ДЛЯ БАКАЛАВРОВ

Национальный исследовательский университет
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

С.В. Назаров, Л.П. Гудыно, А.А. Кириченко

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Практикум

Рекомендовано УМО в области экономики, менеджмента, логистики и бизнес-информатики в качестве **учебного пособия** для студентов вузов, обучающихся по направлению подготовки 080700 «Бизнес-информатика»

КноРус
МОСКВА
2012

Д Л Я  Б А К А Л А В Р О В

Национальный исследовательский университет
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

С.В. Назаров, Л.П. Гудыно, А.А. Кириченко

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Практикум

Рекомендовано УМО в области экономики,
менеджмента, логистики и бизнес-информатики
в качестве **учебного пособия**
для студентов вузов, обучающихся по направлению
подготовки 080700 «Бизнес-информатика»

КноРус
МОСКВА
2012

УДК 004.45(075.8)
ББК 32.973.26я73
Н19

Рецензенты:
кафедра математического обеспечения и технологий программирования МЭСИ (заведующая кафедрой канд. экон. наук, доц. Комлева Н.В.)
А.Н. Сазанович, проректор по информационным технологиям МИРБИС, д-р техн. наук, проф.

Назаров С.В.

Н19 Операционные системы. Практикум : учебное пособие / С.В. Назаров, Л.П. Гудыно, А.А. Кириченко. — М. : КНОРУС, 2012. — 376 с. — (Для бакалавров).

ISBN 978-5-406-00886-7

Содержит материал, обеспечивающий проведение практических, лабораторных, семинарских занятий и курсовое проектирование по основам построения и функционирования современных операционных систем. Рассмотрены организация пользовательского интерфейса на основе современной командной оболочки PowerShell, мультипрограммные вычислительные процессы, управление памятью и устройствами, файловые системы. Уделено внимание вопросам безопасности, защите и восстановлению операционных систем и их сетевым возможностям, а также средствам виртуализации и организации множественных прикладных сред.

Для студентов (бакалавриат) вузов экономического и технического профиля, обучающихся по специальностям: «Бизнес-информатика», «Прикладная информатика в экономике», «Прикладная математика», «Информатика и вычислительная техника», «Вычислительные машины, комплексы, системы и сети», «Математическое обеспечение и администрирование информационных систем» в объеме, определенном Минобрнауки России.

УДК 004.45(075.8)
ББК 32.973.26я73

Назаров Станислав Викторович
Гудыно Лев Петрович
Кириченко Александр Аполлонович

ОПЕРАЦИОННЫЕ СИСТЕМЫ. ПРАКТИКУМ

Сертификат соответствия № РОСС RU. АЕ51. Н 15407 от 31.05.2011 г.

Изд. № 2662. Подписано в печать 27.09.2011. Формат 60×90/16.
Гарнитура «NewtonC». Печать офсетная.
Усл. печ. л. 23,5. Уч.-изд. л. 13,0. Тираж 1500 экз. Заказ №

ООО «КноРус».
129085, Москва, проспект Мира, д. 105, стр. 1.
Тел.: (495) 741-46-28.
E-mail: office@knorus.ru http://www.knorus.ru

Отпечатано в ГУП «Брянское областное полиграфическое объединение».
241019, г. Брянск, пр-т Ст. Димитрова, 40.

ISBN 978-5-406-00886-7

Оглавление

Предисловие

Дисциплина «Операционные системы» является одной из важнейших в подготовке современного специалиста по применению компьютерных систем в различных сферах деятельности человека — экономике и управлении, транспорте и связи, образовании, научных исследованиях и др. Знание основ построения и принципов функционирования операционных систем позволяет организовать в информационных системах различного назначения эффективные вычислительные процессы, решать вопросы информационной безопасности, защиты от сбоев и отказов, а также организовать эффективную службу администрирования. Еще большее значение приобретают эти знания при создании программного обеспечения информационных систем.

Дисциплина «Операционные системы» в достаточно большом объеме включена в Государственный образовательный стандарт по нескольким направлениям обучения: 010200 «Прикладная математика», 010503 «Математическое обеспечение и администрирование информационных систем», 080700 «Бизнес-информатика», 080800 «Прикладная информатика» и др. Учебное пособие предназначено обеспечить практические занятия курсового и дипломного проектирования при подготовке бакалавров перечисленных специальностей. Может быть полезно студентам других специальностей, будущая сфера деятельности которых тесно связана с использованием вычислительной техники. К ним относится ряд специальностей по направлениям обучения: 09000 «Информационная безопасность» и 230000 «Информатика и вычислительная техника».

Материал, ставший основой настоящего учебного пособия, использовался в течение восьми лет при проведении практических занятий со студентами факультета бизнес-информатики Государственного университета «Высшая школа экономики». При подготовке книги авторы рассчитывали не только на студенческую читательскую аудиторию, но и на преподавателей. Как видит читатель, учебное пособие имеет довольно большой объем и в некотором смысле может показаться избыточным. Это сделано в следующих целях. Во-первых, как уже отмечалось выше, пособие рассчитано на использование в учебном процессе студентов различных направлений обучения (хотя, может быть, и родственных). На изучение дисциплины «Операционные системы» у них может отводиться различный объем учебного времени. Во-вторых, при некотором количестве избыточного материала преподаватель может выбрать именно тот, который посчитает наиболее целесообразным. В-третьих, пособие рассчитано на его использование в курсовых и выпускных работах студентов. В-четвертых, оно предоставляет достаточно много материала для научно-исследовательской

работы студентов. Отдельные разделы пособия могут быть использованы в лекционной части курса.

Учебное пособие состоит из восьми глав, содержание которых позволяет достаточно подробно рассмотреть основополагающие принципы построения и функционирования современных операционных систем на примере операционных систем Windows 2000/2003/XP/Vista. Авторы посчитали это целесообразным по причине наибольшего распространения в вузах персональных компьютеров с этими операционными системами. Однако значительное внимание уделено и свободно распространяемым системам Linux, Ubuntu, основанным на принципах операционной системы Unix.

Предлагаемая книга в значительной степени содержит материал одноименного учебного пособия авторов, опубликованного издательством КУДИЦ-ПРЕСС в 2008 г. Исключен неактуальный материал, добавлены новые главы по оболочке командной строки Power Shell и сетевым возможностям операционных систем.

Каждая глава пособия содержит материал, достаточный для постановки не менее трех-четырех практических занятий (работ). Специального разбиения глав на практические работы не делалось (авторы считают, что это задача преподавателя, проводящего занятия). Однако предварительное разбиение обозначено заданиями для самостоятельной работы. Ряд глав завершается перечнем задач, решение которых позволяет углубить понимание основополагающих принципов функционирования операционных систем.

При проведении занятий предусматривается использование значительного количества программных средств, поставляемых комплектно с операционными системами корпорации Microsoft или являющихся составной частью пакетов MS SDK, Windows Resource Kit и Windows Support Tools. Кроме того, используются утилиты других производителей, общедоступное программное обеспечение которых можно получить в Интернете, а также программы, разработанные студентами факультета бизнес-информатики Государственного университета «Высшая школа экономики». Авторы посчитали нецелесообразным приведение текстов программ в приложении к книге (оно было бы очень объемным). Во всех случаях использования таких программ даются только ссылки на источник или производителей программ, а также на сайт упомянутого факультета.

Работа над книгой распределилась следующим образом: предисловие, главы 2—7 — С.В. Назаров; глава 1 — Л.П. Гудыно; глава 8 — совместно С.В. Назаров и А.А. Кириченко, глава 9 — А.А. Кириченко. Общее редактирование книги выполнено С.В. Назаровым.

ГЛАВА 1

РАБОТА В СРЕДЕ КОМАНДНОЙ ОБОЛОЧКИ MICROSOFT POWERSHELL

1.1. Назначение пакета PowerShell

Основной задачей любой операционной системы (ОС) является управление ресурсами компьютерной системы. Именно на основе этих функций управления создается сервис для пользователей. Эффективная профессиональная работа опытного пользователя с ОС компьютера немыслима без овладения интерфейсом, обеспечиваемым командной строкой. Этот вид интерфейса является одним из основных применительно к ОС Unix и Linux. Преимуществом данного интерфейса служит возможность «более утонченного» управления ресурсами системы, чем с помощью графического интерфейса (Graphical Unit Interface — GUI) ОС Windows.

Объяснением этого факта, по-видимому, служит следующее. Unix-системы ориентировались на работу профессионально подготовленных пользователей (операторов, программистов, системных инженеров). В них интерфейс командной строки всегда был и остается традиционно богатым и мощным.

Напротив, Windows-ориентированные системы ведут свое развитие от простых персональных компьютеров. Корпорация Microsoft при разработке для них ОС ориентировалась, в первую очередь, на пользователей-непрофессионалов и закладывала принцип «нулевого администрирования». Согласно этому принципу в различных версиях ОС Windows предусматривалось лишь минимальное участие пользователей в управлении и распределении ресурсов систем. Выполнение этих функций стало прерогативой программ самой ОС. Пользователи же довольствовались в основном сервисом графического интерфейса.

Интерфейс командной строки (Command Prompt Interface) в ОС Windows присутствует и играет для большинства пользователей вспомогательную роль. В свое время он формировался как некое подмно-

жество ОС Unix и особого развития не получил. Однако интерфейс командной строки во многих нештатных ситуациях остается единственным средством определения рассогласований и «тонкой настройки» аппаратно-программных средств. Поэтому в новых версиях ОС Windows на рубеже нового тысячелетия с учетом роста сложности аппаратной и программной частей компьютерных систем добавлен ряд команд, позволяющих выполнять некоторые настройки. Часть команд, заимствованных из MS DOS, получила дополнительные возможности. Например, такие команды, как dir, copy, xcopy, rename и другие в новых редакциях ОС Windows работают с длинными именами файлов.

Начало нового столетия ознаменовалось дальнейшим усложнением структуры компьютеров: появились многоядерные микропроцессоры, повысилась роль распределенных вычислений, что явилось стимулом развития сетевых технологий (сетевые службы, центры обработки данных, «облачные» вычисления). Возможности старых средств стали недостаточными, потребовалась разработка новых средств администрирования и управления ресурсами компьютерных систем.

Windows PowerShell — командная оболочка следующего поколения и язык сценариев от фирмы Microsoft, которые можно использовать вместо устаревшего интерпретатора команд cmd.exe и языка сценариев VBScript. Разработчики Windows PowerShell к разработке нового средства подошли комплексно. Они не просто ввели в интерфейс командной строки набор новых утилит (команд), обеспечивающих возможность достаточно просто выполнять сложные процедуры управления ресурсами компьютерных систем, но и обеспечили совместимость с ранее созданными разработками, в том числе и разработками других платформ, например Unix-ориентированными.

Главной задачей разработки PowerShell было создание среды составления сценариев, которая наилучшим образом подходила бы для современных версий ОС Windows и была бы более функциональной, расширяемой и простой в использовании, чем любой какой-либо аналогичный продукт для любой другой ОС. Компания Microsoft в настоящее время позиционирует эту оболочку как основной инструмент управления ОС и рядом разработанных ею приложений. PowerShell официально включен в качестве стандартного компонента в ОС Windows Server 2008, а также используется в продуктах Microsoft, таких как Exchange Server 2007, System Center Operations Manager 2007, System Center Data Protection Manager V2 и System Center Virtual Machine Manager.

Идея построения и развития принципиально новой командной оболочки основана на нескольких здравых и прозрачных положениях.

Она должна стать мощным средством управления и настроек компьютерных систем. Исходными данными для этого является наличие большого количества ресурсов (объектов), необходимых для выполнения автоматических вычислений. Ресурсы могут быть локальными и сетевыми, привлекаемыми для организации вычислений. Каждый ресурс имеет некоторый особый набор характеристик (свойств), каждая характеристика — количественное или качественное выражение. В целях управления необходимо отслеживать состояние ресурсов не только в статике, но и в динамике вычислений. Нужно иметь средства, чтобы просматривать состояния ресурсов, уметь отбирать необходимую информацию о некоторых из них, отсеивать, сортировать, фильтровать сведения, проводить анализ и обработку отобранных данных и предоставлять результаты пользователям (клиентам и администраторам компьютерных систем). Творческий характер процедур анализа и обработки требует включения в оболочку средств разработки программ-функций. Результаты анализа и обработки должны использоваться в управлении. Простейшим видом управления служит выдача справок о состоянии объектов управления. В более сложных случаях результаты должны использоваться для корректировки и изменения состояний ресурсов, а режим управления становиться автоматизированным или даже автоматическим.

По существу новая командная оболочка представляет собой весьма специфичную базу данных, отражающую ресурсы компьютерных систем, а также и систему управления этой базой данных (СУБД) и ресурсами. СУБД включает все необходимые атрибуты: язык описания структуры и данных, язык запросов, язык манипулирования данными, генератор отчетов и т.д. Правда, некоторые из этих категорий представлены в неявном виде. Самым интересным здесь является то, что база данных с ресурсами и СУБД включаются в контур управления реальных компьютерных систем в автоматизированном режиме.

Отличительной особенностью построения PowerShell служит ее ориентация на объектную модель платформы .NET. Именно средства этой модели обеспечивают возможность взаимодействия различных частей операционной системы друг с другом. Кроме того, объектная модель .NET является самодокументируемой, т.е. каждый ее объект содержит информацию о своей структуре. Это свойство очень важно при интерактивной работе пользователя, поскольку все необходимые сведения о привлекаемых компонентах находятся «под рукой». Принадлежностью модели .NET служат достаточно мощные библиотеки, имеющие большую функциональность.

1.2. Начало работ в среде PowerShell

PowerShell включена во все новые версии ОС Microsoft Windows, начиная с Vista. Если на компьютере пользователя PowerShell отсутствует, то необходимо сначала установить платформу .NET. Загрузить ее дистрибутив можно с сайта Microsoft (http://msdn.microsoft.com/netframework/downloads/updates/default.aspx). После этого можно установить и собственно оболочку PowerShell (http://microsoft.com/powershell) с учетом версии и языка представления справок по системе.

Запуск оболочки осуществляется по одному из трех вариантов:

1) нажать кнопку Пуск, открыть Все программы, найти и выбрать Windows PowerShell;
2) нажать кнопку Пуск, выбрать пункт Выполнить, ввести имя файла powershell, нажать кнопку OK;
3) в командной строке интерпретатора команд cmd.exe ввести имя файла powershell, нажать кнопку OK.

После запуска PowerShell должно открыться командное окно оболочки с приглашением ввода команд (рис. 1.1).

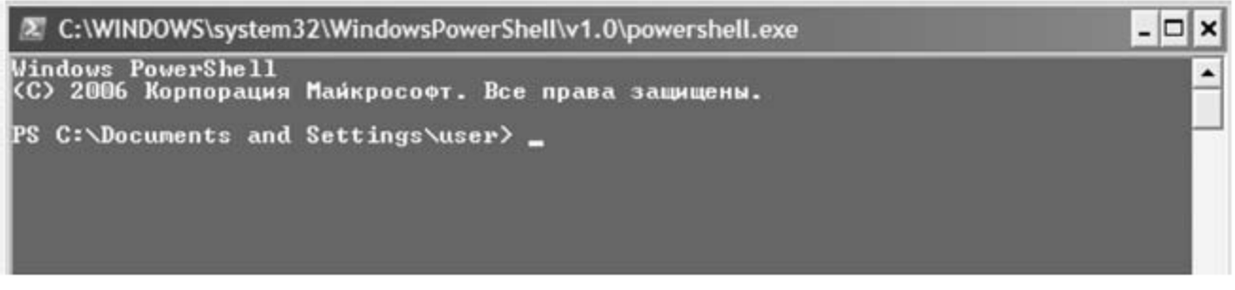


Рис. 1.1

Следует обратить внимание на вид строки приглашения. Она очень похожа на строку приглашения cmd.exe, но в ее начале стоят буквы PS, указывающие на принадлежность к оболочке PowerShel.

Для выхода из среды PowerShell автономного компьютера можно набрать команду exit и нажать клавишу Enter или просто закрыть окно оболочки, но такой способ выхода не является корректным, так как данные проведенного пользователем процесса будут потеряны. При работе в компьютерной сети, с сетевыми ресурсами и с сервером существуют три способа завершения сеанса активного пользователя:

1) командой logoff без параметров;
2) командой shutdown –l, т.е. вызовом Windows-утилиты shutdown.exe с параметром –l;

3) методом Win32Shutdown WMI-класса Win32_Operating-System с параметром 0. Для этого нужно набрать достаточно длинную команду (Get-WMIObject Win32_OperatingSystem). Win32Shutdown(0).

Целесообразно ознакомиться со справочной информацией по данным завершениям. Справки вызываются по командам logoff /? и shutdown /?.

Разработчики Windows PowerShell предполагали, что большинство пользователей этого средства будут работать с ним в интерактивном режиме. Ввод команд можно выполнять как по отдельности, так и группировать их в конвейеры или в пакетные файлы.

При разработке новой оболочки командной строки разработчики постарались учесть все лучшее из накопленного опыта в различных ОС. Они пытались сохранить не только преемственность между прежними и новыми системами, но и предоставить возможность работы администраторам вычислительных систем в привычном для них интерфейсе. Поэтому новым средствам управления командной строки можно назначать дополнительные имена (псевдонимы). Это обеспечивает, например администраторам Unix-ориентированных систем, использование новой среды в привычных для них терминах, что облегчает изучение и применение PowerShell.

При первом сеансе работы желательно посмотреть и сравнить результаты выполнения нескольких команд, уже известных пользователю, например по работе с интерпретатором команд cmd.exe. Практически все команды интерпретатора здесь имеют аналоги с теми же именами (псевдонимами), но представление данных отличается, иногда очень значительно. Прежде всего следует отметить уровень детализации информации. Посмотрим различия в выполнении команды dir. В среде cmd.exe результат имеет вид, показанный на рис. 1.2.

В среде PowerShell имеется специфичная команда — командлет (Cmdlet) Get-ChildItem, которая также имеет дополнительное имя (псевдоним) dir. Эта команда без параметров предоставляет очень похожие данные (рис. 1.3). Приверженцам ОС Unix и Linux более привычным является использование псевдонима этого командлета ls (лист) с теми же функциями. В новой редакции команды dir появился столбец Mode, отражающий возможные режимы использования программных средств, очень похожие на режимы Unix. Режимы определяются отдельно для каталогов (d) и файлов (a). Полный перечень характеристик, выводимый по различным командам, можно просмотреть с помощью командлета Get-Member.

```
C:\Documents and Settings\user>dir
 Том в устройстве C имеет метку WindowsXP
 Серийный номер тома: CCE6-83D4

 Содержимое папки C:\Documents and Settings\user

15.11.2009  18:45    <DIR>          .
15.11.2009  18:45    <DIR>          ..
20.11.2008  13:50           162 387 dd_depcheckdotnetfx30.txt
20.11.2008  13:50               312 dd_dotnetfx3error.txt
20.11.2008  13:51           112 548 dd_dotnetfx3install.txt
20.11.2008  13:50           370 898 dd_msxml_retMSI4CB3.txt
20.11.2008  13:51           140 906 dd_msxml_retMSI4D49.txt
20.11.2008  13:50         7 825 818 dd_netfx_retMSI4CC6.txt
20.11.2008  13:50           134 742 dd_rgb_retMSI4CA5.txt
20.11.2008  13:50             2 488 MAN936.tmp
29.10.2006  22:19            77 108 SDB939.tmp
20.11.2008  13:51            32 716 uxeventlog.txt
20.11.2008  13:50             9 656 VSMsiLog4D38.txt
20.11.2008  13:50    <DIR>          VSW0
20.11.2008  13:50             1 810 VWL937.tmp
20.11.2008  13:50            78 472 WLF938.tmp
21.11.2008  11:42    <DIR>          Главное меню
29.10.2009  16:36    <DIR>          Избранное
15.11.2009  14:37    <DIR>          Мои документы
15.11.2009  14:47    <DIR>          Рабочий стол
              13 файлов      8 949 861 байт
               7 папок  19 641 188 352 байт свободно

C:\Documents and Settings\user>_
```

Рис. 1.2

```
PS C:\Documents and Settings\user> dir

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\Documents and Settings\us
    er


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         20.11.2008     13:50            VSW0
d-r--         21.11.2008     11:42            Главное меню
d----         29.10.2009     16:36            Избранное
d-r--         15.11.2009     14:37            Мои документы
d----         15.11.2009     14:47            Рабочий стол
-a---         20.11.2008     13:50     162387 dd_depcheckdotnetfx30.txt
-a---         20.11.2008     13:50        312 dd_dotnetfx3error.txt
-a---         20.11.2008     13:51     112548 dd_dotnetfx3install.txt
-a---         20.11.2008     13:50     370898 dd_msxml_retMSI4CB3.txt
-a---         20.11.2008     13:51     140906 dd_msxml_retMSI4D49.txt
-a---         20.11.2008     13:50    7825818 dd_netfx_retMSI4CC6.txt
-a---         20.11.2008     13:50     134742 dd_rgb_retMSI4CA5.txt
-a---         20.11.2008     13:50       2488 MAN936.tmp
-a---         29.10.2006     22:19      77108 SDB939.tmp
-a---         20.11.2008     13:51      32716 uxeventlog.txt
-a---         20.11.2008     13:50       9656 VSMsiLog4D38.txt
-a---         20.11.2008     13:50       1810 VWL937.tmp
-a---         20.11.2008     13:50      78472 WLF938.tmp

PS C:\Documents and Settings\user>
```

Рис. 1.3

Командная строка PowerShell кроме набора и выполнения команд предоставляет пользователю возможность вычислений арифметических выражений различной сложности. В простейшем случае она обе-

спечивает вычисления как калькулятор, но при этом обращения к программе «Калькулятор», входящей в набор программ «Стандартные», не происходит. После записи выражения в командной строке и нажатия клавиши Enter результат вычисления отображается на следующей строке. Несколько простых примеров приведено на рис. 1.4.

```
PS C:\Documents and Settings\user> 144/13-21.2*4
-73,7230769230769
PS C:\Documents and Settings\user> (18-4)*3
42
PS C:\Documents and Settings\user> 100/33
3,03030303030303
PS C:\Documents and Settings\user> [System.Math]::Sqrt(625)
25
PS C:\Documents and Settings\user>
```

Рис. 1.4

В более сложных случаях выражения могут включать различные математические функции. Их реализация обеспечивается путем обращения к библиотекам классов платформы .NET, в частности к методам класса System.Math.

Полный список доступных математических функций может быть получен командой, представляющей собой конвейер [System.Math] | Get-Member –Static (рис. 1.5).

```
PS C:\> [System.Math]|Get-Member  -Static

   TypeName: System.Math

Name                MemberType Definition
----                ---------- ----------
Abs                 Method     static System.SByte Abs(SByte value), static Syst...
Acos                Method     static System.Double Acos(Double d)
Asin                Method     static System.Double Asin(Double d)
...
```

Рис. 1.5

Многоточие в последней строке на рис. 1.5 здесь и далее обозначает, что список функций представлен не полностью, а только фрагментом.

При сложных вычислениях может потребоваться сохранение промежуточных результатов в каких-то ячейках памяти. Для этого следует очень простыми средствами определить имя переменной (имена переменных) и определить ей (им) значение. Имена переменных должны начинаться знаком $. Запись только имени переменной после знака доллара означает обращение к выдаче ее значения (рис. 1.6).

На первых сеансах работы пользователей будет полезно использование команд-псевдонимов cls (очистка экрана дисплея) и cd (изменение каталога), знакомых пользователям по работе с интерпретатором команд cmd.exe. Функциональность этих команд остается прежней.

```
PS C:\Documents and Settings\user> $alfa=16/4
PS C:\Documents and Settings\user> $beta=5
PS C:\Documents and Settings\user> $alfa=$alfa+$beta
PS C:\Documents and Settings\user> $alfa
9
PS C:\Documents and Settings\user>
```

Рис. 1.6

Задания для самостоятельной работы

1. Загрузить командную оболочку PowerShell и проделать приведенные выше примеры.
2. Для пользователей cmd.exe в PowerShell определены псевдонимы: cd, cls, copy, del, dir, echo, erase,more, popd, pushd, ren, rmdir, sort, type; для пользователей Unix — псевдонимы: cat, chdir, clear, diff, h, history, kill, lp, ls, mount, ps, pwd, r, rm, sleep, tee, write. Просмотреть работу средств PowerShell под указанными псевдонимами.
3. Опробовать работу PowerShell в режиме калькулятора.
4. Просмотреть список всех математических функций, находящихся в библиотеке [System.Math]. Выбрать некоторые из списка по желанию и просмотреть их работу со значениями переменных.

