложений и пользователей.

Поскольку система UNIX была весьма популярной в университетах и множество студентов горело желанием овладеть новейшими технологиями, именно университеты лидировали в создании организованных групп системных администраторов. Такие учебные заведения, как Университет Пердью, Университет штата Юта, Университет штата Колорадо, Университет штата Мэриленд и Университет штата Нью-Йорк (SUNY) в г. Буффало, стали "рассадниками" специалистов в области системного администрирования.

Кроме того, системные администраторы разработали собственные процессы, стандарты, инструкции и инструменты (например, `sudo`). Большинство из этих продуктов родилось из необходимости, поскольку без них системы работали нестабильно, что, естественно, вызывало недовольство пользователей.

Эви Немет (Evi Nemeth) приобрела известность как "мать системного администрирования" тем, что приглашала на работу в качестве системных администраторов студентов старших курсов Инженерного колледжа (Engineering College) при Университете штата Колорадо. Ее близкие связи с сотрудниками Калифорнийского университета в Беркли, Университета штата Юта и SUNY в г. Буффало позволили создать сообщество экспертов по вопросам системного администрирования, которые делились советами и инструментами. Ее команду часто называли как "munchkins", т.е. "переработчики информации" или "рабы Эви". Члены этой команды посещали различные конференции (в том числе и спонсируемые организацией USENIX) и работали там в качестве служебного персонала в обмен на возможность получать информацию, излагаемую участниками этих конференций.

Уже стало ясно, что системные администраторы должны были стать "мастерами на все руки". В 1980-х годах обычное утро системного администратора могло начаться с монтажа проводов для починки поврежденного переключателя на задней панели миникомпьютера VAX. Днем он мог заниматься очищением лазерного принтера первого поколения от рассыпавшегося тонера. Его обеденный перерыв мог бы быть потрачен на помощь какому-нибудь студенту отладить новый драйвер ядра, а вечерние часы могли быть заполнены записью информации на архивные ленты и уговорами пользователей почистить свои персональные каталоги, чтобы освободить пространство в файловой си-

стеме. Системный администратор был, без преувеличения, бескомпромиссным ангелом-хранителем, который должен был решить любую проблему.

1980-е годы можно было назвать временем ненадежного оборудования. Центральные процессоры были сделаны не из одной кремниевой микросхемы, а из нескольких сотен чипов, каждый из которых мог в любую минуту отказать. И именно системный администратор должен был быстро отыскать неисправный элемент и заменить его работающим. К сожалению, в то время еще не работала Federal Express (частная почтовая служба срочной доставки небольших посылок и бандеролей), и поэтому элементы для замены нужно было находить самим, что зачастую было очень непросто.

Однажды перестал работать наш любимый компьютер VAX 11/780, в результате чего весь университет остался без электронной почты. Мы знали, что недалеко от нас есть фирма, которая собирала компьютеры VAX, предназначенные "для научных исследований" для отправки в Советский Союз (то было время "холодной войны"). Практически без всякой надежды на успех мы заявились на склад с большой суммой наличных в кармане, и после часа переговоров мы все-таки получили необходимую печатную плату. Кое-кто отметил, что в то время в Боулдере (штат Колорадо) было легче достать наркотики, чем запчасти к VAX.

## Документация по системному администрированию и обучение

По мере того как отдельные "компьютерщики" стали считать себя системными администраторами — и причем стало ясно, что такая специализация может обеспечить достойную жизнь, — потребности в документации и соответствующем обучении ощутимо возросли. "Идя навстречу пожеланиям трудящихся", Тим О'Рейлли (Tim O'Reilly) и его команда (именуемая ранее *O'Reilly and Associates*, а теперь *O'Reilly Media*) начали публиковать документацию по системе UNIX, написанную простым языком и основанную на реальном опыте.

🕮 Больше ссылок на источники информации о системном администрировании приведено в главе 32.

В качестве посредника межличностного взаимодействия ассоциация USENIX провела свою первую конференцию, посвященную системному администрированию, в 1987 году. Эта конференция (Large Installation System Administration — LISA) охватила, в основном, западное побережье. Три года спустя был учрежден институт SANS (SysAdmin, Audit, Network, Security) для удовлетворения потребностей специалистов восточного побережья. В настоящее время конференции LISA и SANS обслуживают всю территорию США и до сих пор на достаточно высоком уровне.