Примечание. Обратить внимание на способность оболочки запоминать «историю» набора команд пользователем. Это дает возможность пользователям повторять отдельные последовательности команд, выделять определенные части и даже переносить их в командные файлы — сценарии для последующего использования.

1.3. Структура пакета PowerShell и его справочная система

Разносторонний вид ресурсов компьютерных систем и специфический характер управления каждым из них, видимо, не позволили разработчикам создать единую систему управления ресурсами с четкой и строгой структурой. В связи с постоянным усложнением компьютеров, а также систем и сетей на их основе состав средств управления не может оставаться постоянным, он должен совершенствоваться, пополняться, адаптироваться к новым условиям построения и применения информационных систем. Поэтому разработчики решили сделать новую оболочку предельно простой и хорошо документированной. В интерактивном режиме

пользователь — администратор всегда может посмотреть, какие средства и в каком режиме он может использовать для достижения определенных целей.

Изучение оболочки лучше начинать с уяснения структуры и возможностей справочной системы. Для этого целесообразно сначала ознакомиться с функциями команды (командлета) Get–Help, обеспечивающей получение справочных данных по всем подсистемам PowerShell с различной детализацией. Именно здесь указываются первые сведения о принципах построения новой оболочки и сведения о делении командлетов на группы. Наберем в командной строке фразу get–help или get–help –?, можно воспользоваться псевдонимом help без параметров (рис. 1.7).

```
PS C:\Documents and Settings\user> get-help
РАЗДЕЛ
    Командлет Get-Help

КРАТКОЕ ОПИСАНИЕ
    Отображает сведения о командлетах и концепциях PowerShell.

ПОЛНОЕ ОПИСАНИЕ

СИНТАКСИС
    get-help {<ИмяКомандлета> | <ИмяРаздела>}
    help {<ИмяКомандлета> | <ИмяРаздела>}
    <ИмяКомандлета> -?

    Команды "Get-help" and "-?" отображают справку на одной странице.
    Команда "Help" - на нескольких.
```

Рис. 1.7

По команде get–help * выводится внушительный список разделов справочной системы оболочки, где они разделены на четыре большие группы (Category). Группы имеют обозначения: Alias (псевдоним), Cmdlet (командлеты), Provider (провайдер — программа, обеспечивающая доступ к определенному хранилищу данных) и HelpFile (файл помощи) (рис. 1.8).

```
PS C:\Documents and Settings\user> get-help *

Name                              Category                  Synopsis
----                              --------                  --------
ac                                Alias                     Add-Content
asnp                              Alias                     Add-PSSnapin
clc                               Alias                     Clear-Content
cli                               Alias                     Clear-Item
clp                               Alias                     Clear-ItemProperty
```

Рис. 1.8

Каждая категория может вызываться отдельно, если команду Get–Help набирать с параметром –category и именем группы, например:

PS C:\Documents and Settings\user> Get–Help –Category provider.

Вызов же справки по любому элементу группы производится указанием имени элемента после имени командлета Get–Help, например:

PS C:\Documents and Settings\user> Get–Help Alias.

Каждому пользователю необходимо самостоятельно изучить дерево справочной системы, начиная с общих разделов.

Задания для самостоятельной работы

1. Вызвать обобщенную справку по пакету PowerShel, набрав в командной строке Get–Help без параметров. Затем просмотреть справочные данные по командам Get–Help / ? и help. Убедиться, что каждая команда имеет собственный контекст. Если первые две команды выдают одностраничные справки, то последняя команда дает многостраничную справку.
2. Отобразить все разделы справочной системы, набрав команду Get–Help *. Обратить внимание, что параметр * является шаблоном, обозначающим «любое сочетание символов». Список тем, обсуждение которых представлено в справочной службе PowerShell, можно просмотреть командой get–help about_*.
3. Исследовать структуру PowerShell по перечню разделов справки, набрав ряд команд, указанных в качестве примеров по одному из разделов. Посмотреть, как меняется содержание справочных данных, если в команду справки включаются параметры –detailed или –full.
4. Просмотреть справку по командлету Get-Process, отображающему процессы, активизированные в локальном компьютере пользователя. Для этого набрать в командной строке команду PS C:\ Documents and Settings\user> Get–Help Get–process –Full. Ознакомиться с перечнем характеристик этих процессов.
5. Просмотреть справку по командлету Get–Process, набрав команду PS C:\Documents and Settings\user>Get–process / ? Сравнить ее с предыдущими данными.

1.4. Командлеты

Оболочка PowerShell поддерживает команды четырех типов: командлеты, функции, сценарии и внешние исполняемые файлы.

Командлеты — особый вид команд, очень похожих на внутренние команды традиционных оболочек. Отличительной особенностью командлетов является то, что их имя служит обращением к объектам базового класса Cmdlet платформы .NET. В поставку Windows PowerShell включены более 120 командлетов, каждый из которых предназначен для выполнения достаточно простых функций. Организация командлетов такова, что в любое время можно расширить их состав, не изменяя структуры оболочки. Объединение командлетов в одном классе обеспечивает их единый синтаксис и единые принципы построения. Композиции этих функций при составлении конвейеров, в которых результаты действия одного командлета передаются другому, являются мощным средством анализа и управления ресурсами компьютерных систем. Администраторы-профессионалы с помощью пакета Software Developers Kit (SDK) могут разрабатывать собственные командлеты, расширяя стандартную поставку PowerShell.

Для всех командлетов принят общий принцип их именования в виде глагола и существительного, например Get–Help, Set–Service. Здесь глагол определяет запланированное действие, а существительное — объект, над которым это действие выполняется. Приставка Get предполагает отображение текущей информации на экране монитора об объекте, а Set — изменение режимов или состояний объекта-ресурса. Командлеты Set–* способны коренным образом изменять состояния ресурсов и режимы их работы. Поэтому при их использовании возникают опасения, связанные с безопасностью систем. Большинство командлетов поддерживают так называемый прототипный режим [2], согласно которому сначала просчитывается действие командлета, затем идет уведомление пользователя о предполагаемых изменениях и запрашивается подтверждение (Confirm) на действительное выполнение этих действий.

В общем случае формат командлетов имеет следующую структуру:

имя_командлета –параметр1 –параметр2 аргумент1 аргумент2

где –параметр1 — параметр, не имеющий значения (подобные параметры часто называют переключателями); –параметр2 — параметр, имеющий значение, записанное в поле аргумент; аргумент2 — параметр, не имеющий имени (или просто аргумент).

Примеры использования полей параметров и аргументов приведены в подразделе 1.4.2. Из структуры приведенного формата команд-

лета видно, что задание параметров с помощью слеша /, принятого в оболочке cmd.exe, не используется.

Некоторые параметры поддерживаются практически всеми командлетами (табл. 1.1).

Таблица 1.1

Общие параметры командлетов

Параметр	Тип	Действие
—Verbose	Boolean	Выводит подробные сведения об операциях, таких как результаты мониторинга или журналирование транзакций. Этот параметр эффективен в командлетах, формирующих подробные данные
—Debug	Boolean	Создает подробный отчет об операциях на уровне программирования. Используется в командлетах, создающих данные отладки
—ErrorAction	Enum	Отображает реакцию командлета на возникновение ошибки
—ErrorVariable	String	В дополнение к переменной $error, определяет переменную, сохраняющую ошибки команды при выполнении
—OutVariable	String	Определяет переменную, сохраняющую выходные данные команды при выполнении
—OutBuffer	Int32	Ограничивает количество хранящихся в буфере объектов перед вызовом следующего командлета в конвейере
—WhatIf	Boolean	Предупреждает об изменениях в состоянии системы, которые неизбежно произойдут при выполнении командлета
—Confirm	Boolean	Запрашивает разрешение на выполнение действий, вносящих изменения в систему

Даже имея только начальные сведения о построении PowerShell, можно убедиться, что заложенные в оболочке возможности значительно превышают возможности и удобства работы командной строки cmd.exe, а также графической оболочки Windows.

1.4.1. Работа с дисками

Возможно, что эту часть главы правильнее было бы назвать «Работа с хранилищами данных», однако ни один литературный источник не использует этого термина.

Одним из важнейших ресурсов компьютерных систем является ресурс памяти. С появлением многоядерных микропроцессоров значение этого ресурса еще более возрастает и выходит на передний план.

Фундаментальным положением любой ОС является управление данными, осуществляемое со стороны файловой системы. Файловая система представляет собой дерево вложенных каталогов (папок) и файлов. В командной оболочке PowerShell понятия диска, файла и папок значительно расширены и практически эквивалентны одноименным понятиям Unix- и Linux- ОС. В качестве файлов могут выступать не только данные, находящиеся на внешних носителях, но и физические и логические устройства, диски, их разделы и т.п., что значительно упрощает работу ОС и позволяет средствами файловых систем контролировать работу любых хранилищ данных, как локальных, так и сетевых. Кроме того, используя в качестве псевдонимов названия управляющих операторов ОС, отличных от Windows, можно управлять различно организованными данными.

В каждом сеансе работы пользователю необходимо знать, какие ресурсы памяти не только его компьютера, но и сетевых хранилищ ему доступны. Объем получаемой информации о ресурсах может быть очень большим. Список дисков, доступных пользователю из среды PowerShell, можно получить командой PS C:\> Get-PSDrive (рис. 1.9).

```
PS C:\Documents and Settings\user> Get-PSDrive

Name           Provider      Root                                 CurrentLocation
----           --------      ----                                 ---------------
Alias          Alias
C              FileSystem    C:\                                  ...d Settings\user
cert           Certificate   \
D              FileSystem    D:\
E              FileSystem    E:\
Env            Environment
Function       Function
HKCU           Registry      HKEY_CURRENT_USER
HKLM           Registry      HKEY_LOCAL_MACHINE
Variable       Variable
```

Рис. 1.9

Информация о доступных хранилищах является исходной для работы с ресурсом памяти. По команде PS C:\>Get–PSDrive сообщается точное обозначение диска (Root), его имя (Name), имя провайдера (Provider), поддерживающего этот диск, и текущая локализация (CurrentLocation). Кроме этих данных указываются доступные функции, псевдонимы, переменные окружения, разделы реестра и т.п. В PowerShell встроены специфические провайдеры, обеспечивающие доступ к специальным хранилищам: Alias — для доступа к псевдонимам PowerShell, Certificate — для использования сертификатов X509 цифровой подписи, Environment — для переменных среды Windows, FileSys-

tem — для обращения к файловой системе, Function — для обращения к функциям PowerShell Registry — для обращения к реестру Windows (ветви реестра HKCU-текущего пользователя и HKLM-локальной машины), Variable — для переменных PowerShell (см. рис. 1.9).

Провайдер PowerShell — это NET-приложение, предоставляющее пользователям оболочки доступ к хранилищам в едином формате, напоминающем формат обычных дисков файловой системы. Работа с представленными дисками практически ничем не отличается от работы с обычной файловой системой. Навигация по различным дискам, просмотр их содержимого, обращение к элементам данных выполняется с помощью привычных команд (командлеты Get–Location и Set–Location с их псевдонимами pwd, cd, chdir, sl).

Все перемещения по дискам осуществляются командлетом Get-Location или с помощью его псевдонима cd (chdir — полное имя), как и в файловой системе Windows с использованием интерпретатора cmd.exe. Файловая система в PowerShell управляет только физическими и логическими дисками (C:\, D:\, E:\). Все остальные хранилища управляются собственными провайдерами.

Сама файловая система контролирует только часть пространства, именуемого дисками. Очевидно, что возможности PowerShell гораздо обширнее программы «Проводник» ОС Windows. Список всех провайдеров оболочки может быть получен командлетом Get–PSProvider (рис. 1.10).

```
PS C:\Documents and Settings\user> Get-PSProvider

Name                 Capabilities                  Drives
----                 ------------                  ------
Alias                ShouldProcess                 {Alias}
Environment          ShouldProcess                 {Env}
FileSystem           Filter, ShouldProcess         {C, D, E}
Function             ShouldProcess                 {Function}
Registry             ShouldProcess                 {HKLM, HKCU}
Variable             ShouldProcess                 {Variable}
Certificate          ShouldProcess                 {cert}
```

Рис. 1.10

Навигация по дискам PowerShel ничем не отличается от типичной работы файловой системы. Здесь также сохраняется понятие рабочего или текущего каталога. Путь к этому каталогу устанавливает командлет Get–Location (псевдоним cd) без параметров (рис. 1.11)

Нетрудно убедиться, что аналогом данного командлета является псевдоним pwd, выполняющий те же функции в Unix- и Linux-оболочках.

```
PS C:\Documents and Settings\user> Get-Location

Path
----
C:\Documents and Settings\user
```

Рис. 1.11

Создание новых дисков (хранилищ) не вызывает трудностей. Для примера решим следующую задачу. Создадим новый диск внутри папки user, который будет содержать каталог с именем Mycat (рис. 1.12).

```
PS C:\Documents and Settings\user> New-PSDrive -Name Mycat FileSystem -Root 'C:`
Documents and Settings\user'

Name        Provider      Root                                    CurrentLocation
----        --------      ----                                    ---------------
Mycat       FileSystem    C:\Documents and Settings\user
```

Рис. 1.12

Теперь достаточно снова набрать команду Get–PSDriver и убедиться, что появился новый диск, доступ к которому обеспечивает файловая система.

1.4.2. Работа с файловой системой

При работе с файловой системой пользователь должен уметь создавать каталоги (папки) и файлы, копировать их и перемещать по собственному желанию. В среде PowerShell для этого имеются все необходимые средства.

Изучение любой файловой системы обычно начинают с команд, обеспечивающих получение списка каталогов и файлов. В командной строке интерпретатора cmd.exe такой командой служит команда tree. В PowerShell для этих целей предназначен командлет Get-ChildItem и его псевдоним dir, более привычный для пользователей персональных компьютеров. Отличительной особенностью новых средств является их расширенная функциональность.

Рассмотрим несколько типовых примеров их применения. Использование параметра –Recurse (рекурсия) позволяет отразить содержание любого диска с его каталогами и подкаталогами. Команда

```
>dir 'Documents and Settings' –Recurse
```

позволяет просмотреть полное содержимое каталога Documents and Settings.

По умолчанию командлет не отображает скрытые файлы. Если требуется включить в рассмотрение и их, то в команду следует вставить параметр –Force.

В случаях когда необходимо ограничиться только списком каталогов и подкаталогов, можно задать конвейер, в котором второй командлет Whre-Object со свойством PSIsContainer отфильтрует файлы (рис. 1.13).

```
PS C:\Documents and Settings\user> dir 'c:\program files'|Where-Object {$_.PSIsC
ontainer}

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\program files

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          20.11.2008     15:57            7-Zip
d----          20.11.2008     15:58            Adobe
d----          20.11.2008     15:04            ATI Technologies
d----          29.10.2009     13:17            Business Objects
d----          21.11.2008      9:58            CDBurnerXP
d----          29.10.2009     12:44            CE Remote Tools
d----          29.10.2009     12:44            Common Files
```

Рис. 1.13

Рассмотрим пример, иллюстрирующий использование большего числа полей в формате командлета. Пусть, например, в каталоге c:\windows требуется найти все exe-файлы, имена которых начинаются буквой n, а оканчиваются буквой d (шаблон n*d.exe). Решением этой задачи может служить команда

dir –Recurse –Filter n*d.exe –Path c:\windows

Учитывая, что синтаксис оболочки PowerShell не критичен в отношении прописных и строчных букв, а также допускает сокращение слов до нескольких символов, обеспечивающих однозначное понимание терминов, запись можно сократить:

dir –r –fi n*d.exe c:\windows

В этом выражении псевдоним dir заменяет имя командлета Get-ChildItem, переключатель –r является сокращением –Recurse. Этот переключатель распространяет действие команды не только на указанный каталог, но и на все его подкаталоги. Параметр фильтр –fi (–Filter) с аргументом n*d.exe задает маску файлов для поиска. Еще один параметр –Path с аргументом c:\windows определяет путь к исследуемому каталогу. Имя параметра с ключевым словом –Path можно не записывать, если по контексту выражение не имеет других значений.

Результатом выполнения команды будет информация о двух файлах: файле notepad.exe (блокнот) и файле ntsd.exe (встроенный 32-разрядный отладчик) (рис. 1.14).

```
PS C:\Documents and Settings\user> dir -r -fi n*d.exe c:\windows

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\windows

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         14.04.2008     21:41      69120 NOTEPAD.EXE

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\windows\system32

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         14.04.2008     21:41      69120 notepad.exe
-a---         07.07.2003     20:00      31744 ntsd.exe
```

Рис. 1.14

Другой важной процедурой файловой системы PowerShell является создание новых каталогов и файлов. Эти функции выполняет командлет New–Item. В качестве параметров требуется указать место размещения создаваемого объекта. Для этого в параметре –Path прописывается полный путь к каталогу, в котором создается объект, а в параметре –Type указывается тип объекта — «directory» или «file». Создадим в корневом каталоге диска C:\ новый каталог с именем test_folder (рис. 1.15).

```
    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         06.01.2010     14:59            test_folder

PS C:\> _
```

Рис. 1.15

Результат выполнения команды появляется на экране сразу после ввода команды. В некоторых случаях желательно имя файла определить как дату в формате ГГММДД (%y%m%d). Тогда нужно следующую команду, приведенную на рис. 1.16.

Если в имени файла требуется прописать дату создания в формате ГГГГММДД, то значение параметра –uformat надо записать в виде '%Y%m%d'.

```
PS C:\> New-Item -Path C:\ -Name "$(Get-Date -uformat '%y%m%d')" -Type "director
y"

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         06.01.2010     15:15            100106
```

Рис. 1.16

Войдем в созданный каталог test_folder, выполнив команду cd test_folder (рис. 1.17). Объявим и создадим текстовый файл test_file.txt, в который в качестве заголовка внесем название Test (значение «Test» для параметра –Value). Поскольку в параметре –Path путь к файлу не указан (можно опустить и название параметра), то по умолчанию файл создается в текущем каталоге. После того как файл создан, прочитаем содержимое файла, подав команду type test_file.txt. Команда type является псевдонимом командлета Get–Content.

```
PS C:\> cd test_folder
PS C:\test_folder> New-Item -Path test_file.txt -Type "file"  -Value "Test"

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\test_folder

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         06.01.2010     15:34          4 test_file.txt

PS C:\test_folder> type test_file.txt
Test
```

Рис. 1.17

Копирование файлов в PowerShell практически ничем не отличается от копирования в других командных оболочках. Копирование осуществляется с помощью командлета Rename–Item, имеющего псевдоним ren. Создадим в папке test_folder файл 1.tmp, сделаем ему копию (файл 7.tmp). Для того чтобы сразу видеть результат, укажем параметр –PassThru (рис. 1.18).

Для контроля проведенных работ можно использовать команды-псевдонимы dir и type.

Результаты действий отдельных команд можно записывать в файлы, используя перенаправление вывода данных. Знак > удаляет старое содержимое файла, заменяя его новым, знаки >> дописывают новые данные в конец указанного файла.

```
PS C:\test_folder> New-Item 1.tmp -Type "file" -Value "Test"  -Force

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\test_folder

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         06.01.2010     17:00          4 1.tmp

PS C:\test_folder> ren 1.tmp 7.tmp -PassThru

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\test_folder

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         06.01.2010     17:00          4 7.tmp
```

Рис. 1.18

Удаление каталогов (псевдоним rd) и файлов (псевдоним del) выполняется с помощью командлета Remove-Item. После имени командлета или псевдонима, его заменяющего, прописывается путь к удаляемому объекту и параметры: –Include, значение которого задает шаблон удаляемых файлов, например *.txt, и –Exclude, указывающий, на какие файлы команда не должна действовать, например *.ps1. Так, команда удаления файлов созданной папки test_folder со всеми вложенными подкаталогами может иметь вид PS C:\>del c:\test_folder* –Recurse.

Задания для самостоятельной работы

1. Просмотреть, какие диски (хранилища) доступны пользователю компьютера.
2. Просмотреть содержимое дисков, провайдеры которых не затрагивают файловую систему.
3. Создать один-два подкаталога в текущем каталоге пользователя.
4. Создать новый диск, обеспечивающий обращение к одному из созданных подкаталогов.
5. В одном из подкаталогов создать два новых текстовых файла. В одном файле в качестве заголовка ввести фамилию, в другом — имя студента. Сделать копии файлов. Убедиться в точном выполнении работ.
6. Уничтожить созданные объекты.

1.4.3. Работа с конфигурацией оболочки

В качестве простейшего примера работ в среде PowerShell рассмотрим, каким образом можно изменять ее параметры, обеспечивая комфортность работы пользователя. По своему желанию пользователь может устанавливать размеры и расположение окна PowerShell, характеристики используемых шрифтов, выбор цвета и другие параметры.

Проще всего эти установки выполняются с помощью диалогового окна оболочки. Для его вызова нужно установить курсор мыши на заголовке окна, кликнуть правой кнопкой и выбрать пункт Свойства в появившемся контекстном меню. Должно появиться диалоговое окно (рис. 1.19). Окно имеет несколько вкладок (Общие, Шрифт, Расположение, Цвета), каждая из которых позволяет настраивать определенную группу параметров. Здесь возможно буферирование и запоминание интерактивно выполняемых команд с целью их последующего использования отдельным блоком в сценарии.

После выбора и установки всех нужных параметров следует нажать кнопку OK. Система тут же потребует указаний, к какому объекту применить эти изменения (рис. 1.20). Если изменения свойств должны действовать постоянно, то следует выбрать переключатель **Сохранить свойства для других окон с тем же именем**, если изменения предусмотрены как разовые, то выбирается **Изменить свойства только текущего окна.**

Кроме инструментальных средств настройки командного окна имеется возможность применения чисто программных средств, являющихся неотъемлемой частью самой оболочки.

По умолчанию командлет Get–Host (рис. 1.21) без параметров отображает информацию о самой оболочке (региональные настройки, версия и т.п.).