В 1989 году мы опубликовали первое издание этой книги, имевшей тогда название *UNIX System Administration Handbook*. Она была быстро раскуплена, в основном, из-за отсутствия альтернативы. В то время наш издатель был настолько далек от системы UNIX, что в их производственном отделе все вхождения строки "etc" были заменены строкой "and so on", в результате чего появились такие имена файлов, как `/and so on/passwd`. Мы воспользовались создавшейся ситуацией, чтобы взять под полный контроль все содержимое книги от корки до корки, и сейчас, надо признать, наш издатель стал более осведомленным в вопросах UNIX. Наше 20-летнее сотрудничество с этим издателем дало пищу для других интересных историй, но мы их опустим, дабы не испортить наши вполне дружеские отношения.

## UNIX "при смерти". Рождение Linux (1991–1995)

К концу 1990 года казалось, что UNIX стремительно движется к мировому господству. Бесспорно, именно эту операционную систему выбирали как для ведения бизнеса (например, Taco Bell и McDonald's), так и для исследовательских и научных расчетов. Группа CSRG (Computer Systems Research Group — исследовательская группа по компьютерным системам) в Беркли, состоящая из Кирка Мак-Кузика (Kirk McKusick), Майка Карелса (Mike Karels), Кейта Бостика (Keith Bostic) и многих других, как раз выпустила версию 4.3BSD-Reno, основанную на выпуске 4.3, в которую была добавлена поддержка для процессора CCI Power 6/32 (с кодовым именем "Tahoe").

Коммерческие выпуски UNIX (например, SunOS) также пользовались успехом, который, отчасти, им обеспечили появление Интернета и первые шаги в направлении электронной торговли. Оборудование персонального компьютера (ПК) стало предметом широкого потребления. Оно уже было относительно надежным, недорогим и обеспечивало довольно высокую производительность. И хотя версии системы UNIX, запускаемые на ПК, действительно существовали, все они были коммерческими, причем с закрытым исходным кодом. Назрела ситуация для появления UNIX для ПК с открытым исходным кодом.

В 1991 году, группа разработчиков, трудившихся над выпусками BSD, — Донн Сили (Donn Seeley), Майк Карелс, Билл Джолитц (Bill Jolitz) и Трент Р. Хейн (Trent R. Hein) — вместе с другими приверженцами BSD основали компанию Berkeley Software Design, Inc. (BSDI). Под руководством Роба Колстада (Rob Kolstad) компания BSDI предоставляла исполняемые файлы и исходный код для полностью функциональной коммерческой версии BSD UNIX на платформе ПК. Среди прочего, этот проект доказал, что для массового производства компьютеров можно использовать недорогое оборудование ПК. Компания BSDI продемонстрировала взрывоподобный рост прибыли на заре развития Интернета, поскольку именно ее операционную систему выбирали первые провайдеры услуг Интернета (Internet service providers — ISP).

Пытаясь снова упрятать джинна, который выскочил из бутылки в 1973 году, корпорация AT&T в 1992 году начала судебный процесс против компании BSDI и членов правления университета Калифорнии (Regents of the University of California), заявив о копировании кода и воровстве производственных секретов. Юристам компании AT&T потребовалось более двух лет, чтобы идентифицировать проблемный код. В результате судебного разбирательства из кода BSD было удалено три файла (из более чем 18 000).

К сожалению, этот двухлетний период неопределенности оказал негативное воздействие на весь мир UNIX, операционную систему BSD и подобные ей не-BSD-версии. Многие компании перешли на использование Microsoft Windows, испугавшись, что они могут оказаться во власти компании AT&T, которая практически "задушила в объятиях" свое дитя. К тому времени, когда дым сражений развеялся, оказалось, что компании BSDI и CSRG "смертельно ранены". Эра BSD подходила к концу. Тем временем Линус Торвальдс (Linus Torvalds), студент колледжа в Хельсинки, наигравшись с Minix — свободной Unix-подобной микроядерной операционной системой, распространяемой по лицензии BSD, — начал писать собственную систему-клон UNIX[3]. К 1992 году появилось множество дистрибутивов Linux (включая SuSE и Yggdrasil Linux). В 1994 году мир узнал о создании систем Red Hat и Linux Pro.

[3] ОС Minix была разработана Эндрю Таненбаумом (Andrew S. Tanenbaum), профессором Амстердамского свободного университета.