В команде (Get–Host).UI имя командлета взято в круглые скобки. Это обозначает, что требуется выполнить данный командлет и сформировать выходной объект. Только после этого извлекается свойство объекта UI. Пройдя эту цепочку, получаем доступ к параметрам командного окна (рис. 1.22):

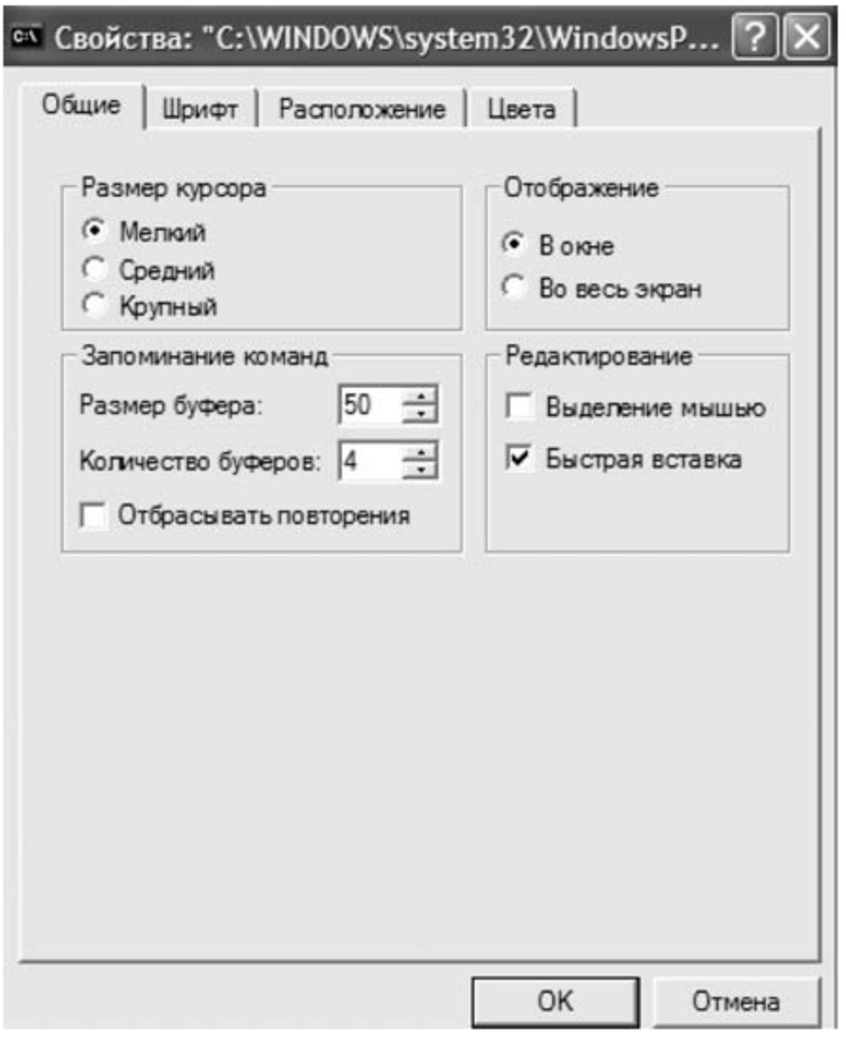


Рис. 1.19

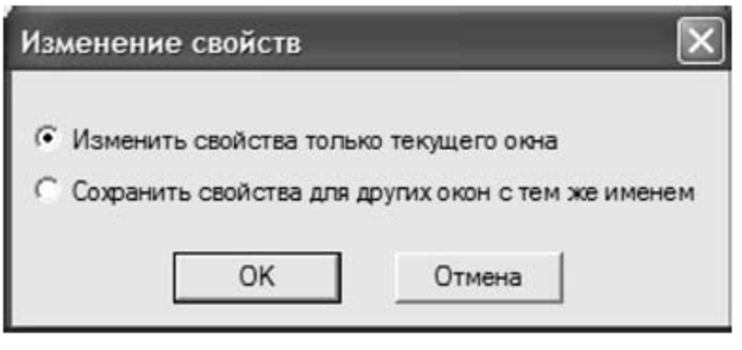


Рис. 1.20

```
PS C:\Documents and Settings\user> Get-Host

Name             : ConsoleHost
Version          : 1.0.0.0
InstanceId       : 1ebee0f3-4ac3-4e80-942e-d5461d484e04
UI               : System.Management.Automation.Internal.Host.InternalHostUserI
                   nterface
CurrentCulture   : ru-RU
CurrentUICulture : ru-RU
PrivateData      : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy


PS C:\Documents and Settings\user> (Get-Host).UI

RawUI
-----
System.Management.Automation.Internal.Host.InternalHostRawUserInterface
```

Рис. 1.21

```
PS C:\Documents and Settings\user> (Get-Host).UI.RawUI


ForegroundColor         : Gray
BackgroundColor         : Black
CursorPosition          : 0,55
WindowPosition          : 0,41
CursorSize              : 25
BufferSize              : 80,300
WindowSize              : 80,33
MaxWindowSize           : 80,62
MaxPhysicalWindowSize   : 160,62
KeyAvailable            : False
WindowTitle             : C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.e
                          xe
```

Рис. 1.22

Значение каждого из перечисленных параметров можно изменить, настраивая внешний вид окна по желанию. При изменениях параметров целесообразно объект RawUI сохранить в качестве значения отдельной переменной [1]. Покажем, например, как изменить цвет фона и текста. Свойство BackgroundColor отвечает за цвет фона, а ForegroundColor — за цвет текста. В качестве цветов можно использовать следующие 16: Black, Gray, Red, Magenta, Yellow, Blue, Green, Cyan, White, DarkGreen, DarkCyan, DarkRed, DarkMagenta, DarkYellow, DarkGray, DarkBlue. Установим желтый цвет текста на темно-синем фоне. Такое сочетание цветов часто используется в различных системах программирования. Желаемый эффект обеспечивается выполнением трех команд, приведенных на рис. 1.23.

```
PS C:\Documents and Settings\user> $a=(Get-Host).UI.RawUI
PS C:\Documents and Settings\user> $a.BackgroundColor = "DarkBlue"
PS C:\Documents and Settings\user> $a.ForegroundColor = "Yellow"
PS C:\Documents and Settings\user> _
```

Рис. 1.23

Действие этих команд обеспечивается сразу после их выполнения. Оно не затрагивает строки, предшествующие этим командам.

Некоторые параметры оболочки, например WindowSize, содержат по две координаты. Для изменения их значений проще всего объявить новую переменную $b=$a.WindowSize. Переопределим значения ширины и высоты командами $b.Width=80 и $b.Height=25, а затем изменим содержимое объекта WindowSize переменной $a, т.е. выполним команду $a.WindowSize=$b.

Последняя строка, изображенная на рис. 1.22, отражает заголовок командного окна PowerShell. Оно достаточно длинное и непривлекательное. Выполнив команду $a.WindowTitle=« Мое командное окно», получаем более приемлемое название (рис. 1.24).

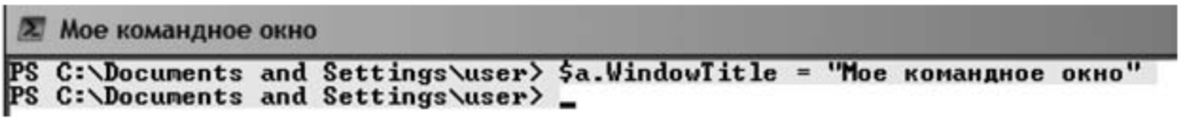


Рис. 1.24

В некоторых случаях необходимо изменить приглашение. Мигающему курсору после букв PS предшествует запись полного пути к текущему каталогу. Вид приглашения командной строки в PowerShell определяется функцией Prompt, которая имеет формат, отражаемый командой, приведенной на рис. 1.25.

```
PS C:\Documents and Settings\user> (Get-Item Function:Prompt).Definition
'PS ' + $(Get-Location) + $(if ($nestedpromptlevel -ge 1) { '>>' }) + '> '
```

Рис. 1.25

Не рассматривая подробно составные части формата, так как это можно сделать, используя справочную систему, укажем, как создать приглашение, эквивалентное командной строке cmd.exe (рис. 1.26).

```
PS C:\Documents and Settings\user> Function Prompt{"$(Get-Location) > "}
C:\Documents and Settings\user >
```

Рис. 1.26

В данной команде используется конструкция «$(Get–Location)..., называемая подвыражением (subexpression). Подвыражение — это блок кода на языке PowerShell, который в строке заменяется значением, полученным в результате выполнения этого кода.

Все иллюстрированные выше настройки выполнялись в интерактивном режиме, и их действие распространяется лишь на время текущего сеанса работы. После окончания сеанса работы в оболочке PowerShell они утрачивают силу.

Для сохранения настроек с целью их регулярного, а может быть, и повседневного использования необходимо создать файл с соответствующим набором команд-настроек. Этот файл текстового типа получил название профиль. Профиль — это сценарий, который будет загружаться и активизировать необходимые настройки при каждом запуске оболочки PowerShell. Значение профиля очень близко значениям файлов autoexec.bat — для ранних и autoexec.nt — для современных версий ОС Windows. Все они предназначены для автоматического выполнения требуемых подготовительных работ. Корректно составленный профиль призван обеспечить не только комфортные условия работы пользователя, но и создать удобства для администрирования. Разработка и распространение профилей позволяет использовать единые условия работы пользователей на группе компьютеров в распределенной среде, например в локальных компьютерных сетях.

В зависимости от уровня выполняемых настроек и значимости администрируемых ресурсов можно формировать профили четырех видов:

1) действующие на всех пользователей сети и на все их оболочки PowerShell (хосты);
2) действующие на всех пользователей сети с использованием единой оболочки PowerShell;
3) действующие только на текущего пользователя и на все оболочки;
4) действие которых распространяется только на текущего пользователя и только на хост powershell.exe.

Видимо, поэтому в инструментальных средствах оболочки (см. рис. 1.19. Свойства, вкладка. Общие) предусмотрена возможность работы с несколькими буферами, в которых одновременно подготавливаются блоки команд сразу для нескольких профилей.

Каждый тип профиля имеет свое место хранения в особой зоне ОС Windows. При работе с оболочкой на автономном компьютере используется только пользовательский профиль, относящийся к последнему типу. Место его расположения и имя файла можно определить по значению специальной переменной $profile (рис. 1.27).

```
PS C:\Documents and Settings\user> $profile
C:\Documents and Settings\user\Мои документы\WindowsPowerShell\Microsoft.PowerS
hell_profile.ps1
```

Рис. 1.27

С помощью командлета Test–Path можно установить факт наличия созданного профиля (рис. 1.28).

```
PS C:\Documents and Settings\user> Test-Path $profile
False
```

Рис. 1.28

Если профиль создан, то программа отобразит True, в противном случае — False.

По умолчанию, даже если профиль создан, выполнение блока его команд запрещено. Параметр политики выполнения устанавливается Restricted. Это сделано для обеспечения безопасности системы.

Автоматическое выполнение сценариев профиля с цифровой подписью разрешается при параметрах политики выполнения AllSigned и RemoteSigned. Возможно выполнение сценариев и без цифровой подписи при параметре Unrestricted. В этом случае система выдает предупреждение и требует подтверждения со стороны пользователя. Узнать, какая политика является активной, можно с помощью команды, изображенной на рис. 1.29.

```
PS C:\Documents and Settings\user> Get-ExecutionPolicy
Restricted
```

Рис. 1.29

Переустановка значения параметра обеспечивается командлетом SetExecutionPolicy, и далее через пробел пишется название политики выполнения, например RemoteSigned. Запоминать имена возможных политик нет необходимости. Установив курсор после имени командлета и нажимая клавишу Tab, можно перебрать все варианты политик выполнения, оставив необходимую.

Задания для самостоятельной работы

1. Вызвать диалоговое окно оболочки для проверки и установки параметров командного окна. Внимательно изучить рубрики вкладок Общие, Шрифт, Расположение, Цвета. Просмотреть, как изменение различных параметров отражается на внешнем виде командного окна.
2. Изучить последовательность команд для программного изменения свойств консоли. По образцу и подобию установить

новые размеры командного окна, применить другие сочетания цветов.

3. Изменить название командного окна.

1.4.4. Работа с объектами

Оболочка PowerShell относится к объектно-ориентированной среде, в которой все действия совершаются над объектами — ресурсами компьютерных систем. Каждый объект в общем случае включает совокупность данных, отражающих свойства объекта, и модули обработки этих данных (методы объекта).

Фундаментом оболочки следует считать платформу Microsoft .NET, так как она изначально предназначалась для разработки различных информационных систем. В составе этой платформы уже заложен набор сетевых служб и серверов, что позволяет создать программный образ функционирования любой системы. Часть этой платформы Microsoft .NET Framework предназначается для разработки приложений. Она дает большие преимущества для разработчиков программ в части использования различных систем программирования.

Самым мощным достоинством платформы Microsoft .NET служит наличие в ней обширной библиотеки классов (тысячи классов), содержащих готовые, отлаженные и постоянно пополняемые методы обработки. Мощнейшим средством обработки данных в оболочке PowerShell является использование конвейеров. Конвейер — последовательность команд, разделенных вертикальной чертой (|), в которой результаты обработки одной команды передаются другой команде. В оболочке PowerShell по конвейеру передаются целые объекты, наборы свойств отформатированных данных с требуемой детализацией. Последовательный и пошаговый принцип обработки конвейера позволяет решать как задачи анализа сложных систем, так и на его базе строить новые информационные системы любой сложности.

Покажем на примерах, какой арсенал средств может применять пользователь в своей работе. Очень часто при анализе многопрограммной работы компьютера возникает задача определения, какие процессы и как используют ресурсы системы. Начальную информацию можно получить, включив командлет Get–Process, который отражает часть данных о всех активных процессах. Результатом является список из нескольких десятков запущенных в системе процессов.

Каждая запись содержит набор параметров-характеристик процессов. Некоторые из них интуитивно понятны (ProcessName — имя процесса, CPU(s) — время, затрачиваемое на работу процесса в секундах), другие имеют не всегда понятные сокращения (рис. 1.30).

```
PS C:\Documents and Settings\user> Get-Process

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    107       5     1160       3556    32     0,16   3164 alg
     76       3      816       3264    26     0,09    196 ati2evxx
     67       2      600       2452    20     0,05   1136 ati2evxx
     46       2      916       2604    32     0,23    536 BatteryManager
    166       7     4236       6264    51     0,09    748 BTStackServer
```

Рис. 1.30

Для понимания смысла всех сокращений и выяснения полной структуры объекта целесообразно использовать конвейер двух командлетов Get-Process|Get-Member. Для сокращения длины команды и уменьшения трудоемкости набора можно записать конвейер из псевдонимов этих командлетов — gps|gm (рис. 1.31).

```
PS C:\Documents and Settings\user> gps|gm

   TypeName: System.Diagnostics.Process

Name                        MemberType     Definition
----                        ----------     ----------
Handles                     AliasProperty  Handles = Handlecount
Name                        AliasProperty  Name = ProcessName
NPM                         AliasProperty  NPM = NonpagedSystemMemorySize
PM                          AliasProperty  PM = PagedMemorySize
VM                          AliasProperty  VM = VirtualMemorySize
WS                          AliasProperty  WS = WorkingSet
add_Disposed                Method         System.Void add_Disposed(Event...
add_ErrorDataReceived       Method         System.Void add_ErrorDataRecei...
```

Рис. 1.31

На экран выводится весь перечень свойств процессов. Перед списком указывается, к какому .NET-типу относятся все названия свойств (System.Diagnostics.Process). Если требуется только вывести определенную категорию свойств, то следует задать значение типа параметра (рис. 1.32).

В оболочке PowerShell имеется несколько конфигурационных файлов, необходимых для отображения объектов различных типов. Они находятся в том же каталоге, что и powershell.exe и имеют названия, заканчивающиеся *format.ps1xm1 (рис. 1.33).

Второй файл в этом списке dotnettypes.format.ps1xm1 предназначается для форматирования объектов System.Diagnostics.Process (см. рис. 1.33).

```
PS C:\Documents and Settings\user> gps|gm -MemberType Property

   TypeName: System.Diagnostics.Process

Name                       MemberType Definition
----                       ---------- ----------
BasePriority               Property   System.Int32 BasePriority {get;}
Container                  Property   System.ComponentModel.IContainer Conta...
EnableRaisingEvents        Property   System.Boolean EnableRaisingEvents {ge...
```

Рис. 1.32

```
PS C:\WINDOWS\system32\WindowsPowerShell\v1.0> dir

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\WINDOWS\system32\WindowsP
    owerShell\v1.0

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          29.10.2009     15:46            examples
d----          29.10.2009     15:47            ru
-----          09.12.2007     15:34      22120 certificate.format.ps1xml
-----          09.12.2007     15:34      60703 dotnettypes.format.ps1xml
-----          09.12.2007     15:34      19730 filesystem.format.ps1xml
-----          09.12.2007     15:34     250197 help.format.ps1xml
-----          25.03.2008     11:09     330240 powershell.exe
-----          09.12.2007     15:34      65283 powershellcore.format.ps1xml
-----          09.12.2007     18:04      13394 powershelltrace.format.ps1xml
-----          13.07.2007      0:47       4608 pwrshmsg.dll
-----          26.03.2008      4:55      20992 pwrshsip.dll
-----          09.12.2007     15:34      13540 registry.format.ps1xml
-----          09.12.2007     15:35     129836 types.ps1xml
```

Рис. 1.33

Приведенные примеры показывают, что количество выводимых данных может быть огромно, и требуется иметь средства, позволяющие отсеивать ненужную информацию и выделять требуемую. Обычно для этого используются процедуры сортировки и фильтрации. Именно сортировка призвана служить сокращению числа операций перебора при поиске требуемых данных. Создание рабочих массивов данных, упорядоченных по определенному параметру, позволяет использовать методы дихотомии согласно зависимости

$$n = \log_2 N,$$

где n — число проб, в результате которых находятся данные в отсортированном массиве; N — количество элементов в исходном анализируемом массиве.

За операции сортировки отвечает командлет Sort–Object. В качестве его параметра указываются имена свойств, по которым упорядочиваются объекты. Выведем список процессов, упорядоченный в соответствии с требуемым процессорным временем (рис. 1.34).

Для получения списка, упорядоченного в порядке убывания процессорного времени, должен быть включен параметр –descending (рис. 1.35).

```
PS C:\> gps|Sort-Object -property cpu

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
      0       0        0         16     0                0 Idle
     70       3      700       2304    14       0,02  1780 sqlbrowser
     60       2      956       2816    30       0,02  1816 SNMWLANService
     30       2      628       2580    26       0,02   664 Mctray
     90       4     1432       1056    37       0,03  1624 naPrdMgr
     84       2      948       3540    20       0,03  1620 sqlwriter
     29       2      544       1836    22       0,03   756 NMSAccessU
    116       4     1256       3528    41       0,03   556 UdaterUI
     19       1      168        400     4       0,03   812 smss
```

Рис. 1.34

```
PS C:\> gps|Sort-Object -property cpu -descending

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    335      13    32608      50632   213   104,59   4040 WINWORD
     97       4     1556       5124    39    56,78    528 SynTPEnh
    354      15    60252      65296   269    38,78   1480 mcshield
    227      10     7436        464    68    21,48   1532 vstskmgr
     55       2      744        596    28    17,31    876 spider
    576       6     1752       1600    25    17,17    872 csrss
   1488      70    15208      24288   109    11,09   1380 svchost
    286       0        0        228     2     9,20      4 System
```

Рис. 1.35

Часто в отсортированном списке наибольший интерес представляют записи, имеющие максимальные или минимальные значения некоторых параметров. Для выявления подобных объектов в списке командлетов имеется Select–Object, который позволяет в отсортированном списке отбирать несколько первых (–First) или последних (–Last) записей. Например, для выявления пяти процессов, использующих наибольшие объемы памяти (свойство WS), можно сформировать команду-конвейер, состоящую из трех командлетов (рис. 1.36).

```
PS C:\> gps|Sort-Object WS|Select-Object -Last 5

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    118       4    16880      18640    58     5,56   3384 mspaint
    402      15    12968      19664    88     3,59    324 explorer
   1476      70    15000      24300   108    11,45   1380 svchost
    335      13    33512      50780   216   117,61   4040 WINWORD
    354      16    61784      66528   270    39,00   1480 mcshield
```

Рис. 1.36

Для фильтрации данных обычно используют командлет Where–Object. Например, для определения данных об остановленных службах (свойство Status равно «Stopped») следует сформировать команду, приведенную на рис. 1.37.

Появляется информация о нескольких десятках остановленных службах. Следует обратить внимание, что при сравнении свойств

в фигурных скобках командлета не используются знаки =, <, >, а операторы сравнения задаются мнемоническими сокращениями (табл. 1.2).

```
PS C:\> Get-Service|Where-Object {$_.Status -eq "Stopped"}

Status   Name               DisplayName
------   ----               -----------
Stopped  Alerter            Оповещатель
Stopped  AppMgmt            Управление приложениями
Stopped  aspnet_state       Служба состояний ASP.NET
Stopped  BITS               Фоновая интеллектуальная служба пер...
```

Рис. 1.37

Таблица 1.2

Операторы сравнения в PowerShell

Оператор	Значение	Пример (возвращается значение True)
–eq	Равно	10 –eq 10
–ne	Не равно	9 –ne 10
–lt	Меньше	3 –lt 4
–le	Меньше или равно	3 –le 4
–gt	Больше	4 –gt 3
–ge	Больше или равно	4 –ge 3
–like	Сравнение на совпадение с учетом подстановочного знака во втором операнде	«file.doc» –like «f*.doc»
–notlike	Сравнение на несовпадение с учетом подстановочного знака во втором операнде	«file.doc» –notlike «f*.rtf»
–contains	Содержит	1, 2, 3 –contains 1
–notcontains	Не содержит	1, 2, 3 –notcontains 4

После выделения требуемой информации и отсева ненужной возникают задачи определения характеристик выделенных объектов. Применение некоторых командлетов позволяет решать часть из них.

Одной из типовых задач является определение суммарного объема некоторой группы файлов. Подсчитаем объем памяти с точностью до байта, занимаемый каталогом, например \user. Решение этой задачи можно обеспечить последовательностью команд, изображенных на рис. 1.38.

Первая строка формирует переменную $TotalLength со значением нуль, вторая строка представлена конвейером из двух командлетов,

подсчитывающим суммарный объем памяти. Командлет ForEach-Object обеспечивает циклическое накопление суммы. Третья строка считывает полученный итог. Эту же задачу в более расширенном функциональном формате можно решить, используя командлет Measure-Object (рис. 1.39).

```
PS C:\Documents and Settings\user> $TotalLength=0
PS C:\Documents and Settings\user> dir¦ ForEach-Object {$TotalLength+=$_.Length}

PS C:\Documents and Settings\user> $TotalLength
9023033
```

Рис. 1.38

```
PS C:\Documents and Settings\user> dir¦ Measure-Object -Property Length -Sum


Count    : 17
Average  :
Sum      : 9023033
Maximum  :
Minimum  :
Property : Length
```

Рис. 1.39

Этот вариант может дать больше расчетных данных об объекте. Достаточно указать, какие характеристики интересуют пользователя (рис. 1.40).

```
PS C:\Documents and Settings\user> dir¦ Measure-Object -Property Length -Sum -Mi
nimum -Maximum -Average


Count    : 17
Average  : 530766,647058824
Sum      : 9023033
Maximum  : 7825818
Minimum  : 312
Property : Length
```

Рис. 1.40

Следует помнить, что все командлеты имеют строго ограниченную функциональность, поэтому используются только для решения узких, типовых задач. Творческий характер анализа и обработки характеристик объектов в основном переносится в функции.

1.5. Функции

Набор командлетов оболочки PowerShell вполне можно отнести к языку запросов. В терминах СУБД Microsoft Access они обеспечивают запросы-выборки, в которых имя командлета указывает на объект,

с которым работает пользователь, а вариация параметров является, по существу, инструкцией к тому, какие данные и в каком виде должны быть представлены в результате выполнения запроса. Комбинированное действие командлетов в виде конвейеров позволяет получить более сложные виды запросов: запросы с группировкой, перекрестные запросы, запросы с параметрами, запросы-действия и т.п.

Функциональность каждого командлета изменить нельзя, так как их программный код из оболочки не доступен. Только функции и сценарии позволяют формировать программный код, который пользователь создает по своему желанию. Обработка данных в зависимости от контекста работ и специфики аппаратных и программных ресурсов может осуществляться с различной степенью детализации.

Функция в PowerShell — блок кода, имеющий уникальное имя. Этот блок активизируется при первом к нему обращении и остается действительным до завершения текущего сеанса работы с оболочкой. Функции могут быть очень простыми — без параметров и очень сложными — с формальными и замещаемыми параметрами. Они могут включаться в конвейеры и возвращать значения не только некоторых переменных, а даже целых массивов переменных различного типа данных.

Ограниченный объем учебного пособия не позволяет глубоко ознакомить с особенностями встроенного языка программирования, но следует отметить, что функциональность языковых средств программирования очень высокая, и это позволяет создавать очень эффективные программы. Язык программирования оболочки PowerShell требует отдельного рассмотрения. Ниже будут приведены примеры построения функций различной сложности. Контекст программ, включаемый в функции, достаточно прост и особых пояснений не требует.

Для определения функции используется формат

Function Имя_функции {тело функции} [аргументы],

где Function — ключевое слово, которым объявляется новая функция; Имя_функции — присваиваемое уникальное имя; {тело функции} — набор операторов встроенного языка программирования, обеспечивающих обработку данных. Тело функции обязательно заключается в фигурные скобки; [аргументы] — набор аргументов и параметров функции. Квадратные скобки указывают, что аргументы и параметры не являются обязательным элементом, они могут отсутствовать.

На рисунке 1.41 приведен пример формирования простейшей функции без аргументов [1]. Создадим функцию MyFunc, формирующую текстовое сообщение:

```
PS C:\Documents and Settings\user> Function MyFunc {"Всем привет!"}
PS C:\Documents and Settings\user>
```

Рис. 1.41

Для активизации функции надо в командную строку записать ее имя (рис. 1.42).

```
PS C:\Documents and Settings\user> MyFunc
Всем привет!
```

Рис. 1.42

Аналогично можно создавать функции, извещающие пользователя о начале и прекращении каких-либо работ в системе.

В более сложных случаях функции могут использовать аргументы, которые передаются ей при запуске. Имеется два вида обработки аргументов: с помощью переменной $Args и путем задания формальных параметров. Рассмотрим оба варианта обработки аргументов.