Феноменальный успех Linux основан на многих факторах. Мощная поддержка всех, кому понравилась эта система, и обширный список программ из архива GNU сделали Linux неуязвимой системой. Она прекрасно работает в любых производственных средах, и поговаривают, что на основе Linux можно построить более надежную и более производительную систему, чем на основе любой другой операционной системы. Интересно также отметить, что отчасти своим успехом Linux обязана блестящей возможности, предоставленной ей действиями компании AT&T против BSDI и университета Калифорнии в Беркли. Тот неуместный судебный процесс вселил страх в сердца приверженцев UNIX прямо на заре электронной торговли и в начале эры Интернета.

Но не все ли равно теперь? Главное, что в результате всех этих перипетий осталась огромная потребность в системных администраторах. Багаж знаний и опыт, накопленный системными администраторами при обслуживании систем UNIX, полностью применимы к Linux, и большинство "UNIX-лоцманов" заботливо продолжали вести своих пользователей через бурные моря 1990-х. Возмите себе на заметку: хороший системный администратор должен всегда оставаться спокойным во время любого шторма.

## Мир Windows (1996–1999)

В 1993 году компания Microsoft выпустила ОС Windows NT. Эта "серверная" версия Windows, которая имела популярный пользовательский интерфейс, имела значительный эффект, причем как раз тогда, когда корпорация AT&T старалась убедить всех в том, что она больше никому не позволит обманывать себя в лицензионных вопросах. Как следствие, многие организации приняли Windows в качестве предпочтительной платформы для совместных вычислений. Это был конец 1990-х. Вне всякого сомнения, платформа Microsoft прошла долгий путь развития, и для некоторых организаций это был наилучший выбор.

К сожалению, сначала администраторы UNIX, Linux и Windows использовали в своей работе конкурентные подходы, соперничая за "место под солнцем" путем противопоставления аргументов из рекламы пива: "отличный вкус" против "меньшего объема"[4]. Многие администраторы систем UNIX и Linux начали срочно изучать Windows, убежденные в том, что в противном случае им придется "положить зубы на полку". А на горизонте уже показалась Windows 2000. С приближением "миллениума" будущее UNIX выглядело все более мрачным.

## Расцвет UNIX и Linux (с 2000-го по настоящее время)

С приходом Интернета все стрались понять, что "настоящее", а что — всего лишь мираж. Когда страсти "от новизны" немного улеглись, стало ясно, что многие организации с успешными техническими стратегиями использовали UNIX или Linux вместе с Windows, а не только что-то одно. Конкурентной войны больше не было.

Системные администраторы UNIX и Linux, которые пополнили свой квалификационный багаж системой Windows, стали еще более ценными специалистами. Они могли теперь "наводить мосты" над пропастью между двумя мирами и эффективно использовать обе системы на благо своей организации. Оценки экспертов, использующих методику определения наилучшего соотношения цены/качества (total cost of ownership — TCO), показали, что этот показатель для сервера Linux был значительно ниже аналогичного показателя для сервера Windows.

[4] Для ясности: в Windows действительно "меньше объема".

В настоящее время можно говорить о процветании UNIX и Linux. Коммерческие версии UNIX (AIX, Solaris и HP-UX) вполне отвечают требованиям соответствующих рынков. Linux и ПК-ориентированные UNIX-версии продолжают увеличивать свою долю рынка, причем Linux — единственная операционная система, у которой доля рынка по серверам растет постоянно (по данным 2010 года). Кроме того, не надо забывать о том, что такая современная операционная система от компании Apple, как Mac OS X, также построена на основе UNIX[5].

Множество новых достижений в системах UNIX и Linux относится к сфере виртуальных и облачных вычислений. (Подробнее об этих технологиях см. главу 24.) И снова-таки, в этих средах есть один общий элемент: системные администраторы. Ваш бесценный опыт как системного администратора обязательно найдет применение: и в материальной, и виртуальной среде!

## Завтрашний день UNIX и Linux

Не важно, в каком направлении пойдет развитие UNIX и Linux в ближайшие несколько лет, одно можно сказать точно: UNIX и Linux будут по-прежнему востребованы! Системные администраторы заботятся о целостности компьютерной инфраструктуры, решают сложнейшие проблемы эффективности и масштабируемости систем и снабжают квалифицированными рекомендациями в области компьютерных технологий как рядовых пользователей, так и руководителей организаций.

Мы — системные администраторы! Без нас — никак!