В оболочке PowerShell имеется переменная $Args, которая в общем случае является массивом. Элементы массива могут быть параметрами функции, заданными при ее запуске. Переопределим предыдущую функцию таким образом, чтобы она могла принимать переменные значения (рис. 1.43).

```
PS C:\Documents and Settings\user> Function MyFunc {"Привет, $Args!"}
```

Рис. 1.43

Переменная $Args помещена в тело функции, заключенное в двойные кавычки. Это обозначает, что при запуске функции она примет значения аргументов и вставит их в строку результата (рис. 1.44).

```
PS C:\Documents and Settings\user> MyFunc Андрей Сергей Павел
Привет, Андрей Сергей Павел!
```

Рис. 1.44

Внутри переменной $Args, являющейся массивом, можно обращаться к элементам массива по их порядковому номеру. Например, если требуется подсчитать сумму нескольких чисел и знать их количество, то функцию SumArgs можно определить, как показано на рис. 1.45.

Смысл данной программы достаточно понятен. У этой функции может быть переменное число аргументов. Пусть требуется сложить пять слагаемых: 50, 14, 4, 7, 33 (рис. 1.46).

```
PS C:\> Function SumArgs {"Количество аргументов: $($Args.Count)"
>> $n=0
>> For ($i=0; $i -lt $Args.Count; $i++) { $n+=$Args[$i] }
>> "Сумма аргументов: $n"}
>>
```

Рис. 1.45

```
PS C:\> SumArgs 50 14 4 7 33
Количество аргументов: 5
Сумма аргументов: 108
```

Рис. 1.46

Подобную функцию вполне можно использовать для обработки чисел массивов, образующихся при выполнении некоторых командлетов и их конвейеров. Другим методом учета аргументов является задание формальных параметров функции, значения которых замещаются значениями аргументов. Это типовой прием во многих системах программирования.

Рассмотрим несколько примеров. Определим функцию, обеспечивающую сложение двух аргументов (рис. 1.47).

```
PS C:\Documents and Settings\user> Function Add ($x,$y) {$x+$y}
```

Рис. 1.47

При определении функции выражение в круглых скобках устанавливает порядок ввода и анализа переменных, а выражение в фигурных скобках формирует тело функции. По умолчанию эта функция, как и другая функция PowerShell, ведет себя полиморфным образом. Она учитывает и «приспосабливается» к желаниям пользователей. Просмотрим несколько вариантов работы функции с различными типами данных (рис. 1.48).

```
PS C:\Documents and Settings\user> Add 5 7
12
PS C:\Documents and Settings\user> Add "10" "20"
1020
PS C:\Documents and Settings\user> Add 1.5 2.4
3,9
PS C:\Documents and Settings\user> Add 1.5 4
5,5
PS C:\Documents and Settings\user> Add 4 1.5
6
PS C:\Documents and Settings\user> Add 5 1.4
6
```

Рис. 1.48

Примеры показывают, что программа по-разному себя ведет, принимая данные различных типов. Контекст выполнения функции

строится в зависимости от типа данных. Она может складывать целые числа, числа с плавающей точкой. Если аргументы имеют данные различных типов, то программа их приводит к типу первого слагаемого. Последние две строки демонстрируют правила округления и перевода вещественных чисел в целочисленную форму. Вторая строка показывает, что при «сложении» строковых данных включается конкатенация (соединение) строк (рис. 1.49).

Контекст выполнения функции можно отследить командой

```
PS C:\Documents and Settings\user> (Add 2 3).GetType().FullName
System.Int32
PS C:\Documents and Settings\user> (Add "2" "3").GetType().FullName
System.String
```

Рис. 1.49

Основные достоинства PowerShell заключаются в реализации различных конвейеров. Функции как средства обработки играют здесь очень важную роль. С их помощью можно перебирать, анализировать, фильтровать и обсчитывать элементы потоковой информации, а также разрабатывать новые командлеты. Покажем работу функции в конвейере при поступлении потока данных. Для передачи потоковых данных в PowerShell служит переменная $Input, которая предназначается для хранения коллекции входящих объектов.

Создадим функцию Sum, обеспечивающую суммирование элементов входящего потока [1] (рис. 1.50).

```
PS C:\> Function Sum {
>> $n=0
>> ForEach ($i in $Input) { $n+=$i }
>> $n
>> }
>>
```

Рис. 1.50

Создадим простейший входной поток из целых чисел от 1 до 10. В этом случае функция должна подсчитать сумму членов арифметической прогрессии (рис. 1.51).

```
PS C:\> 1..10|Sum
55
PS C:\> _
```

Рис. 1.51

Запуск конвейера формирует ответ: подсчитанная сумма равна 55.

1.6. Сценарии

Частично построение сценариев затрагивалось в подразделе 1.4.2 при формировании профилей. В общем случае сценарий представляет собой текстовый файл, содержащий набор команд управления в необходимой последовательности. Как и в любой системе программирования, для создания программ следует выбирать текстовый редактор, имеющий минимальное количество невидимых символов форматирования, например редактор Блокнот (Notepad). Файл со сценарием должен иметь расширение *.ps1. Создать файл со сценарием можно четырьмя различными способами:

1) вызвать текстовый редактор и вручную набрать в нужной последовательности команды управления. После окончания набора сохранить файл с нужным расширением;
2) выполнить необходимые команды в оболочке PowerShell, скопировать их с консоли в буфер Windows и вставить в текстовый файл, открытый во внешнем текстовом редакторе;
3) при работе в окне PowerShell включить с помощью командлета Start–Transcript режим протоколирования команд. После окончания сеанса выдать команду Stop–Transcript, что прекращает процесс формирования журнала.
4) при работе в оболочке PowerShell команды, подлежащие размещению в сценарии, оформляются в виде строковых данных (заключаются в одиночные апострофы) и с помощью символов > и >> одна за другой перенаправляются в файл сценария *.ps1. Символ > размещает строку в начале файла, а символы >>записывают строки в конец файла, например:

```
PS C:\>'Команда — строка, помещаемая в сценарий' >> *.ps1
```

Искусство написания сценариев требует большого опыта работы пользователя в качестве администратора компьютерных систем. Имеются даже библиотеки с сотнями сценариев, которые можно использовать в типовых вариантах управления (http://www.ecom.ru/catalog_18/fileData/file_23_4.zip).

Рассмотрим в качестве примера процесс подготовки простого файла-сценария и его выполнения. Пусть он будет выполнять действия, аналогичные функции, созданной в подразделе 1.5 (подсчет количества аргументов и суммы их значений). Пример одновременно продемонстрирует, как можно передавать значения аргументов в сценарии.

В каталоге пользователя создадим подкаталог PScript, в котором и сформируем требуемый файл (рис. 1.52).

```
PS C:\Documents and Settings\user> md PScript

    Каталог: Microsoft.PowerShell.Core\FileSystem::C:\Documents and Settings\user

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d-----        08.01.2010     12:35            PScript
```

Рис. 1.52

Войдем в созданный каталог и построчно введем знакомую программу. Знаки ‘@ указывают на начало и конец программы. Последняя строка программы присваивает файлу сценария имя SumArgs.ps1 (рис. 1.53).

```
PS C:\Documents and Settings\user> cd PScript
PS C:\Documents and Settings\user\PScript> @'
>> "Количество аргументов: $($Args.count)"
>> $n
>> for($i=0; $i -lt $Args.count; $i++) { $n+=$Args[$i] }
>> "Сумма аргументов: $n"
>> '@ > SumArgs.ps1
>>
```

Рис. 1.53

Попытка сразу выполнить файл сценария к успеху обычно не приводит. По соображениям безопасности здесь имеется несколько уровней защиты. Сначала надо проверить значение политики выполнения сценариев. По умолчанию она устанавливается Restricted (рис. 1.54). Может быть установлена и более строгая AllSigned.

```
PS C:\Documents and Settings\user\PScript> Get-ExecutionPolicy
Restricted
```

Рис. 1.54

В этих случаях требуется установить политику RemoteSigned. Именно она позволяет выполнять неподписанные локальные сценарии. Еще одной ступенью защиты является специфический запуск сценариев. При запуске сценария обязательно должен прописываться полный путь к файлу сценария *.ps1. Чтобы снизить трудоемкость набора длинных путей к файлу сценария, при его запуске из текущего каталога разрешается сокращенная запись пути к файлу. \SumArgs. Здесь знак точки (.) и соответствует текущему каталогу, как в Unix- и Linux-системах (рис. 1.55).

```
PS C:\Documents and Settings\user\PScript> Set-ExecutionPolicy RemoteSigned
PS C:\Documents and Settings\user\PScript> .\SumArgs 10 20 30 40
Количество аргументов: 4
Сумма аргументов: 100
PS C:\Documents and Settings\user\PScript>
```

Рис. 1.55

Сценарий выполнен.

1.7. Примеры работ в Windows PowerShell

Первое знакомство с новой командной оболочкой Windows PowerShell и полученные знания позволяют оценить универсальность и многогранность данного инструмента. Квалифицированное его использование требует глубоких знаний аппаратного, программного и информационного обеспечения компьютерных систем. Не вдаваясь в подробности построения и функционирования отдельных ресурсов, рассмотрим несколько интересных примеров определения некоторых характеристик компьютерных систем [1]. Нужно отметить, что отдельные управляющие конструкции — конвейеры — громоздки, требуют внимательности и терпения. Ошибки даже в одном символе недопустимы.

Получение информации о BIOS. Вывод всех характеристик BIOS можно получить выполнением команды, приведенной на рис. 1.56.

```
PS C:\> Get-WMIObject Win32_BIOS|Select-Object -Property *

Status                : OK
Name                  : Phoenix FirstBIOS(tm) Notebook Pro Version 2.0  09XA
Caption               : Phoenix FirstBIOS(tm) Notebook Pro Version 2.0  09XA
SMBIOSPresent         : True
__GENUS               : 2
__CLASS               : Win32_BIOS
__SUPERCLASS          : CIM_BIOSElement
__DYNASTY             : CIM_ManagedSystemElement
```

Рис. 1.56

Служебные характеристики для WMI (Windows Management Instrumentation), имена которых начинаются двумя знаками подчеркивания, можно убрать, если ввести параметр –ExcludeProperty __* (рис. 1.57).

```
PS C:\> Get-WMIObject Win32_BIOS|Select-Object -Property * -ExcludeProperty __*

Status                : OK
Name                  : Phoenix FirstBIOS(tm) Notebook Pro Version 2.0  09XA
Caption               : Phoenix FirstBIOS(tm) Notebook Pro Version 2.0  09XA
SMBIOSPresent         : True
BiosCharacteristics   : {4, 7, 8, 9...}
BIOSVersion           : {PTLTD   - 6040000, Phoenix FirstBIOS(tm) Notebook Pro V
                        ersion 2.0  09XA, Ver 1.00PARTTBL?}
...
```

Рис. 1.57

Вывод характеристик ОС. Список основных характеристик (дата установки, загрузочное устройство и т.п.) ОС можно получить при об-

ращении к экземпляру класса WMI Win32_OperatingSystem. Конвейер блокирует выдачу служебных свойств WMI (рис. 1.58)

```
PS C:\> Get-WMIObject Win32_OperatingSystem¦Select-Object -Property * -ExcludePr
operty __*


Status                                    : OK
Name                                      : Microsoft Windows XP Professional¦C
                                            :\WINDOWS¦\Device\Harddisk0\Partiti
                                            on1
FreePhysicalMemory                        : 136368
FreeSpaceInPagingFiles                    : 864860
FreeVirtualMemory                         : 2055676

...
```

Рис. 1.58

Получение информации о физической памяти компьютера. Экземпляры класса Win32_PhysicalMemory позволяют определить характеристики памяти компьютера (рис. 1.59).

```
__GENUS                 : 2
__CLASS                 : Win32_PhysicalMemory
__SUPERCLASS            : CIM_PhysicalMemory
__DYNASTY               : CIM_ManagedSystemElement
__RELPATH               : Win32_PhysicalMemory.Tag="Physical Memory 0"
__PROPERTY_COUNT        : 30
__DERIVATION            : {CIM_PhysicalMemory, CIM_Chip, CIM_PhysicalComponent, CI
                          M_PhysicalElement...}
__SERVER                : N296611
__NAMESPACE             : root\cimv2
__PATH                  : \\N296611\root\cimv2:Win32_PhysicalMemory.Tag="Physical
                          Memory 0"
BankLabel               : Bank 0
Capacity                : 536870912

...
```

Рис. 1.59

Оставив самые важные свойства, следующий конвейер дает табличное представление данных (рис. 1.60).

```
PS C:\> Get-WMIObject Win32_PhysicalMemory¦Format-Table BankLabel, Capacity, Des
cription

BankLabel                                  Capacity Description
---------                                  -------- -----------
Bank 0                                    536870912 Физическая память
```

Рис. 1.60

Данные, представленные на рис. 1.60, можно перевести в формат HTML-документа и сохранить в файле mem.html:

```
PS C:\> Get-WMIObject Win32_PhysicalMemory¦Select-Object BankLabel, @{Name="Capa
city, Mb"; Expression={$_.Capacity/1Mb}}, Description¦ ConvertTo-HTML¦Out-File c
:\mem.html
```

Командлет Invoke–Item передает данные файла браузеру (рис. 1.61).

```
PS C:\> Invoke-Item c:\mem.html
```

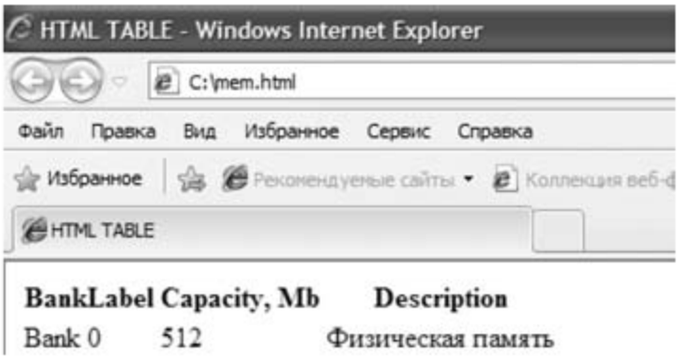


Рис. 1.61

Данные, представленные браузером, не отформатированы. Если в PowerShell создать файл с таблицей стилей styles.css, то можно изменить внешний вид результата, как это показано на рис. 1.62.

```
PS C:\> @'
>> Body {background-color:aqua; }
>> body,table, td,th {font-family:Tahoma; color:Black; Font-size:10pt}
>> th {font-weight:bold; background-color:silver; }
>> td { background-color:white; }
>> '@ > c:\styles.css
```

Рис. 1.62

Для связывания HTML-файла с таблицей стилей в раздел заголовка (Head) требуется вставить тег (рис. 1.63).

<link rel='stylesheet' href='c:\styles.css' type='text/css' />

```
PS C:\> Get-WMIObject Win32_PhysicalMemory|Select-Object BankLabel, @{Name="Capa
city, Mb"; Expression={$_.Capacity/1Mb}}, Description| ConvertTo-HTML -Head "<li
nk rel='stylesheet' href='c:\styles.css' type='text/css' />"|Out-File c:\mem.htm
l
```

Рис. 1.63

Вновь передаем данные браузеру и получаем отформатированные данные (рис. 1.64).

```
PS C:\> Invoke-Item c:\mem.html
```

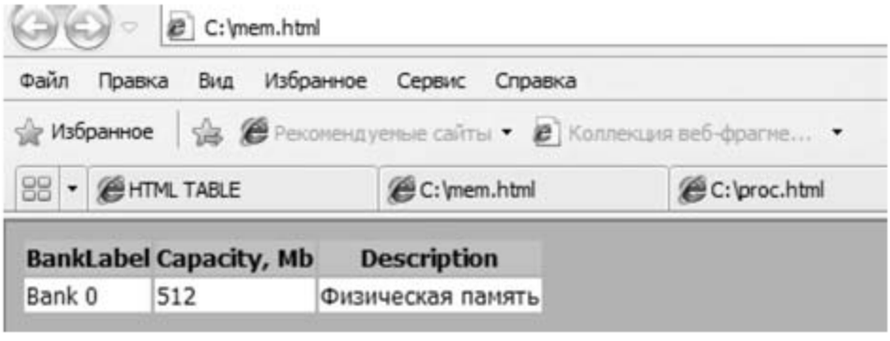


Рис. 1.64

Получение информации о процессорах. Обратимся к экземпляру класса Win32_Processor с целью получения свойств процессора (рис. 1.65).

```
PS C:\> Get-WMIObject Win32_Processor ¦ Format-List *

__GENUS                       : 2
__CLASS                       : Win32_Processor
__SUPERCLASS                  : CIM_Processor
__DYNASTY                     : CIM_ManagedSystemElement
__RELPATH                     : Win32_Processor.DeviceID="CPU0"
__PROPERTY_COUNT              : 46
__DERIVATION                  : {CIM_Processor, CIM_LogicalDevice, CIM_LogicalEle
                                ment, CIM_ManagedSystemElement}
__SERVER                      : N296611
__NAMESPACE                   : root\cimv2
__PATH                        : \\N296611\root\cimv2:Win32_Processor.DeviceID="CP
                                U0"
AddressWidth                  : 32
...
```

Рис. 1.65

Оставим только важные свойства и снова обратимся к таблице стилей (рис. 1.66).

```
PS C:\> Get-WMIObject Win32_Processor ¦ Select-Object DeviceID, Nane, Manufactur
er, Description,CurrentClockSpeed, MaxClockSpeed, L2CacheSize, L2CacheSpeed ¦ Co
nvertTo-HTML -Head "<link rel='stylesheet' href='c:\styles.css' type='text/css'
/>" ¦Out-File c:\proc.html
```

Рис. 1.66

Сформированный файл proc.html передаем браузеру (рис. 1.67).

Подобную информацию можно получить по любому устройству компьютера, сетевому устройству, по любому программному модулю программного обеспечения.

```
PS C:\> Invoke-Item c:\proc.html
```

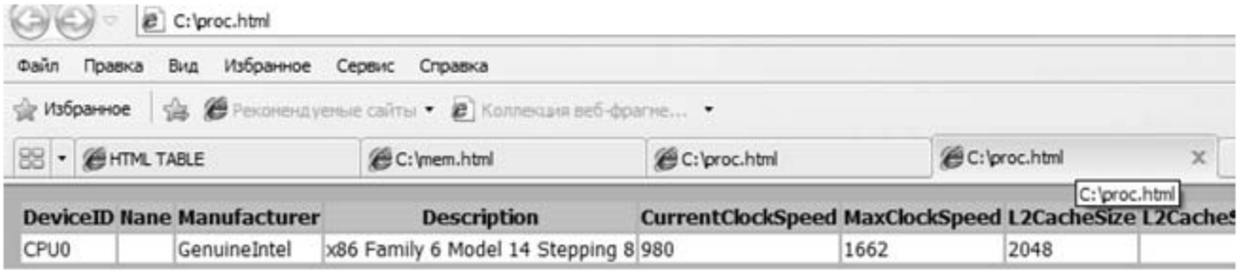


Рис. 1.67

Задания для самостоятельной работы

1. Рассмотреть примеры работы с объектами в подразделе 1.4.4. Уяснить основные варианты работы в интерактивном режиме.

2. Выполнить примеры, иллюстрирующие варианты построения и применения функций в PowerShell. Проверить работу функций с различными наборами данных.
3. Создать файл сценария, приведенный в подразделе 1.6, и запустить его в работу. Найти в Интернете по указанному в подразделе 1.6 адресу библиотеку сценариев. Ознакомиться с содержимым библиотеки, скачать 1—2 простейших сценария и запустить их в работу.
4. Ознакомиться с содержанием примеров выполнения работ по определению характеристик ресурсов компьютерных систем.

ГЛАВА 2

МУЛЬТИПРОГРАММНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ

2.1. Общие сведения о компонентах вычислительного процесса

Современные ОС содержат встроенные средства, с помошью которых можно получить достаточно полную информацию о компонентах вычислительного процесса. Диспетчер задач (Task Manager) ОС Windows позволяет получить обобщенную информацию об организации вычислительного процесса с детализацией до выполняющихся прикладных программ (приложений) и процессов, но не позволяет отслеживать потоки.

Для запуска диспетчера задач и просмотра компонентов вычислительного процесса нужно выполнить определенные действия.

1. Щелкнуть правой кнопкой мыши по панели задач и выбрать строку Диспетчер задач, или нажать клавиши Ctrl + Alt + Del, или нажать последовательно Пуск → Выполнить → taskmgr. На рисунке 2.1 изображено окно Диспетчера задач Windows XP, а на рис. 2.2 — Windows 7 и Vista.

2. Для просмотра приложений перейти на вкладку Приложения. Здесь можно завершить приложение (кнопка Снять задачу), переключиться на другое приложение (кнопка Переключиться) и создать новую задачу (кнопка Новая задача). В последнем случае после нажатия кнопки Новая задача в появившемся окне (рис. 2.3) нужно ввести имя задачи.

3. Просмотр (мониторинг) процессов осуществляется переходом на вкладку Процессы (развернуть окно на весь экран для удобства просмотра). Таблица процессов включает все процессы, запущенные в собственном адресном пространстве, в том числе все приложения и системные сервисы. Обратите внимание на процесс Бездействие системы — фиктивный процесс, занимающий процессор при простое системы.