## Рекомендуемая литература


- Mckusick, Marshall Kirk, Keith Bostic, Michael J. Karels and John S. Quarterman. *The Design and Implementation of the 4.4BSD Operating System (2nd Edition)*. Reading, MA: Addison-Wesley, 1996.
- Salus, Peter H. *A Quarter Century of UNIX*. Reading, MA: Addison-Wesley, 1994.
- Salus, Peter H. *Casting the Net: From ARPANET to Internet and Beyond*. Reading, MA: Addison-Wesley, 1995.
- Salus, Peter H. *The Daemon, the Gnu, and the Penguin*. Marysville, WA: Reed Media Services, 2008. Эта книга также опубликована на сайте `www.groklaw.net`.


[5] Даже устройства iPhone, производимые компанией Apple, используют урезанный вариант системы UNIX, а операционная система Android (от компании Google) включает абстракции, взятые из ядра Linux.

# В защиту AIX

*Диалог с Дэном Фостером (Dan Foster)*

Система AIX известна с 1980-х годов, но в этом издании книги она впервые рассмотрена в качестве примера. Мы и ранее обсуждали внесение системы AIX в некоторые предыдущие издания, но всегда нам мешал тот факт, что она слишком сильно отличается от других версий UNIX. Если говорить точнее, нам было удобнее описывать их без упоминания AIX.

Мы хотели встретить AIX с распростертыми объятиями. Тем не менее внимательные читатели могли бы отметить определенное постоянство тона в отношении AIX, который звучит не совсем хвалебно. Подобно бездомному щенку, система AIX, казалось бы, всегда ведет себя как-то не так и никогда не понимает, почему вдруг все расстроены.

От этого всего нам не по себе: кому нравится прогонять щенка? Чтобы вернуть себе душевное равновесие, мы попросили Дэна Фостера, одного из наших рецензентов из мира AIX, описать достоинства системы AIX. Итак, мы выступим с обвинением против AIX, а Дэну предоставим слово для ее защиты.

## Наши бурные обвинения

AIX — операционная система, созданная для мэйнфреймов IBM в 1970-е годы и мучительно "застрявшая" в теле системы UNIX. Хотя структура UNIX поддерживает эту систему в работоспособном состоянии, AIX совершенно не заинтересована в соблюдении UNIX-соглашений. Она использует множество различных "шиньонов", "корсетов" и "косметических наборов", чтобы представить более гармоничный образ системы во вкусе IBM. Это — открытая, состоящая из модулей система, которая стремится быть закрытой и монолитной.

Тех, кто подходит к системе AIX как к UNIX, ожидает ряд затруднений. AIX не питает иллюзий, что администраторы действительно понимают, что они делают или должны сделать для модификации системы. Вместо простоты, модульности и гибкости, AIX предлагает структуру. Были затрачены значительные усилия, чтобы каталогизировать административные операции и собрать их "под крышей" интерфейсных средств администратора системы (System Management Interface Tool — SMIT). Если необходимой вам операции не окажется в этом каталоге,... ну что ж, не стоит из-за этого сильно переживать.

К сожалению, SMIT — не единственный дополнительный уровень косвенности AIX. Операции SMIT преобразуются в команды оболочки, поэтому каждая операция SMIT требует специальной команды, которая бы реализовала ее одним шагом. Отсюда и изобилие семейств команд (таких, как `crfs`/`chfs`/`rmfs`), которые реализуют заранее предписанные рецепты. Эти команды повышают уровень сложности системы и накладные расходы, не создавая больших ценностей, в то время как другие системы UNIX прекрасно обходятся без них.

Поскольку административные операции занимают промежуточное положение в программном обеспечении, система AIX не видит никакой необходимости сохранять информацию в текстовых файлах. Вместо этого она скопила множество разнообразных двоичных форматов и журналов регистрации, в основном, в виде подсистемы ODM (Object Data Manager — управление объектными данными). Системные администраторы могут использовать обобщенные ODM-команды для просмотра и модификации этих данных, но это уже из области черной магии, что, мягко говоря, удручает. В целом, ODM — это "черный и загадочный континент" с многочисленными "темными" углами, напоминающий системный реестр Windows.

Если сорвать "панцирь" AIX, то под ним обнаружатся печальные маленькие гомункулы UNIX, лежащие скрученными там и сям. Это "существо" нельзя назвать здоровым: оно испещрено возрастными морщинами, его кожа бледна вследствие отгороженности от внешнего мира и от усовершенствований UNIX последних десятилетий. Вероятно, компания IBM видит для своей системы не мейнстрим UNIX, а совсем другое направление развития.