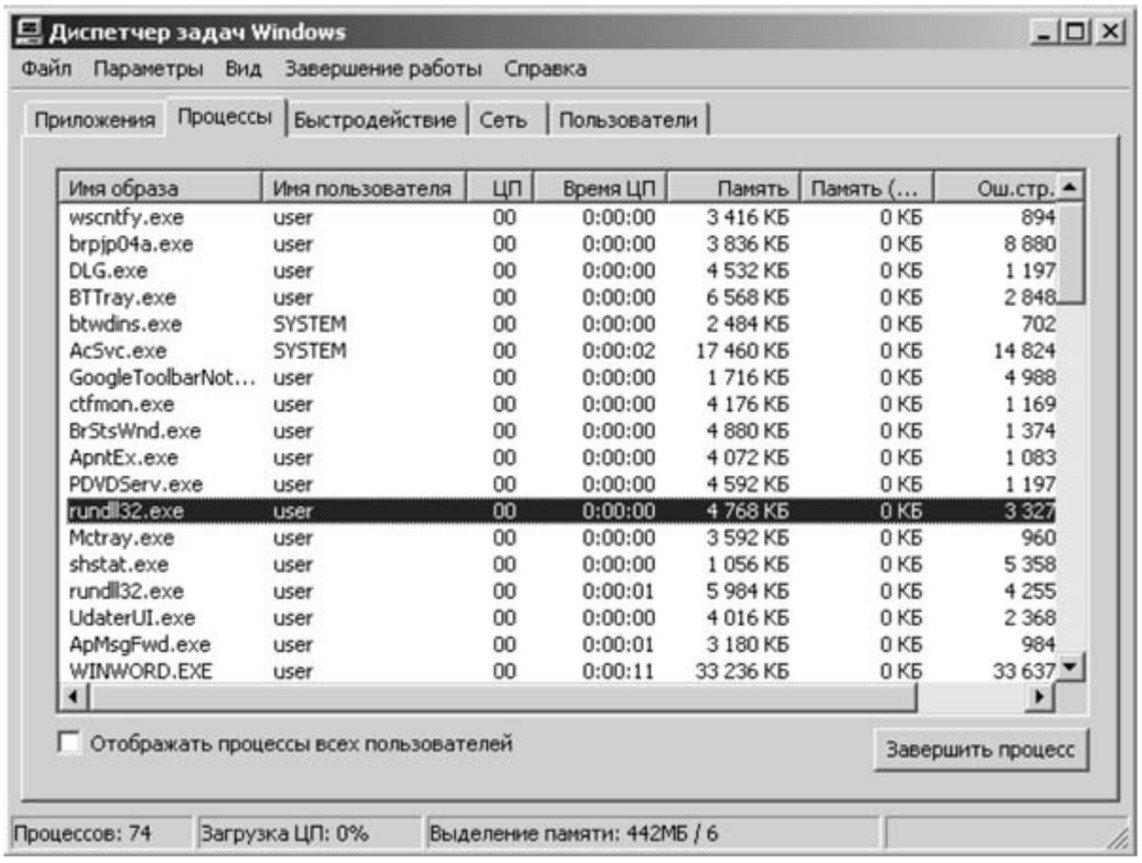

Диспетчер задач Windows
Файл Параметры Вид Завершение работы Справка
Приложения Процессы Быстродействие Сеть Пользователи
Имя образа | Имя пользователя | ЦП | Время ЦП | Память | Память (... | Ош.стр.
wscntfy.exe user 00 0:00:00 3 416 КБ 0 КБ 894
brpjp04a.exe user 00 0:00:00 3 836 КБ 0 КБ 8 880
DLG.exe user 00 0:00:00 4 532 КБ 0 КБ 1 197
BTTray.exe user 00 0:00:00 6 568 КБ 0 КБ 2 848
btwdins.exe SYSTEM 00 0:00:00 2 484 КБ 0 КБ 702
AcSvc.exe SYSTEM 00 0:00:02 17 460 КБ 0 КБ 14 824
GoogleToolbarNot... user 00 0:00:00 1 716 КБ 0 КБ 4 988
ctfmon.exe user 00 0:00:00 4 176 КБ 0 КБ 1 169
BrStsWnd.exe user 00 0:00:00 4 880 КБ 0 КБ 1 374
ApntEx.exe user 00 0:00:00 4 072 КБ 0 КБ 1 083
PDVDServ.exe user 00 0:00:00 4 592 КБ 0 КБ 1 197
rundll32.exe user 00 0:00:00 4 768 КБ 0 КБ 3 327
Mctray.exe user 00 0:00:00 3 592 КБ 0 КБ 960
shstat.exe user 00 0:00:00 1 056 КБ 0 КБ 5 358
rundll32.exe user 00 0:00:01 5 984 КБ 0 КБ 4 255
UdaterUI.exe user 00 0:00:00 4 016 КБ 0 КБ 2 368
ApMsgFwd.exe user 00 0:00:01 3 180 КБ 0 КБ 984
WINWORD.EXE user 00 0:00:11 33 236 КБ 0 КБ 33 637
Отображать процессы всех пользователей
Завершить процесс
Процессов: 74
Загрузка ЦП: 0%
Выделение памяти: 442МБ / 6

Рис. 2.1

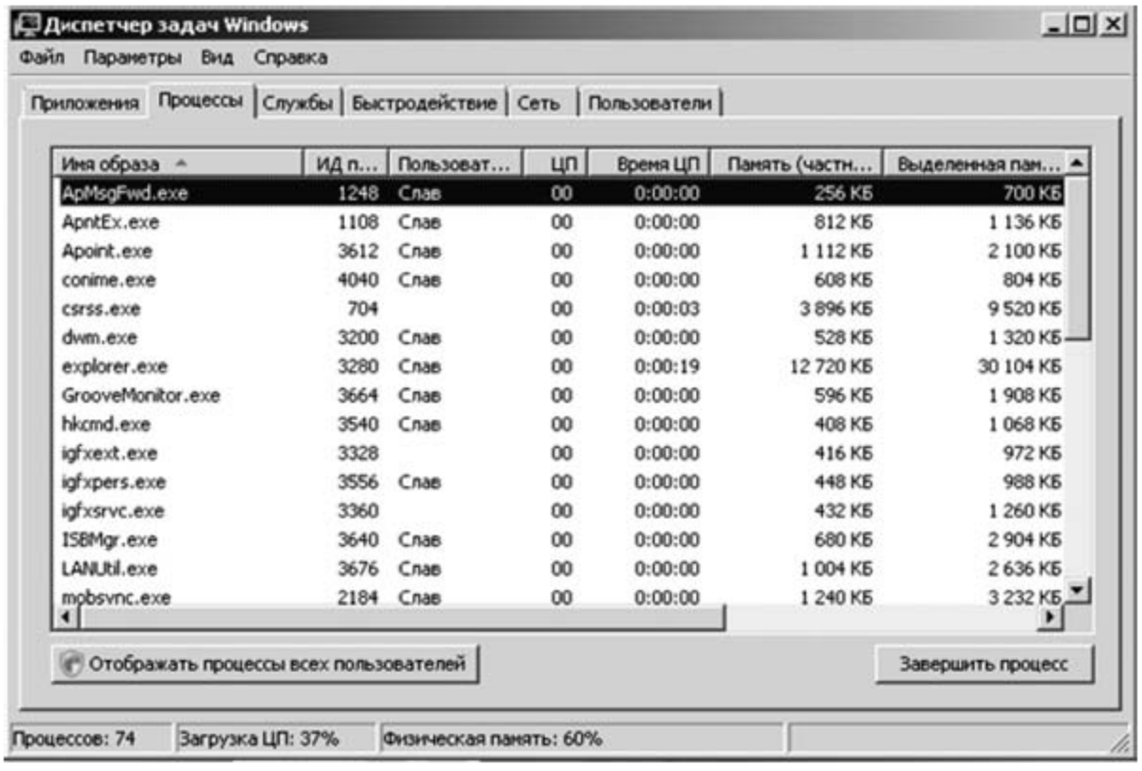

Диспетчер задач Windows
Файл Параметры Вид Справка
Приложения Процессы Службы Быстродействие Сеть Пользователи
Имя образа | ИД п... | Пользоват... | ЦП | Время ЦП | Память (частн... | Выделенная пам...
ApMsgFwd.exe 1248 Слав 00 0:00:00 256 КБ 700 КБ
ApntEx.exe 1108 Слав 00 0:00:00 812 КБ 1 136 КБ
Apoint.exe 3612 Слав 00 0:00:00 1 112 КБ 2 100 КБ
conime.exe 4040 Слав 00 0:00:00 608 КБ 804 КБ
csrss.exe 704 00 0:00:03 3 896 КБ 9 520 КБ
dwm.exe 3200 Слав 00 0:00:00 528 КБ 1 320 КБ
explorer.exe 3280 Слав 00 0:00:19 12 720 КБ 30 104 КБ
GrooveMonitor.exe 3664 Слав 00 0:00:00 596 КБ 1 908 КБ
hkcmd.exe 3540 Слав 00 0:00:00 408 КБ 1 068 КБ
igfxext.exe 3328 00 0:00:00 416 КБ 972 КБ
igfxpers.exe 3556 Слав 00 0:00:00 448 КБ 988 КБ
igfxsrvc.exe 3360 00 0:00:00 432 КБ 1 260 КБ
ISBMgr.exe 3640 Слав 00 0:00:00 680 КБ 2 904 КБ
LANUtil.exe 3676 Слав 00 0:00:00 1 004 КБ 2 636 КБ
mobsync.exe 2184 Слав 00 0:00:00 1 240 КБ 3 232 КБ
Отображать процессы всех пользователей
Завершить процесс
Процессов: 74
Загрузка ЦП: 37%
Физическая память: 60%

Рис. 2.2

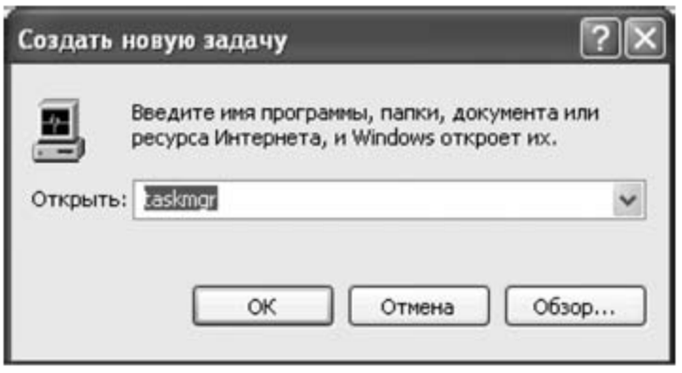


Рис. 2.3

4. Если требуется просмотреть 16-разрядные процессы, то в меню Параметры следует выбрать команду Отображать 16-разрядные задачи (рис. 2.4).

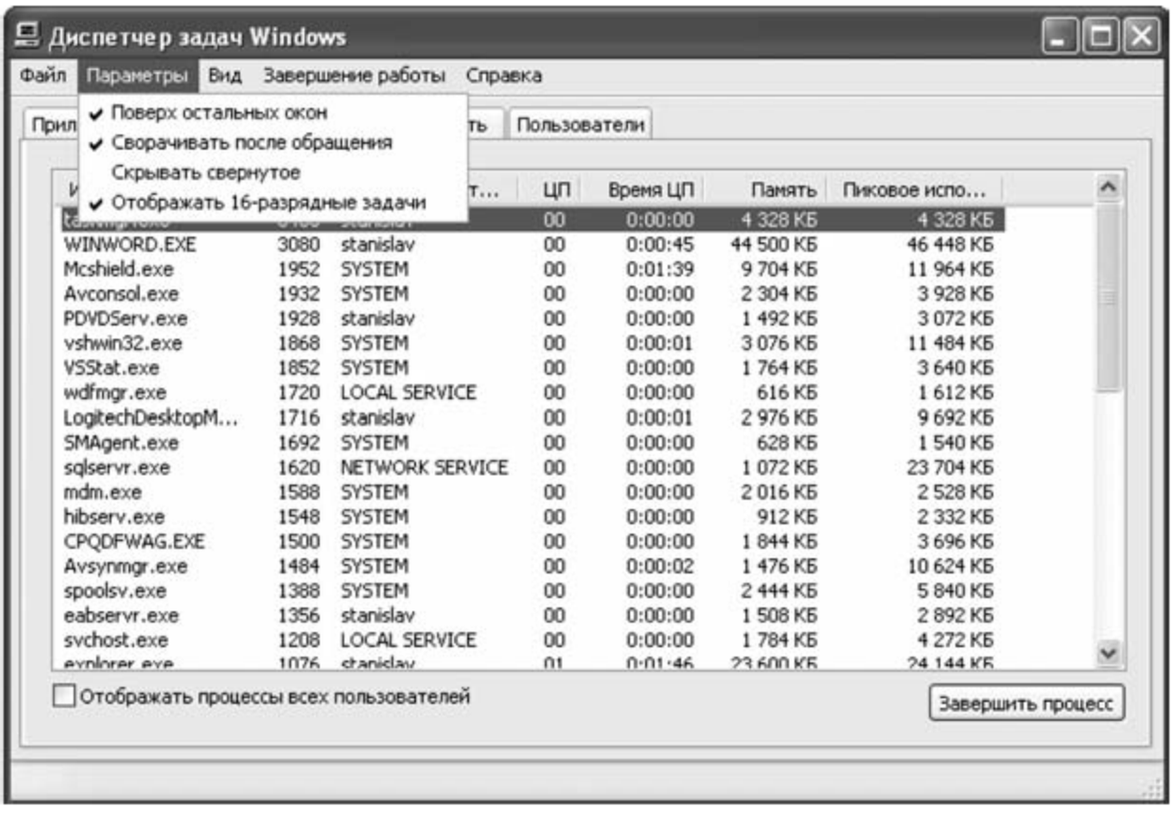


Рис. 2.4

5. Для выбора просматриваемых показателей (характеристик) с помощью команды Выбрать столбцы (меню Вид) установить флажки рядом с показателями, которые требуется отображать.

В качестве примера рассмотрим процессы приложения Word. Для этого нужно выполнить следующие действия:

1) запустить MS Word. Щелкнуть правой клавишей мыши по названию приложения и в появившемся контекстном меню выбрать строку Перейти к процессам. Произойдет переход на вкладку

Процессы. Можно просмотреть число потоков и другие характеристики процесса;

2) изменить приоритет процесса. На вкладке Процессы щелкнуть правой клавишей мыши по названию процесса и выбрать в контекстном меню строку Приоритет (рис. 2.5). Изменив приоритет, в колонке Базовый приоритет можно увидеть его новое значение (обратить внимание на предупреждение);

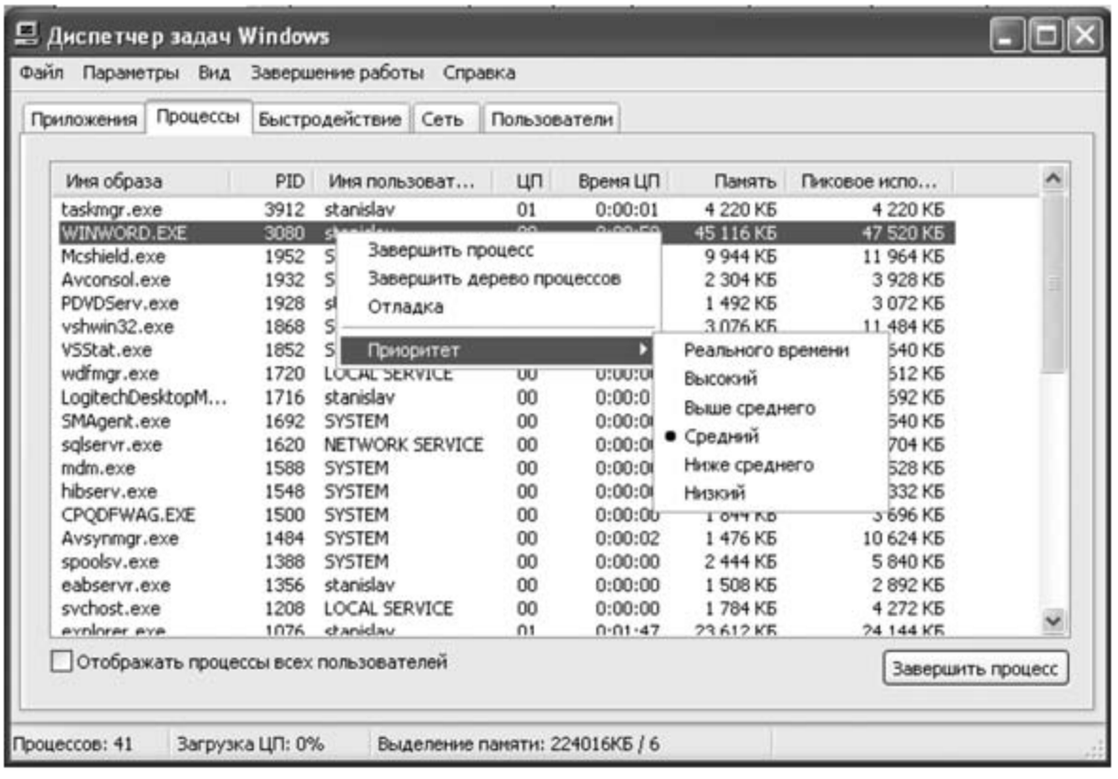


Рис. 2.5

3) изменить скорости обновления данных. Войти в меню Вид и выбрать команду Скорость обновления (рис. 2.6). Установить требуемую скорость (высокая — каждые 0,5 с, обычная — каждую секунду, низкая — каждые 4 с, приостановить — обновления нет). Следует иметь ввиду, что с повышением скорости мониторинга возрастают затраты ресурсов компьютера на работу ОС, что в свою очередь вносит погрешность в результаты мониторинга.

Диспетчер задач позволяет получить обобщенную информацию об использовании основных ресурсов компьютера. Для этого необходимо:

1) перейти на вкладку Быстродействие (рис. 2.7). Верхние два окна показывают интегральную загрузку процессора и хронологию загрузки, нижние два окна — те же показатели, но по использованию памяти;

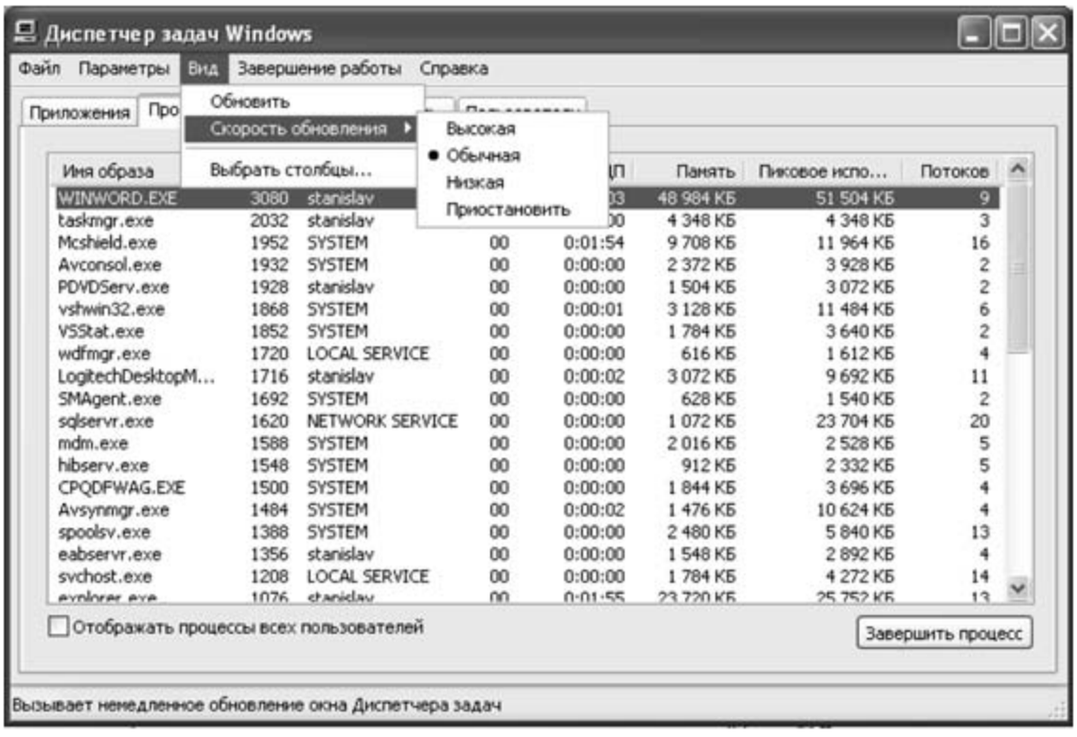


Рис. 2.6

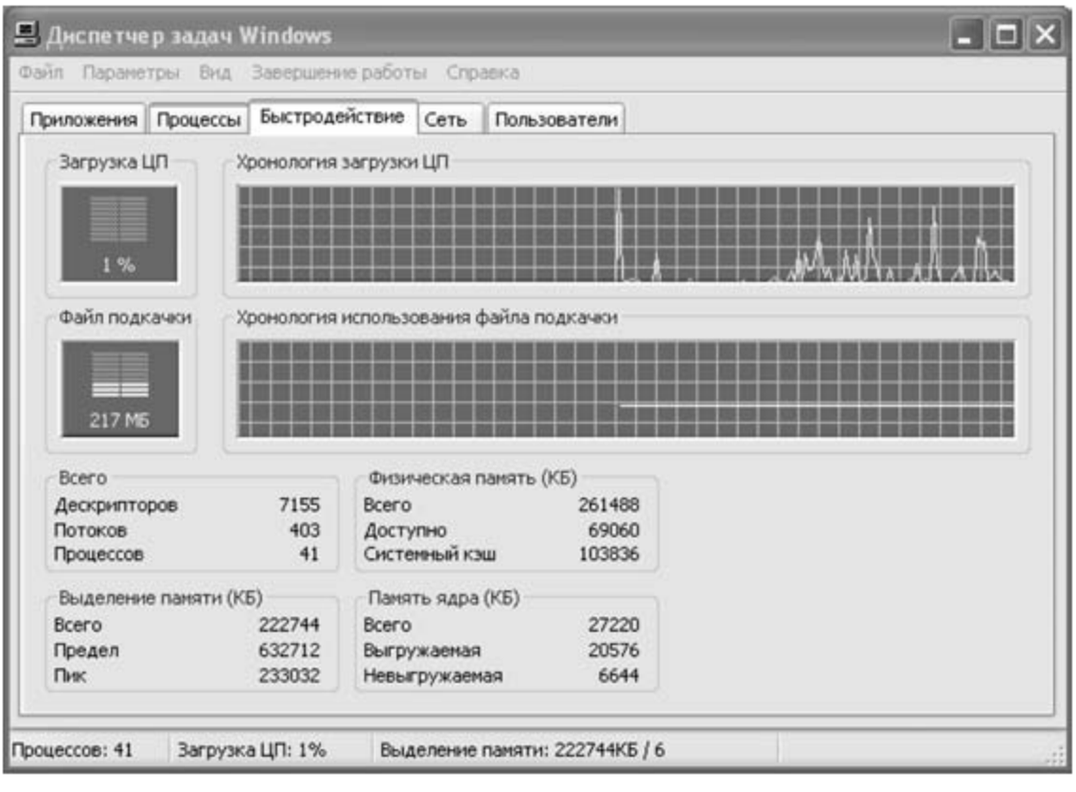


Рис. 2.7

2) для просмотра использования процессора в режиме ядра (красный цвет) войти в меню Вид и щелкнуть на строке Вывод времени ядра.

В нижней части окна вкладки Быстродействие отображается информация о количестве процессов и потоков, участвующих в мультипрограммном вычислительном процессе, общем количестве дескрипторов (описателей) объектов, созданных ОС, а также информация о доступной и выделенной памяти для реализации приложений. Кроме того, приводятся сведения о выделении памяти под ядро ОС с указанием выгружаемой и невыгружаемой памяти ядра и объеме системного кэша.

Задания для самостоятельной работы

1. Исследовать мультипрограммный вычислительный процесс на примере выполнения самостоятельно разработанных трех задач (например, заданий по курсу программирования).
2. Для одной из задач определить PID, загрузку и время работы центрального процессора, базовый приоритет процесса, использование памяти, хронологию использования ЦП в режиме ядра. Изменить приоритет процесса и установить, влияет ли это на время выполнения приложения.
3. Монопольно выполнить каждую из трех задач, определить время их выполнения. Запустить одновременно (друг за другом) три задачи, определить время выполнения пакета.
4. Ответить на следующие вопросы:
   1) в каком случае суммарное время выполнения задач больше: при последовательном или одновременном выполнении?
   2) как изменилось время выполнения каждой отдельной задачи?
   3) как изменится время выполнения отдельной задачи при изменении ее приоритета? Окажет ли влияние изменение приоритета одной задачи на время выполнения другой задачи? Объяснить результаты.

2.2. Просмотр и анализ информации о заданиях, процессах и потоках

Ряд программ как производителей ОС, так и сторонних фирм позволяют получить более детальную информацию о компонентах вычислительного процесса. В работе [6] рассмотрена программа Process

Viewer (Просмотр процесса), позволяющая увидеть процессы и потоки простым одноступенчатым способом как на локальном, так и на удаленном компьютере сети (рис. 2.8)[1].

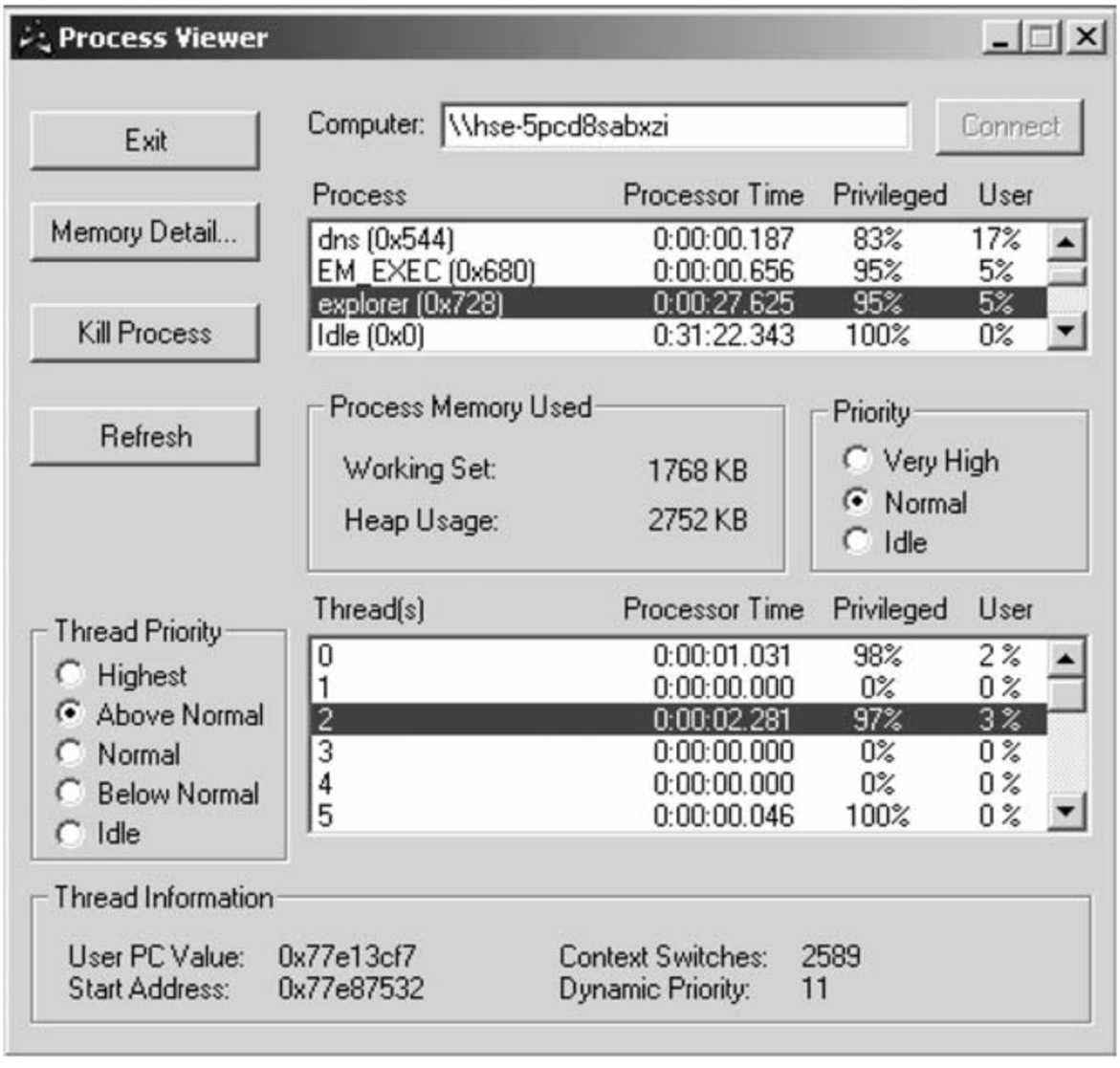


Рис. 2.8

В верхней части окна (см. рис. 2.8) перечисляются все процессы, выполняемые на выбранном компьютере. Статистические данные включают количество общего процессорного времени и количество времени, проведенного в режиме пользователя и в режиме ядра. Под списком процессов приводятся статистические данные для выделенного процесса, включающие количество используемой памяти и приоритет процесса.

В нижней части окна перечисляются потоки выделенного процесса. Здесь также выводятся статистические данные по использованию про-

[1] Большинство используемых далее программ являются составной частью пакетов MS SDK, Windows Resource Kit и Windows Support Tools.

цессора, но уже потоками. Слева от перечня потоков даются текущие значения их приоритетов. Поток получает то же значение приоритета, что и связанный с ним процесс, однако это базовый приоритет, который во время выполнения потока может меняться ОС. Динамический приоритет выделенного потока и другая информация о нем помещается в нижней части окна.

Щелчком по кнопке Memory Details можно вывести окно с информацией об использовании памяти процессом (рис. 2.9). Поле адресного пространства (User Address Space) содержит наименование просматриваемого адресного пространства. Значение Total Commit указывает на память, используемую всем процессом. Если щелкнуть по стрелке комбинированного окна, то можно увидеть перечень файлов .dll и .exe, используемых данным приложением (рис. 2.10). Выбрав один из этих элементов, можно увидеть, какая память используется данной частью приложения.

Запустив программу, можно просмотреть с ее помощью текущие характеристики процессов и потоков, в том числе свои задачи.

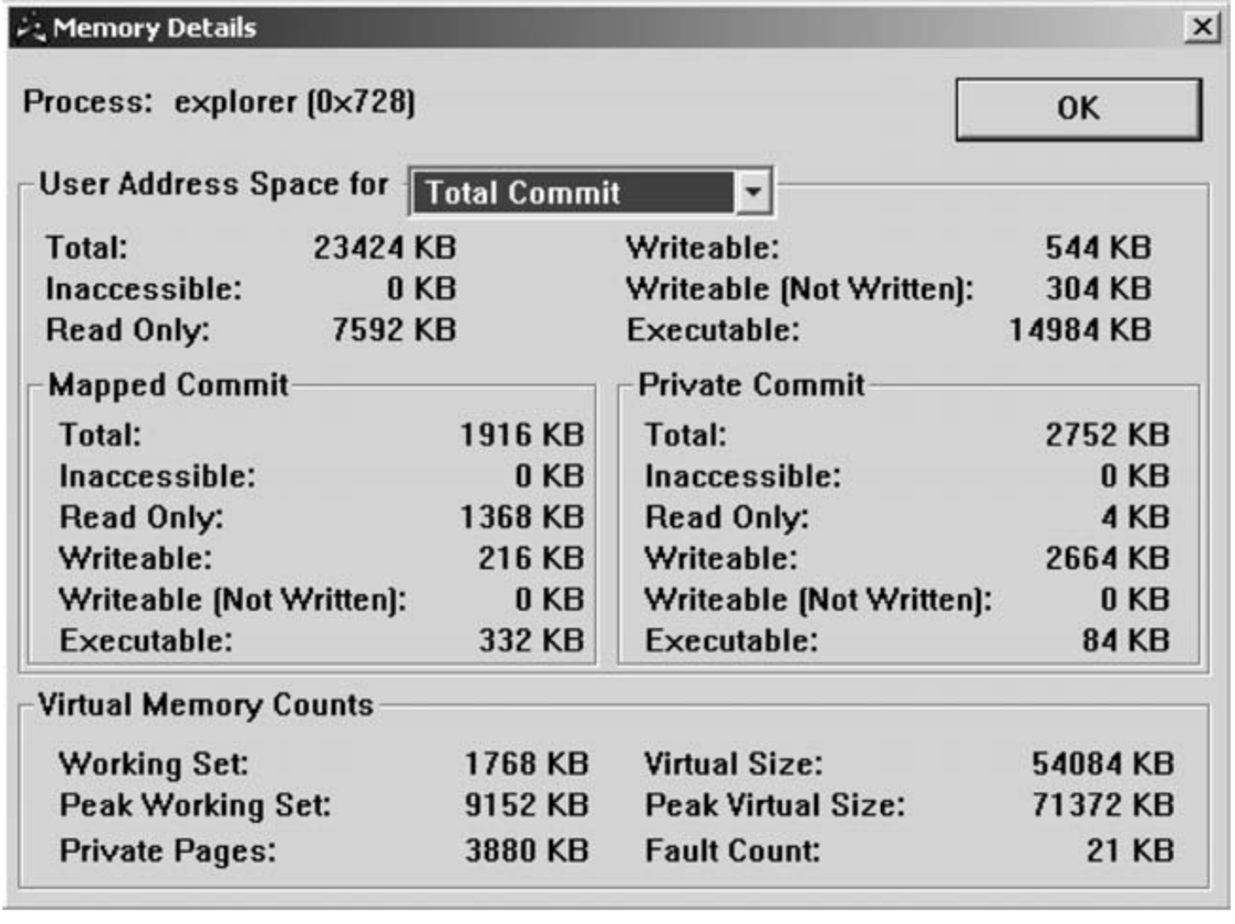


Рис. 2.9

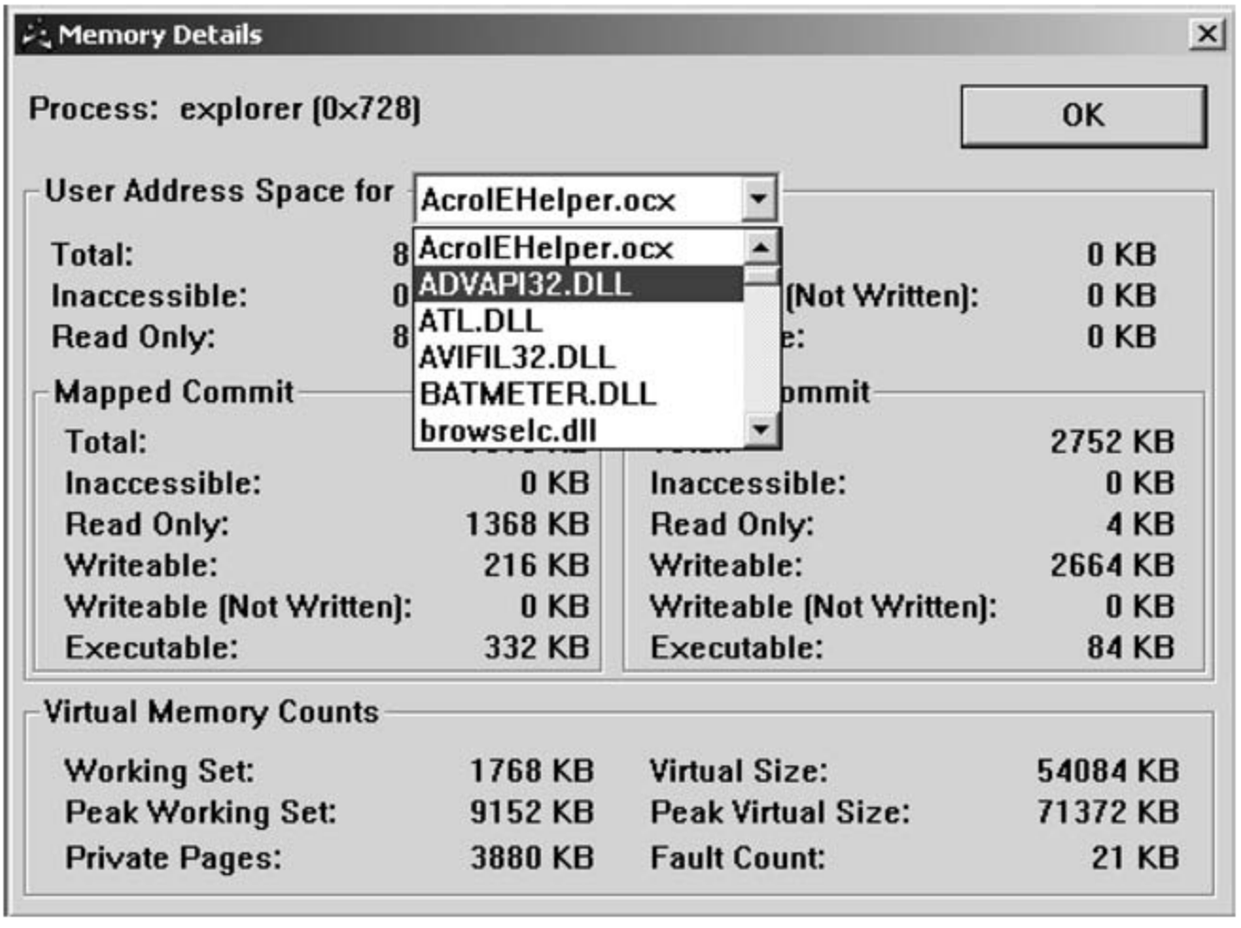


Рис. 2.10

Производитель ОС Windows компания Microsoft предлагает утилиту Microsoft Spy++. Программа обладает большими возможностями и поставляется комплектно с ОС. Ее можно использовать для просмотра процессов, потоков и ресурсов, используемых ими, в том числе потоков пользовательского интерфейса (UI-потоков).

Проиллюстрируем возможности утилиты для просмотра потоков пользовательского интерфейса. Для этого запустим программу в режиме Process (меню Spy -> Processes). Найдем процесс Explorer и щелкнем по значку + слева от него. На экране высветятся потоки процесса. Около потоков UI расположен значок + (рис. 2.11), у рабочих потоков он отсутствует. Щелкнув по значку + потока UI, можно увидеть объекты, поддерживаемые этим потоком. Просмотреть свойства любого из этих объектов можно, щелкнув по нему правой клавишей мыши и выбрав из контекстного меню строку Свойства (рис. 2.12). На вкладке Process можно прочитать идентификаторы процесса и потока, чтобы наблюдать за его производительностью.

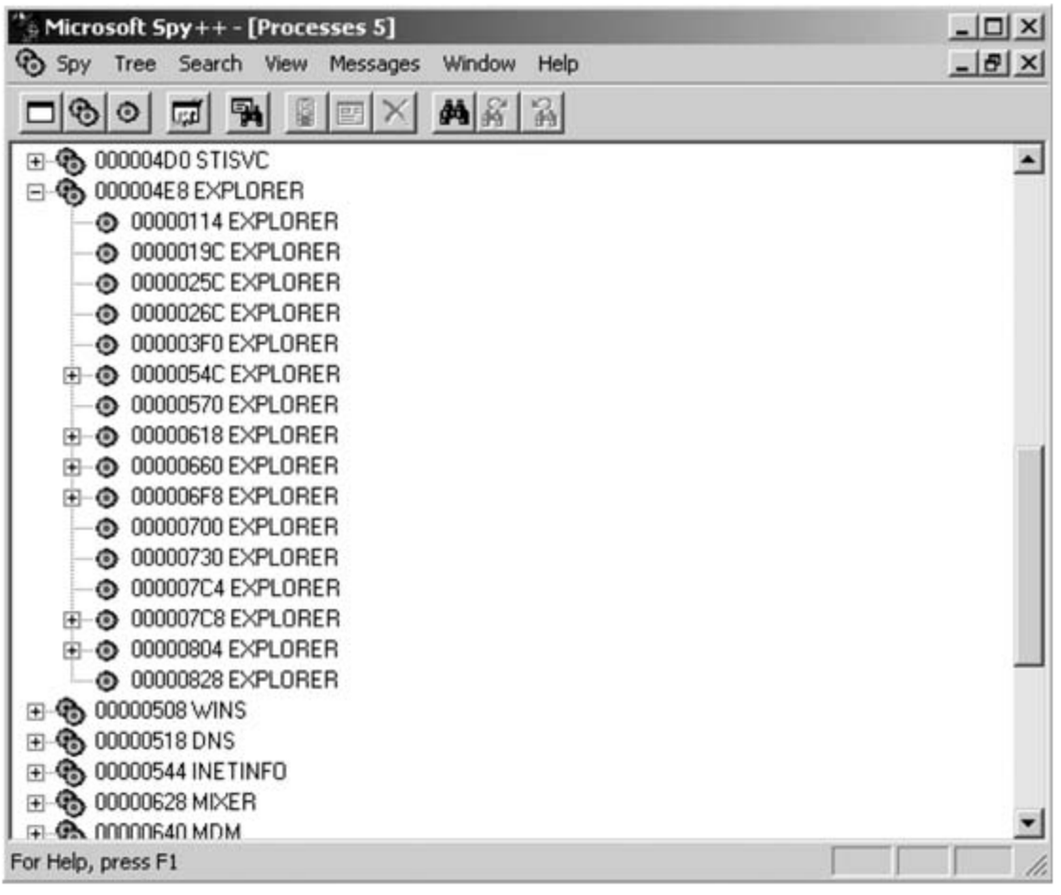


Рис. 2.11

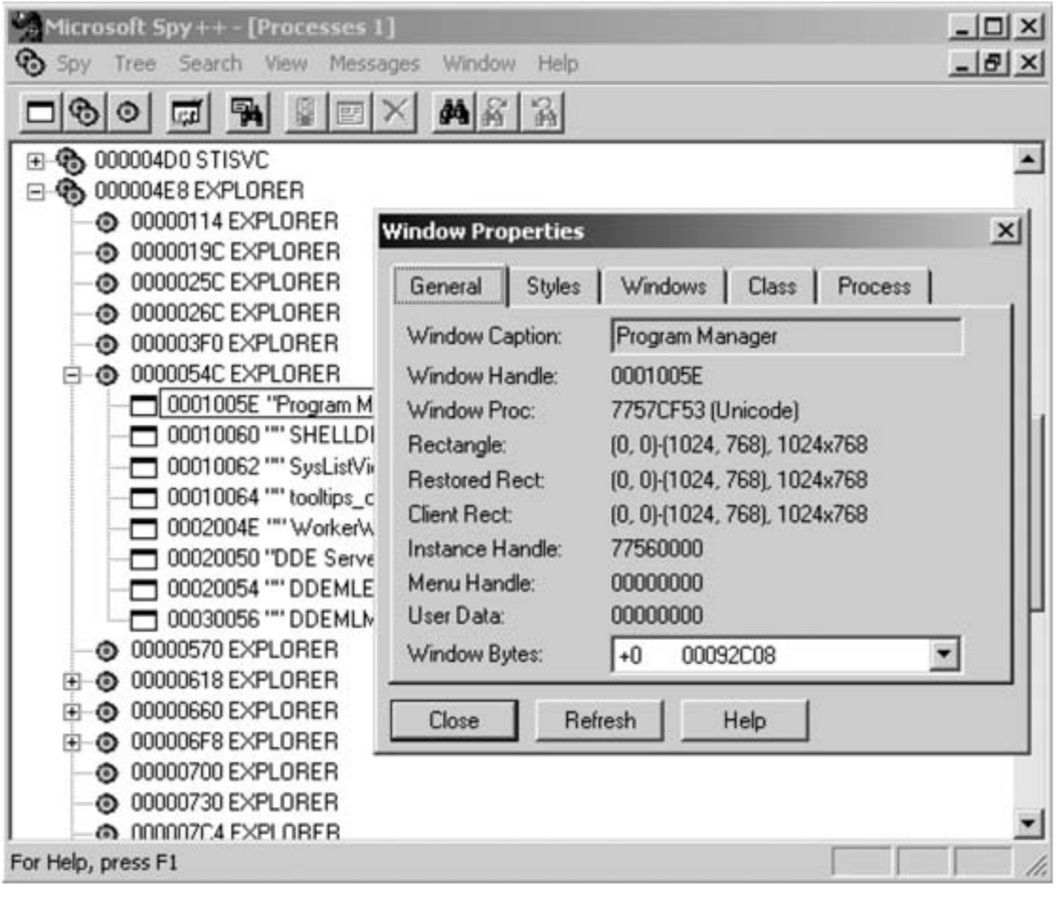


Рис. 2.12

Программа CPU Stress, рассмотренная в [6], нагружает процессор, чтобы проконтролировать его работу в заданных условиях. На рисунке 2.13 показано начальное окно программы. Как видно, программа рассчитана на запуск от одного до четырех потоков. Кроме задания количества потоков, утилита позволяет менять приоритет потоков процесса и уровень их активности. Программа начинает работать сразу после запуска. Ее можно использовать для определения возможностей компьютера и их деградации во времени.

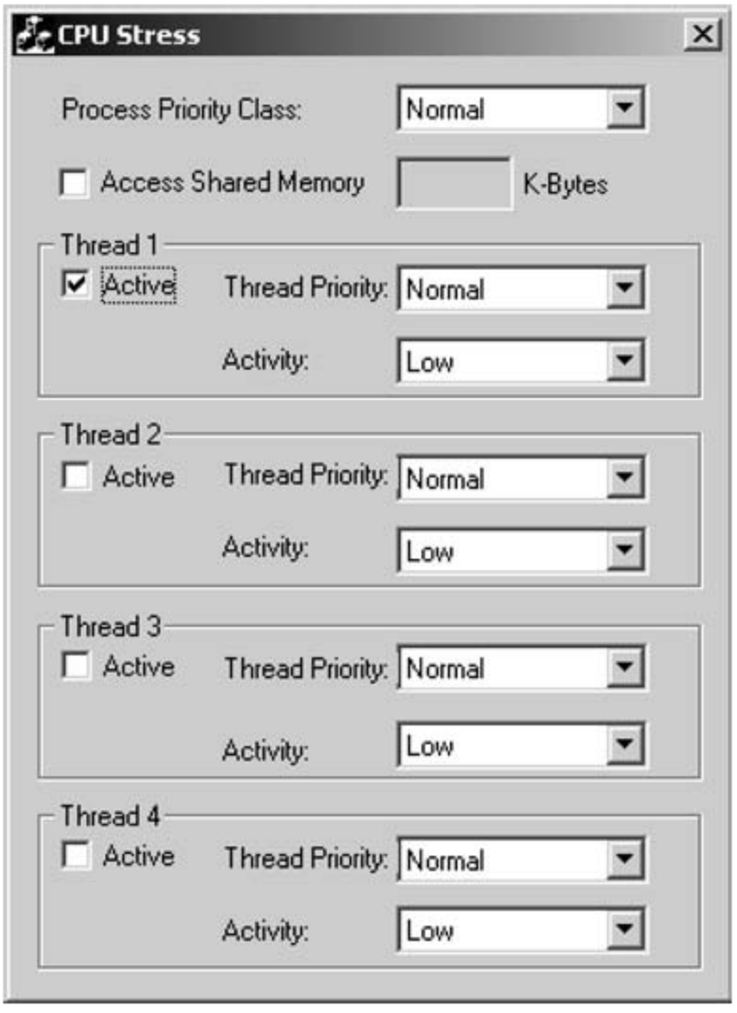


Рис. 2.13

Интересные возможности по изучению вычислительного процесса предоставляет программа Scheduling Lab (http://files.zipsites.ru/books/computers/windows/windows2000/ richter4ru.zip) [11]. Программа позволяет проводить эксперименты с классами приоритетов процессов и относительными приоритетами потоков и исследовать их влияние на общую производительность системы. После запуска программы открывается окно, показанное на рис. 2.14. Изначально первичный поток работает очень активно, и степень использования процессора

доходит до 100%. Все действия потока сводятся к увеличению на единицу исходного значения и выводу текущего результата в правое окно списка.

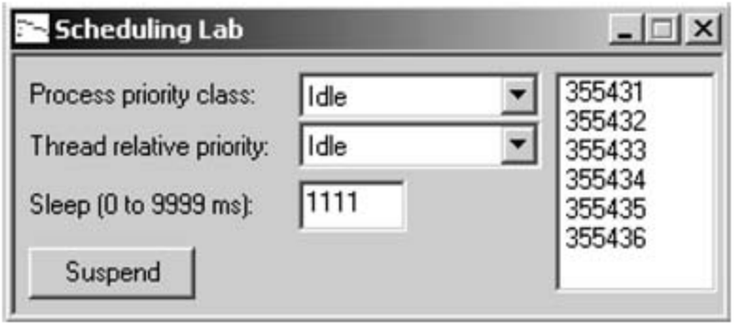


Рис. 2.14

Для анализа влияния изменения приоритета потока следует запустить несколько экземпляров (3—4) программы и понаблюдать за нагрузкой процессора, используя оснастку Производительность (см. ниже). В начале теста процессор будет загружен на 100%, и все экземпляры программы получают примерно равные кванты процессорного времени. Если поднять класс приоритета одному из экземпляров программы, бóльшую долю процессорного времени начнет получать именно этот экземпляр, а аналогичные показатели для других экземпляров упадут. Однако они никогда не опустятся до нуля, поскольку действует механизм динамического повышения приоритета (убедиться в этом можно, построив диаграммы изменения приоритетов процессов).

Наиболее крупной единицей работы для ОС является задание, приведенное в работе [8]. Программа JobLab [11] позволяет проводить эксперименты по созданию, выполнению, расширению и завершению заданий, состоящих из нескольких процессов (их число определяет пользователь) и потоков. После запуска программы открывается окно, показанное на рис. 2.15.

Запущенная программа создает объект-задание с именем JobLab, который можно наблюдать в оснастке Производительность и Диспетчере задач. Программа также создает порт завершения ввода-вывода и связывает с ним объект-задание. Это позволяет отслеживать уведомления от задания и отображать их в списке в нижней части окна.

Изначально в задании нет процессов, и никаких ограничений для него не установлено. Поля в верхней части окна позволяют задавать базовые и расширенные ограничения путем ввода соответствующих значений. Пустое поле означает отсутствие ограничения. Кроме базовых ограничений, можно задавать ограничения по пользовательскому интерфейсу. Пометив флажок Preserve Job Time When Applying Limits, можно

изменять ограничения, не сбрасывая значений элементов Period Time: User = xx, Kernel = xx. Этот флажок становится недоступным при наложении ограничений на процессорное время для отдельных заданий.