## Дэн Фостер: аргументы в защиту

Да уж, вы нарисовали не очень приятную картину. Но, я думаю, будет справедливо охарактеризовать многие из перечисленных вами недостатков как синдром "не мы придумали", другими словами, как противодействие всему, что не подчиняется стандарту UNIX.

Есть определенная доля истины в вашем заявлении о том, что система AIX стремится быть больше, чем просто еще одним клоном UNIX. И это совсем не плохо. Система AIX предназначена не для ковбоев. Ее назначение — упростить администрирование системы, помочь в обеспечении надежности и устойчивости ее работы. По своим целям она серьезно отличается от других систем UNIX, поэтому и производит несколько иное впечатление.

AIX позаимствовала у IBM-систем AS/400 ряд полезных инструментов. Например, она использует централизованное средство регистрации ошибок, которое приложения могут легко задействовать через API. Такая система упрощает протоколирование ошибок, уведомление о событиях администратора и выявление проблем. Протокол Syslog реализует некоторые из этих функций, в основном, для UNIX, но, судя по тому, как описана регистрация ошибок во многих разделах этой книги, единообразия в его использовании не наблюдается. Несколько лет спустя компания Sun реализовала аналогичный подход с помощью демона **fmd** (fault management daemon — защита от ошибок и неисправностей) в системе Solaris 10.

Теперь возьмем подсистему управления оборудованием. AIX предоставляет централизованные средства диагностики для практически любого поддерживаемого системой устройства. Они позволяют даже регистрировать действия, связанные с ремонтом оборудования (в некоторых случаях возможно отключение неисправных светодиодов и генерирование соответствующих уведомлений), обеспечивая таким образом ведение журнала контрольных записей. Эта система предусматривает и другие действия по оказанию помощи через процедуры обратного вызова, но не оставляет вас "на произвол судьбы", как большинство версий UNIX.

Отвечая на ваши конкретные претензии, могу сказать, что команды, предназначенные для выполнения конкретных задач, — это просто особенность, а не дефект системы! Они (команды) зачастую оказываются полезными для системных администраторов.

- Как показано в приведенном вами примере, наборы AIX-команд имеют ясные имена, говорящие сами за себя (чего нельзя сказать о командах UNIX). Семейство команд **mk*** предназначено для создания объектов, семейство **rm*** — для их удаления, **ch*** — для модификации, а **ls*** — для отображения текущего состояния. Эти структурированные семейства команд генерируют предсказуемые результаты и позволяют сократить время, необходимое на их изучение. Они уменьшают вероятность выполнения неправильной команды, если, например, вы засиделись до глубокой ночи и с затуманенными глазами все же продолжаете работать или если вы только начинаете осваивать систему AIX. Конечно, в таких ситуациях хорошую службу сослужит вам и меню SMIT (интерфейсные средства администратора системы).

- Команды могут проверять допустимость своих аргументов, а файлы конфигурации — нет. Команда-"обертка" способна гарантировать, что предлагаемое вами изменение не разрушит систему. Если вы все же намерены это сделать, команда может "пожаловаться на вас" или, попросту, отказаться внести такое изменение в систему. Это гораздо лучше, чем *потом* "пожинать горькие плоды", случайно изменив содержимое файла конфигурации.
- Специализированные команды упрощают написание сценариев. В них не только сочетаются функции и допустимые аргументы, но они также освобождают сценарии от необходимости анализировать сложные файлы конфигурации и управлять ими. Это делает сценарии короче, надежнее и проще для обслуживания (и изучения!). Воспринимайте эти команды как высокоуровневую библиотеку функций администрирования, которая встроена в операционную систему и способна понимать практически любые языки, на которых пишутся сценарии.
- Использование упомянутого интерфейса администратора уменьшает зависимость от конкретных файловых форматов или реализаций. Он позволяет IBM изменять свои серверные приложения и вводить новые технологии, не разрушая существующих сценариев. Например, в настоящее время подсистема ODM сохраняет свои данные в файлах BDB (Berkeley DB — высокопроизводительная встраиваемая база данных, реализованная в виде библиотеки). Однако IBM не могла просто заменить ODM протоколом LDAP (Lightweght Directory Access Protocol — облегченный протокол службы каталогов) или некоторыми другими будущими технологиями, сохранив ODM-команды и пользовательский интерфейс в прежнем виде.