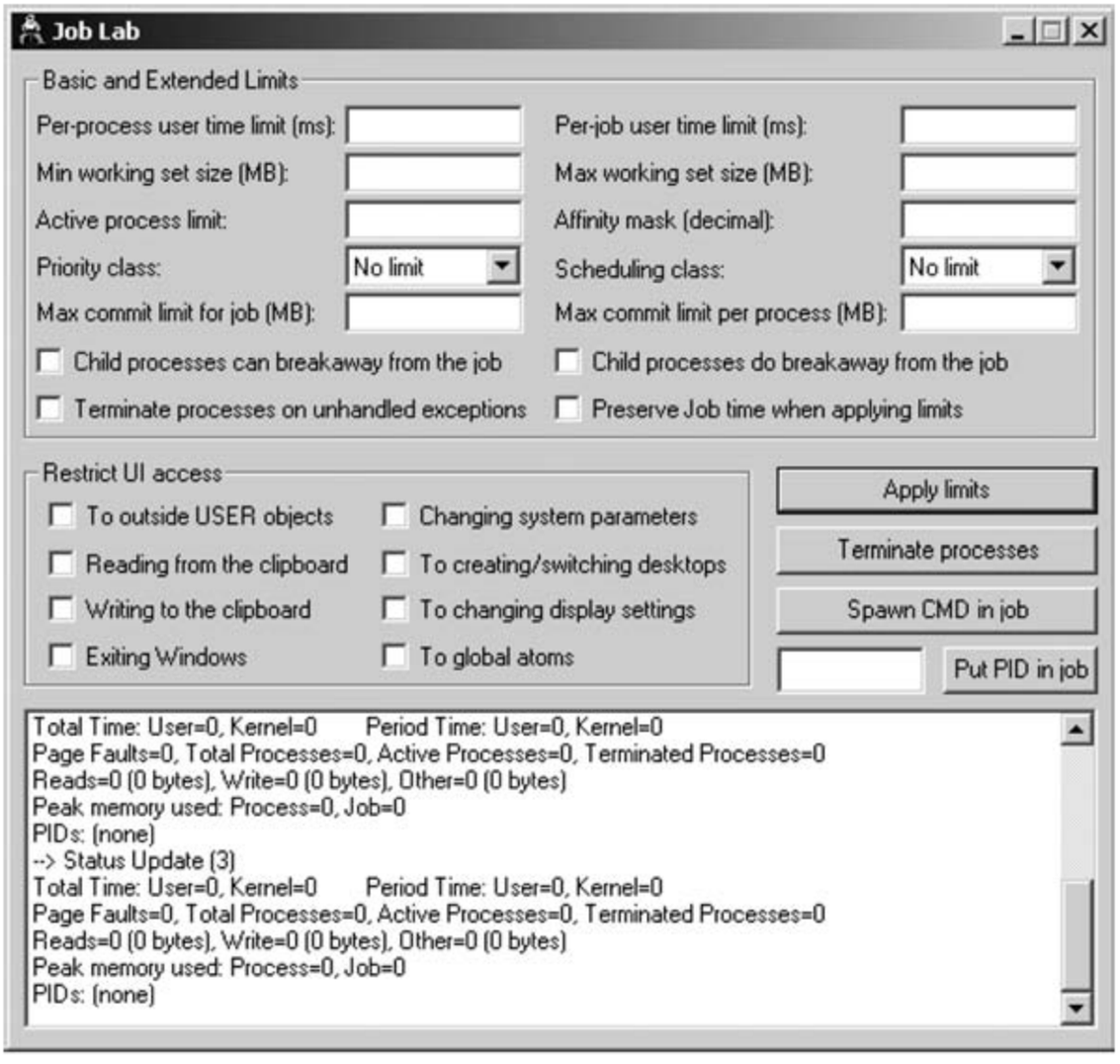


Рис. 2.15

Кнопка Terminate Processes уничтожает все процессы в задании. Кнопка Spawn CMD in job запускает командный процессор, сопоставляемый с заданием. Из этого процесса можно запускать дочерние процессы (в том числе предложенные в этом разделе: CPU Stress и Scheduling Lab) и наблюдать, как они ведут себя, став частью задания. Кнопка Put PID in job позволяет связать существующий свободный процесс (например, свою задачу) с заданием, т.е. включить его в задание.

Текст в нижней части окна каждые 10 с обновляет информацию о статусе задания: базовые и расширенные сведения, статистику ввода-вывода, а также пиковые объемы памяти, занимаемые процессом и заданием.

Задания для самостоятельной работы

1. Исследовать процесс выполнения задания пользователя, сформированного с помощью программы JobLab, добавляя в задание процессы, запуская кнопкой Spawn CMD in job программы CPU Stress и Scheduling Lab. Предварительно создать журнал счетчиков (см. подраздел 2.4), выбрав счетчики, характеризующие использование процессора и основные характеристики процессов.
2. Провести запись журналов в формате CVS, а затем в двоичном формате. Построить диаграммы и графики, используя оснастку Производительность и Excel.
3. Создать журнал Оповещения, выбрав счетчики по своему усмотрению, обосновав решение.

2.3. Детальное исследование вычислительного процесса

В операционных системах Windows имеются средства, позволяющие детально анализировать вычислительные процессы. К таким средствам относится Системный монитор и Оповещения и журналы производительности. Для доступа к этим средствам нужно выполнить последовательность действий:

Пуск -> Программы -> Администрирование -> Производительность

Откроется окно Производительность, содержащее две оснастки: Системный монитор и Оповещения и журналы производительности (рис. 2.16). Оснастки являются инструментом системного администратора. Доступ к ним можно получить, запустив Microsoft Management Consol (mmc). Для этого нужно выполнить следующие действия:

Пуск -> Выполнить -> Открыть mmc -> OK.

Откроется окно Консоль 1, в котором можно выбрать нужную оснастку.

Системный монитор позволяет анализировать вычислительный процесс, используя различные счетчики. Объектами исследования

являются практически все компоненты компьютера: процессор, кэш, задание, процесс, поток, физический диск, файл подкачки, очереди сервера, протоколы и др.

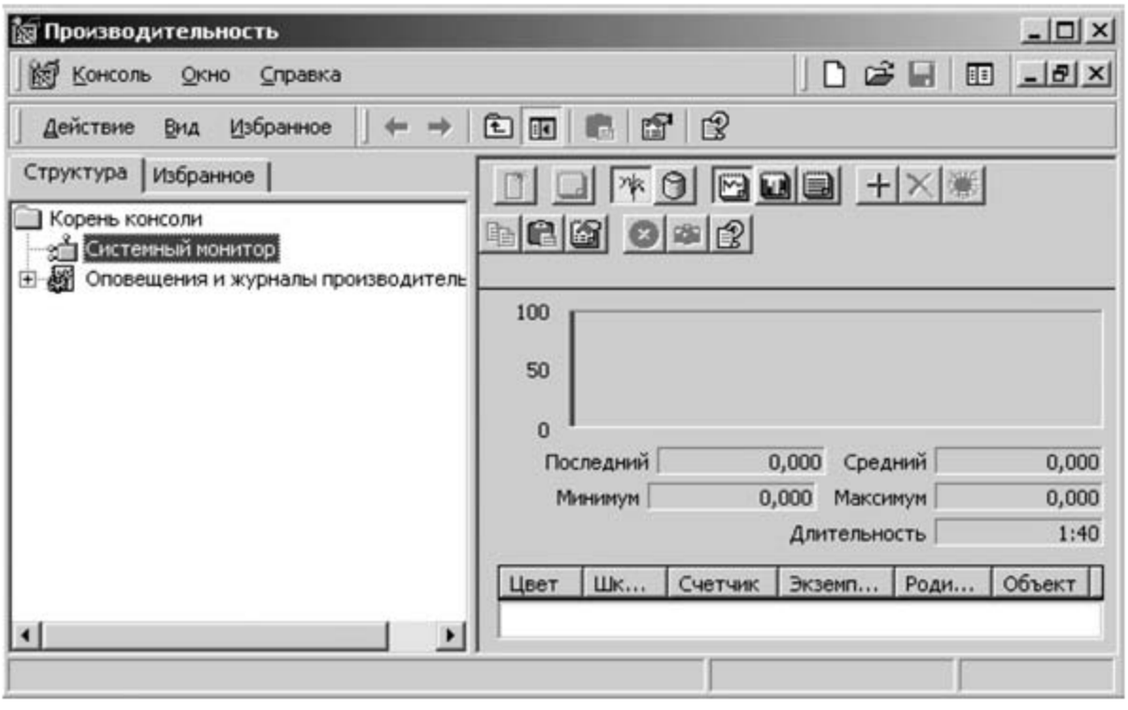


Рис. 2.16

Для просмотра и выбора объектов мониторинга и настройки счетчиков нужно выполнить следующие действия:

1) открыть оснастку Производительность. На панели результатов (правая панель) щелкнуть правой клавишей мыши и выбрать в контекстном меню строку Добавить счетчики или щелкнуть на кнопке Добавить (значок +) на панели инструментов;
2) в появившемся окне Добавить счетчики выбрать объект мониторинга, например процессор, а затем — нужные счетчики из списка Выбрать счетчики из списка, например % времени прерываний, нажимая кнопку Добавить. Для потока можно определить:
   — число контекстных переключений в 1 с;
   — состояние потока (для построения графа состояний и переходов);
   — текущий приоритет (для анализа его изменения);
   — базовый приоритет;
   — % работы в привилегированном режиме и др.

Нажав кнопку Объяснение, можно получить информацию о счетчике. При выборе нескольких однотипных объектов, например потоков, нужно их указать в правом поле Выбрать вхождения из списка (рис. 2.17).

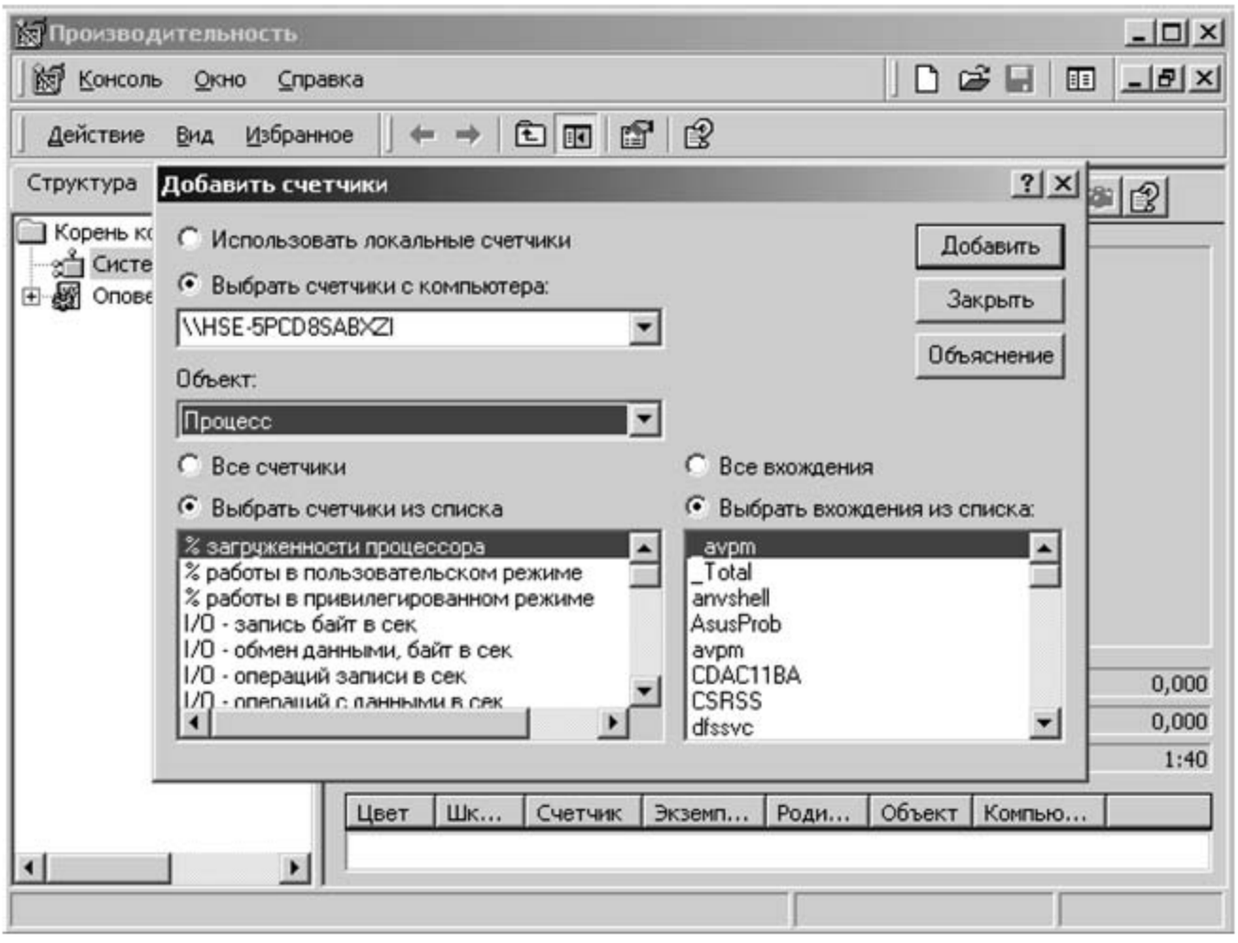


Рис. 2.17

Для удобства работы предусмотрена настройка вида отображаемой информации. Просмотр информации производительности возможен в виде графика, гистограммы и отчета. Для настройки внешнего вида окна нужно щелкнуть на графике правой кнопкой мыши и выбрать команду Свойства (рис. 2.18).

На вкладке Общие можно задать вид информации (график, гистограмма, отчет), отображаемые элементы (легенда, строка значений, панель инструментов), данные отчета и гистограммы (максимальные, минимальные и т. д), период обновления данных и др.

На вкладке Источник задается источник данных. На вкладке Данные можно для каждого счетчика задать цвет, ширину линии, масштаб и др.

На вкладке График можно задать заголовок, вертикальную и горизонтальную сетку, диапазон значений вертикальной шкалы. На вкладках Цвета и шрифты можно изменить набор цветов и шрифт.

Режимы График и Гистограмма не всегда удобны для отображения результатов анализа, например при большом количестве счетчиков, меняющих свое значение в разных диапазонах величин. Режим Отчет

позволяет наблюдать реальные значения счетчиков, так как не использует масштабирующих множителей. В этом режиме доступна только одна опция — изменение интервала опроса.

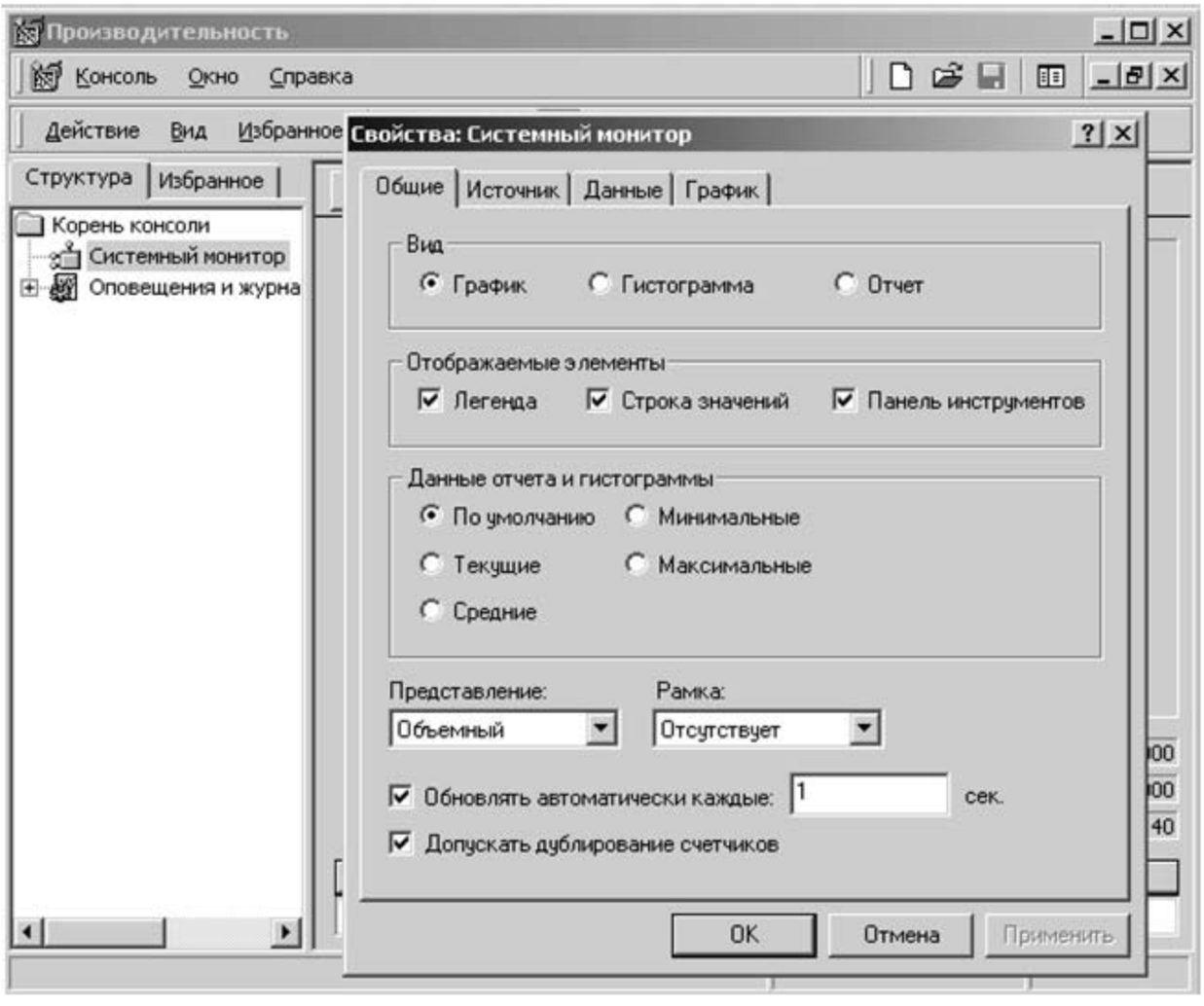


Рис. 2.18

Полученная с помощью Монитора производительности информация позволяет наглядно произвести экспресс-анализ функционирования нужного компонента вычислительного процесса или устройства компьютера. Например, на рис. 2.19 показана следующая информация об использовании процессора: процент работы процессора в пользовательском и привилегированном режимах и количество прерываний в 1 с.

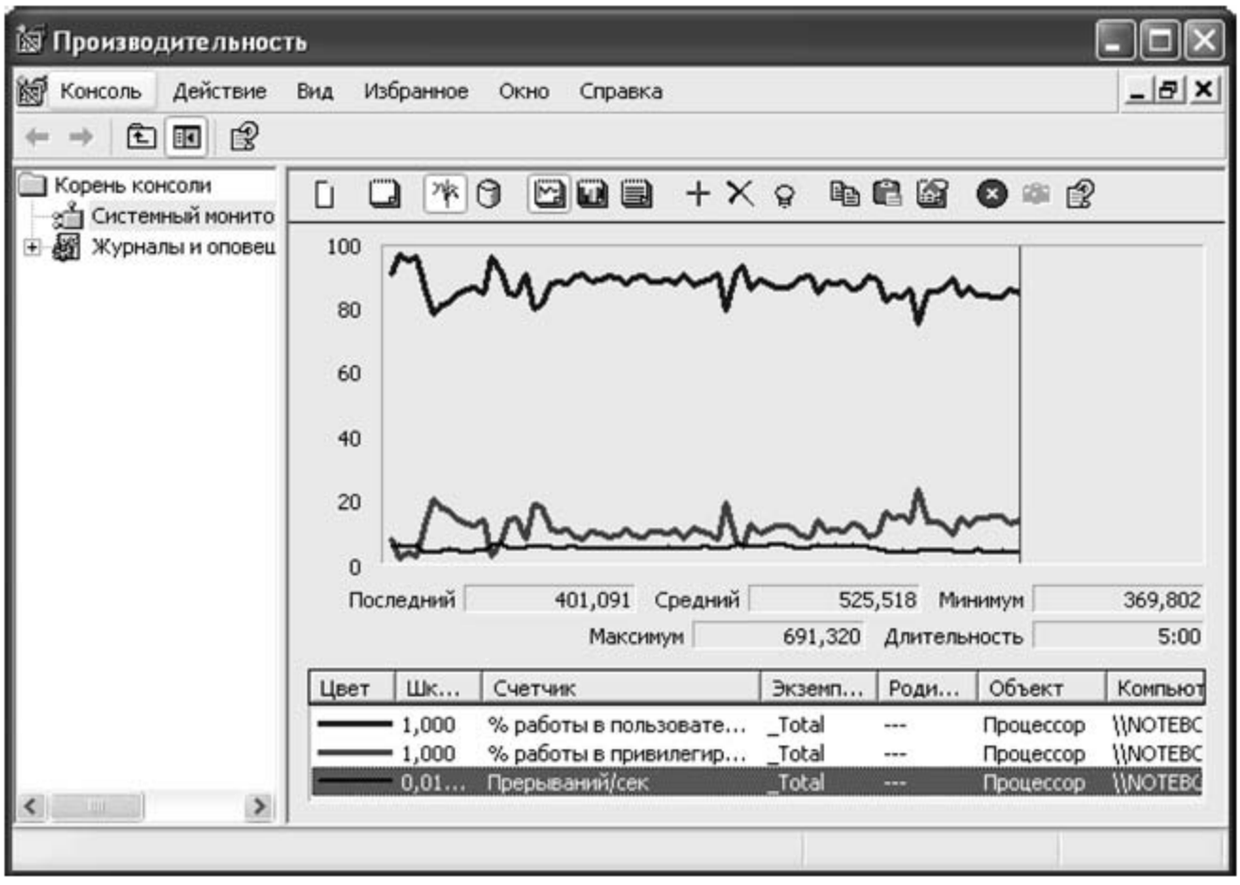


Рис. 2.19

Задания для самостоятельной работы

1. Запустить предварительно разработанную программу. Используя возможности оснастки Производительность, получить диаграммы, характеризующие использование процессора при его нагрузке различным количеством потоков, меняя их активность и уровни приоритета.
2. Исследовать свои задачи (приложения). Определить характеристики процессов: % загрузки процессора (в пользовательском и привилегированном режимах), % времени прерываний, количество прерываний, базовый приоритет, обращения к диску, время выполнения процесса.

2.4. Запись и представление результатов анализа вычислительного процесса

Оснастка Оповещения и журналы производительности содержит три компонента: Журналы счетчиков, Журналы трассировки и Опове-

щения, которые можно использовать для записи и просмотра результатов исследования вычислительного процесса. Данные, созданные при помощи оснастки, можно просматривать как в процессе сбора, так и после его окончания.