Просто не будьте ненавистниками SMIT! SMIT — гибкая система, которая реализует разнообразные интерфейсы (X11, веб-страница, клиент командной строки). Эта гибкость означает, что вы можете использовать один тот же интерфейс, сидя за офисным компьютером или работая дома глубокой ночью.

Интерфейс SMIT упрощает сложные процедуры и помогает начинающим администраторам быстрее выйти на должный уровень квалификации. Он используется и поддерживается уже многие годы, в течение которых постоянно совершенствовались различные пользовательские интерфейсы. Его легко использовать независимо от вашего отношения к UNIX вообще или к AIX в частности. И потом, SMIT — это прекрасное средство обучения, даже для опытных администраторов. Вы можете заполнить SMIT-форму требуемыми для вас значениями, после чего SMIT будет работать именно с теми командами, которые вам нужны. Ничего подобного нет ни в какой другой системе, и это замечательно!

Что касается вашего определения AIX как не "настоящей" UNIX-версии, то это просто неправда. Изначально система AIX была основана на ОС BSD, и следы этого происхождения (например, использование mbuf-кластеров в сетевом стеке AIX) сохранились по сей день. Позже внимание разработчиков переключилось на совершенствование базовых составляющих системы System V. Система AIX была сертифицирована на соответствие стандартам SUS (Single Unix Specification), X/Open и POSIX. Кроме того, компания IBM воспользовалась легендарной переносимостью системы UNIX и перевела AIX в ранг систем, которые могут работать на машинах широкого диапазона: от персональных компьютеров серии PS/2 до мэйнфреймов. Наконец, суперкомпьютер Deep Blue (высокопроизводительный кластер IBM HPC), который одержал победу в матче из шести партий с чемпионом мира по шахматам Гарри Каспаровым в 1996 и 1997 годах, работал под управлением системы AIX!

## Л



## М



## Н

## О



## П

## P



## C

## T

## Х



## Ц



## Э



## Я

**Системное администрирование**
**UNIX/LINUX**

*"Как автор, редактор и издатель, я никогда не придавал большого значения конкуренции — за исключением нескольких случаев. Это один из таких случаев. UNIX System Administration Handbook — это одна из немногих книг, на которые мы равняемся".* — **Тим О'Рейли (Tim O'Reilly), основатель компании O'Reilly Media (из предисловия)**

---

Данное издание, появившееся в год, когда исполняется 20 лет со дня появления мирового бестселлера по системному администрированию UNIX, стало еще лучше благодаря описанию распространенных вариантов системы Linux: Ubuntu, openSUSE и RHEL.

Системное администрирование в книге рассматривается с практической точки зрения. Она представляет собой бесценный справочник как для начинающих администраторов, так и для опытных профессионалов. В ней подробно описываются эффективные методы работы и рассматриваются все аспекты системного администрирования, включая управление памятью, проектирование и управление сетями, электронную почту, веб-хостинг, создание сценариев, управление конфигурациями программного обеспечения, анализ производительности, взаимодействие с системой Windows, виртуализацию, DNS, безопасность, управление провайдерами IT-услуг и многое другое. В данной книге отражены текущие версии следующих операционных систем.

Ubuntu® Linux  openSUSE® Linux  Red Hat® Enterprise Linux

Oracle America® Solaris™ (бывший Sun Solaris)  HP HP-UX®  IBM AIX®

---

*"Эта книга увлекательна и полезна как справочник. Если вы используете системы UNIX и Linux, она должна стать вашей настольной книгой. В ней кратко и без лишних разглагольствований написано об истории этих систем. Она содержит точную информацию, которая излагается в яркой и запоминающейся форме".* — **Джейсон А. Наннелли (Jason A. Nunnelley)**

*"Это всесторонний справочник о том, как обслуживать и поддерживать работоспособность систем UNIX и Linux. Авторы излагают факты, сопровождая их практическими советами и реальными примерами. Их точка зрения на различия между системами представляет ценность для всех, кто работает в неоднородных вычислительных средах".* — **Пат Парсегян (Pat Parseghian)**


ВИЛЬЯМС
Издательский дом "Вильямс"
www.williamspublishing.com

PRENTICE HALL
PEARSON EDUCATION

ISBN 978-5-8459-1740-9
11052
9 785845 917409