Файл журнала счетчиков состоит из данных для каждого указанного счетчика на указанном временном интервале. Для создания журнала необходимо выполнить следующие действия:

1) запустить оснастку Производительность;
2) дважды щелкнуть на значке Оповещения и журналы производительности;
3) выбрать значок Журналы счетчиков, щелкнуть правой кнопкой мыши в панели результатов и выбрать в контекстном меню пункт Новые параметры журнала (рис. 2.20);

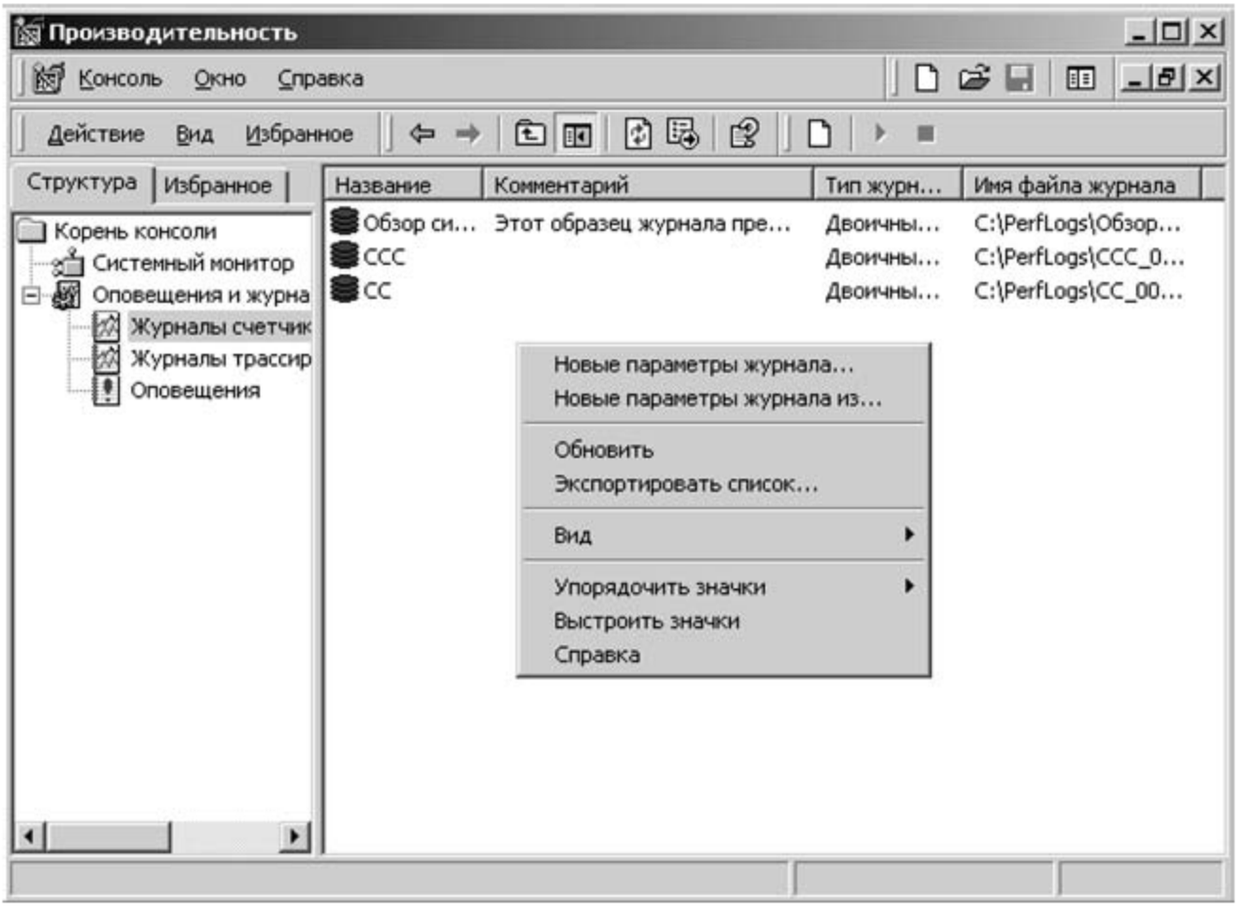


Рис. 2.20

4) в открывшемся окне ввести произвольное имя журнала и нажать кнопку OK;
5) в новом окне (рис. 2.21) на вкладке Общие добавить нужные счетчики и установить интервал съема данных;

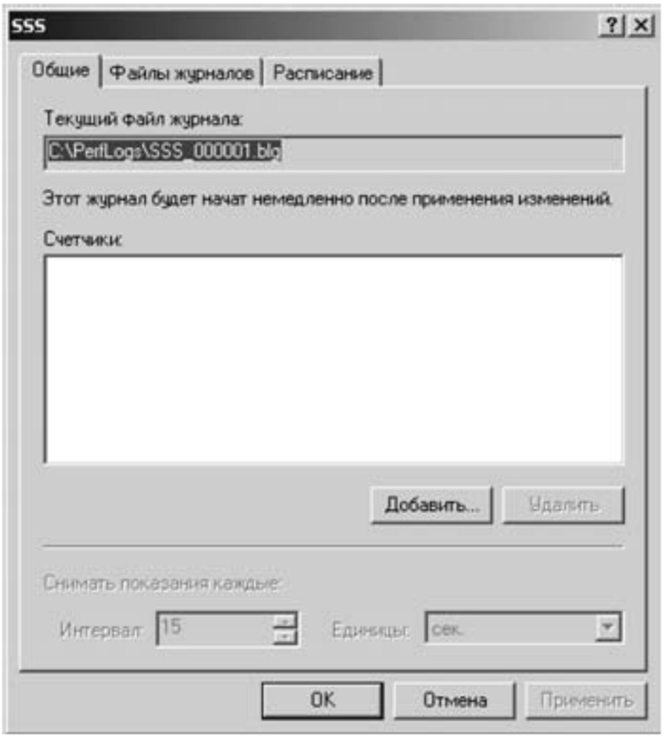


Рис. 2.21

6) на вкладке Файлы журналов можно выбрать размещение журнала, имя файла, добавить комментарий, указать тип журнала и ограничить его объем. Возможны следующие варианты:
   — текстовый файл CVS (данные сохраняются с использованием запятой в качестве разделителя);
   — текстовый файл TSV (данные сохраняются с использованием табуляции в качестве разделителя);
   — двоичный файл для регистрации прерывающейся информации;
   — двоичный циклический файл для регистрации данных с перезаписью;
7) на вкладке Расписание выбрать режим запуска и остановки журнала (вручную или по времени). Для запуска команды после закрытия журнала установить флажок Выполнить команду и указать путь к исполняемому файлу;
8) после установки всех значений нажать кнопки Применить и OK.

Задания для самостоятельной работы

1. Исследовать свои приложения с записью результатов в Журнал счетчиков, выбрав следующие счетчики: % загруженности работы

процессора в привилегированном и пользовательском режимах, % времени прерываний, % использования выделенной памяти, частота обращений к диску, скорость обмена с диском.

2. Выполнить следующие действия:
   — запустить журнал (частота съема данных 10 с, файл типа cvs);
   — запустить исследуемую программу;
   — через 2—3 мин остановить журнал;
   — просмотреть Результаты, открыв файл журнала в Excel (для удобства просмотра нужно, используя меню Формат, дать Перенос по словам для заголовков и растянуть ширину ячеек с числовыми данными, пример на рис. 2.22). Объяснить полученные результаты;

	A	B	C	D
1	(PDH-CSV 4.0) (Pacific Standard Time)(420)	\\CPQ12693627157\Processor(_Total)\% Privileged Time	\\CPQ12693627157\Processor(_Total)\% Processor Time	\\CPQ12693627157\Processor(_Total)\% User Time
2	12:005 AM	2,47E-07	100	1,12E-07
3	12:007 AM	31	100	69
4	12:009 AM	29	100	71
5	12:011 AM	24	100	76
6	12:013 AM	25	100	75
7	12:015 AM	28,5	100	71,5
8	12:017 AM	27,8607	100	72,1393
9	12:019 AM	30	100	70
10	12:021 AM	30,5	100	69,5
11	12:023 AM	27	100	73
12	12:025 AM	27	100	73
13	12:027 AM	26	100	74
14	12:029 AM	25,5	100	74,5
15	12:031 AM	30	100	70
16	12:033 AM	25,5	100	74,5
17	12:035 AM	26	100	74
18	12:037 AM	30,5	100	69,5
19	12:039 AM	32	100	68
20	12:041 AM	26,5	100	73,5
21	12:043 AM	25,5	100	74,5
22	12:045 AM	26,5	100	73,5
23	12:047 AM	28	100	72

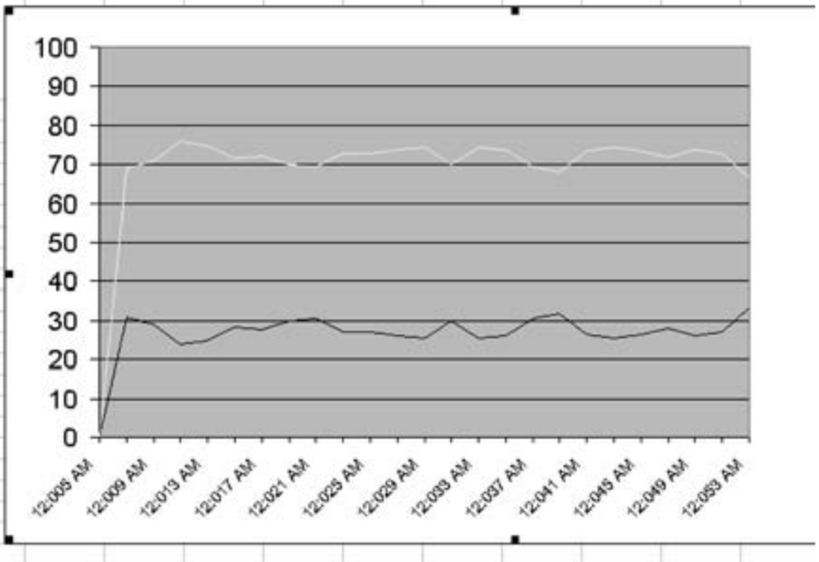


Рис. 2.22

   — исследовать программу еще раз, указав тип журнала — двоичный (чтобы потом можно было просмотреть диаграммы).

3. Просмотреть собранную информацию в консоли Производительность. Для этого выполнить следующие действия:
   — дважды щелкнуть по значку Системный монитор;
   — щелкнуть правой клавишей мыши в правом поле и выбрать в контекстном меню строку Свойства (рис. 2.23);
   — перейти на вкладку Источник;
   — щелкнуть на кнопке Файл журнала и указать его размещение, используя кнопку Обзор;

— нажать кнопку Диапазон времени и выбрать диапазон представления результатов, передвигая левую и правую планки;

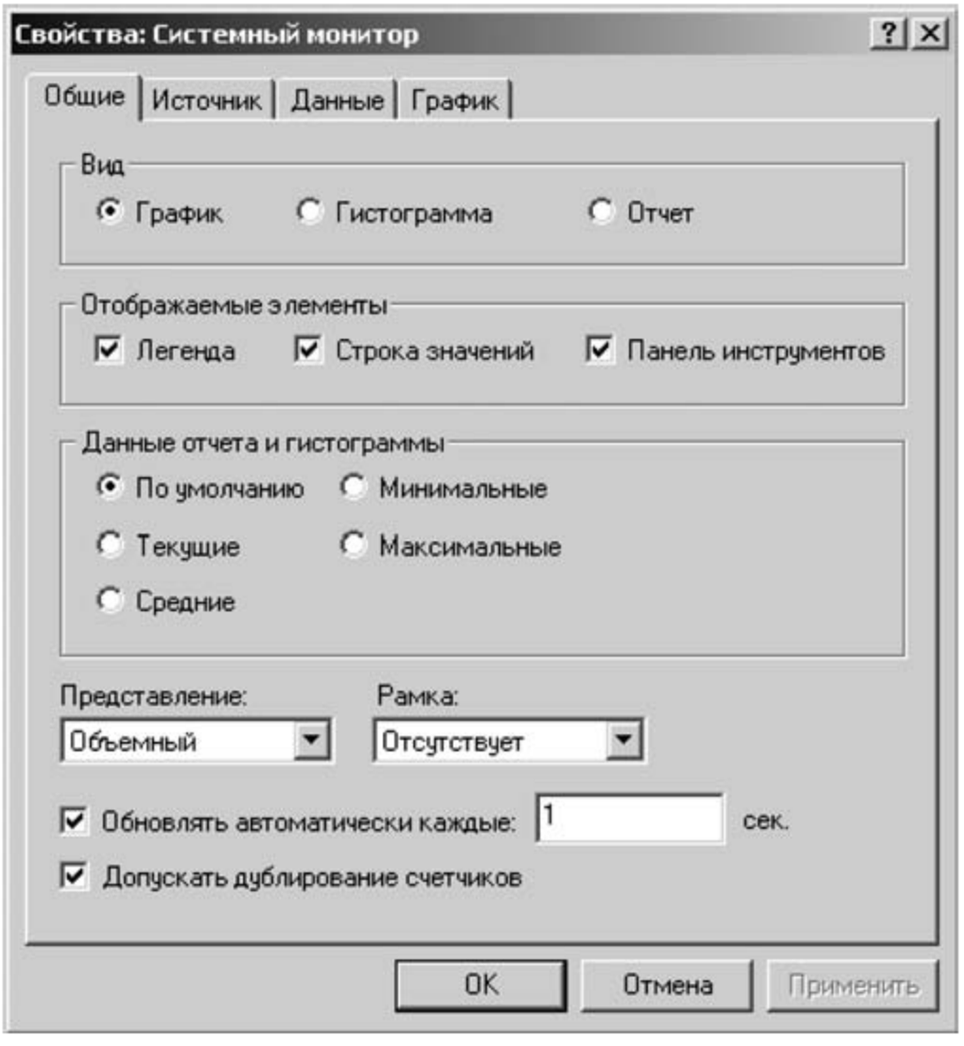


Рис. 2.23

— нажать кнопку OK;
— добавить счетчики, выбрав их из журнала (необязательно сразу все, можно просматривать отдельно каждый счетчик или несколько счетчиков);
— просмотреть полученные диаграммы. Объяснить полученные результаты.

2.5. Создание журнала трассировки и оповещений

В отличие от журналов счетчиков журналы трассировки находятся в ожидании определенных событий. Для интерпретации содержимого журнала трассировки необходимо использовать специальный анализатор.

Для создания журнала трассировки необходимо выполнить следующие действия:

1) запустить оснастку Производительность;
2) щелкнуть по значку Журналы трассировки;
3) щелкнуть правой кнопкой мыши в панели результатов и выбрать в контекстном меню пункт Новые параметры журнала (рис. 2.24);

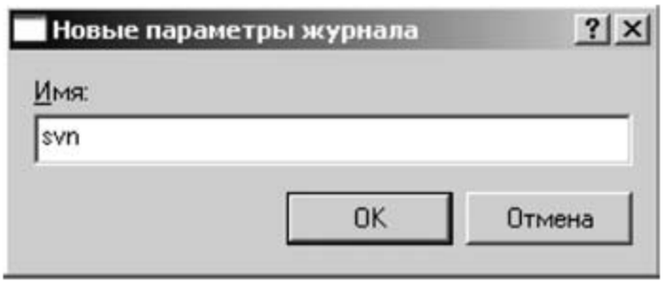


Рис. 2.24

4) в открывшемся окне ввести произвольное имя журнала и нажать кнопку OK;
5) по умолчанию файл журнала создается в папке PerfLogs в корневом каталоге и к имени журнала присоединяется серийный номер;
6) на вкладке Общие указать путь и имя созданного журнала (по умолчанию оно уже есть);
7) на этой же вкладке выбрать События, протоколируемые системным поставщиком или указать другого поставщика;
8) на вкладке Файлы журналов выбрать тип журнала:
   — файл циклической трассировки (журнал с перезаписью событий, расширение etl);
   — файл последовательной трассировки (данные записываются, пока журнал не достигнет предельного размера, расширение etl);
9) на этой же вкладке выбрать и размер файла;
10) на вкладке Дополнительно можно указать размер буфера журнала;
11) на вкладке Расписание выбрать режим запуска и остановки журнала (вручную или по времени).

В ряде случаев для обнаружения неполадок в организации вычислительного процесса удобно использовать Оповещения. С помощью этого компонента можно установить оповещения для выбранных счетчиков. При превышении или снижении относительно заданного значения выбранными счетчиками оснастка посредством сервиса Messenger оповещает пользователя.

Для создания оповещений необходимо выполнить следующие действия:

1) щелкнуть на значке Оповещения;
2) щелкнуть правой кнопкой мыши в панели результатов и выбрать в контекстном меню пункт Новые параметры оповещений;
3) в открывшемся окне ввести произвольное имя оповещения и нажать кнопку OK;
4) в появившемся окне (рис. 2.25) на вкладке Общие можно задать комментарий к оповещению и выбрать нужные счетчики;

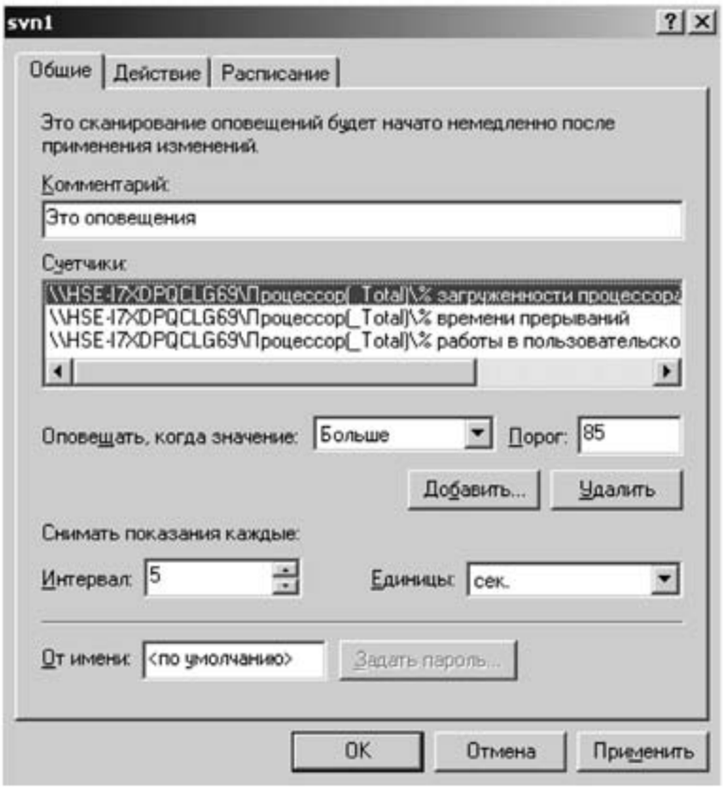


Рис. 2.25

5) в поле Оповещать выбрать предельные значения для счетчиков;
6) в поле Снимать показания каждые выбрать период опроса счетчиков;
7) на вкладке Действие (рис. 2.26) можно выбрать действие, которое будет происходить при запуске оповещения, например Послать сетевое сообщение и указать имя компьютера;
8) на вкладке Расписание выбрать режим запуска и остановки наблюдения.

Если в компьютере произойдет событие, предусмотренное в Оповещениях, в журнал событий Приложение будет сделана соответствующая запись. Для ее просмотра нужно зайти в оснастку Просмотр со-

бытий, где и можно увидеть сведения о событии, например такие, как показано на рис. 2.27.

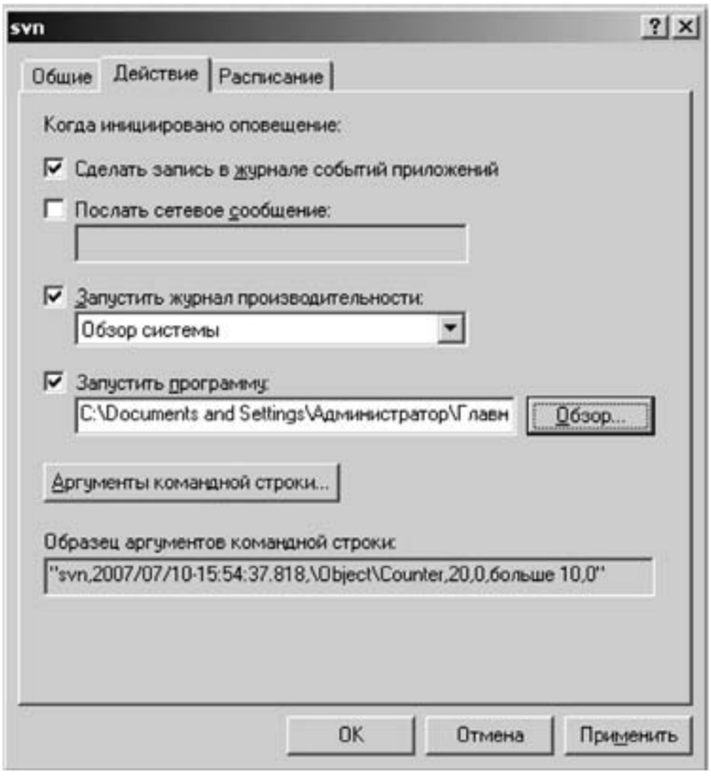


Рис. 2.26

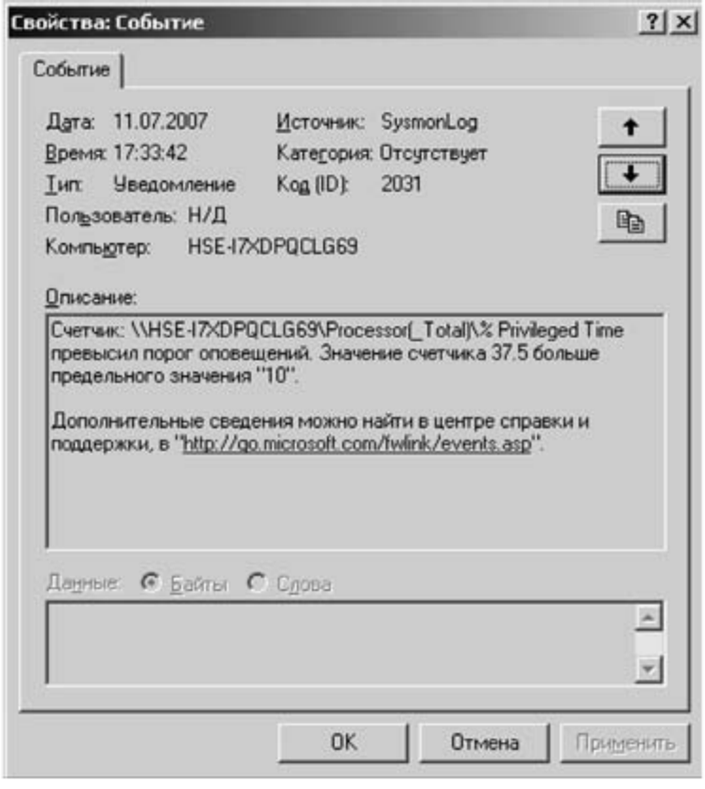


Рис. 2.27

Задания для самостоятельной работы

1. Создать журнал трассировки для исследования своего приложения.
2. Создать Оповещения по выбранным счетчикам для своего приложения.
3. Просмотреть журнал событий.
4. Объяснить полученные результаты.

2.6. Многопоточные вычислительные процессы

2.6.1. Внутреннее устройство, контекст и стек потока

Любой поток состоит из двух компонентов:
1) объекта ядра, через который ОС управляет потоком. Там же хранится статистическая информация о потоке;
2) стека потока, который содержит параметры всех функций и локальные переменные, необходимые потоку для выполнения кода.

Создав объект ядра — поток, система присваивает счетчику числа его пользователей начальное значение, равное двум. Затем система выделяет стеку потока память из адресного пространства процесса (по умолчанию резервирует 1 Мбайт адресного пространства процесса и передает ему всего две страницы памяти, далее память может добавляться). После этого система записывает в верхнюю часть стека два значения (стеки строятся от старших адресов памяти к младшим). Первое из них является значением параметра pvParam, который позволяет передать функции потока какое-либо инициализирующее значение. Второе значение определяет адрес функции потока pfnStartAddr, с которой должен будет начать работу создаваемый поток.

У каждого потока собственный набор регистров процессора, называемый контекстом потока. Контекст отображает состояние регистров процессора на момент последнего исполнения потока и записывается в структуру CONTEXT, которая содержится в объекте ядра — потоке.

Указатель команд (IP) и указатель стека (SP) — два самых важных регистра в контексте потока. Когда система инициализирует объект ядра — поток, указателю стека в структуре CONTEXT присваивается тот адрес, по которому в стек потока было записано значение pfnStartAddr, а указателю команд — адрес недокументированной функции BaseTbreadStart (находится в модуле Kernel32.dll).

Новый поток начинает выполнение этой функции, в результате чего система обращается к функции потока, передавая ей параметр pvParam. Когда функция потока возвращает управление, BaseTbreadStart вызывает ExitTbread, передавая ей значение, возвращенное функцией потока. Счетчик числа пользователей объекта ядра — потока — уменьшается на единицу, и выполнение потока прекращается.

При инициализации первичного потока его указатель команд устанавливается на другую недокументированную функцию — BaseProcessStart. Она почти идентична BaseTbreadStart. Единственное различие между этими функциями в отсутствии ссылки на параметр pvParam. Функция BaseProcessStart обращается к стартовому коду библиотеки C/C++/C#, который выполняет необходимую инициализацию, а затем вызывает входную функцию main, wmain, WinMain, Main. Когда входная функция возвращает управление, стартовый код библиотеки C/C++/C# вызывает ExitProcess.

Простейшую модель работы стека потока можно изучить по программе, приведенной в учебных материалах авторов на сайте www.hse.ru. При представлении стека в статической памяти для стека выделяется память, как для вектора. В дескрипторе этого вектора кроме обычных для вектора параметров должен находиться также указатель стека — адрес вершины стека. Указатель стека может указывать либо на первый свободный элемент стека, либо на последний записанный в стек элемент. (Все равно, какой из этих двух вариантов выбрать, важно в последствии строго придерживаться его при обработке стека.) В дальнейшем мы будем считать, что указатель стека адресует первый свободный элемент и стек растет в сторону увеличения адресов.

При занесении элемента в стек элемент (рис. 2.28) записывается на место, определяемое указателем стека, затем указатель модифицируется таким образом, чтобы он указывал на следующий свободный элемент (если указатель указывает на последний записанный элемент, то сначала модифицируется указатель, а затем производится запись элемента). Модификация указателя состоит в прибавлении к нему или в вычитании из него единицы (помните, что наш стек растет в сторону увеличения адресов).

Операция исключения элемента состоит в модификации указателя стека (в направлении, обратном модификации при включении) и выборке значения, на которое указывает указатель стека. После выборки слот, в котором размещался выбранный элемент, считается свободным. Операция очистки стека сводится к записи в указатель стека начального значения — адреса начала выделенной области памяти. Определение размера стека сводится к вычислению разности указателей: указателя стека и адреса начала области.

Общепринятым методом управления вызовами процедур и возвратами из них является использование стека. При обработке вызова процессор помещает в стек адрес возврата. При возврате из процедуры процессор использует адрес вершины стека. Модель позволяет в динамике демонстрировать обработку вложенных процедур (рис. 2.28). Для этого необходимо нажать кнопку Обработка вложенных процедур, а затем кнопку Запустить демонстрацию (рис. 2.29).

Рис. 2.28

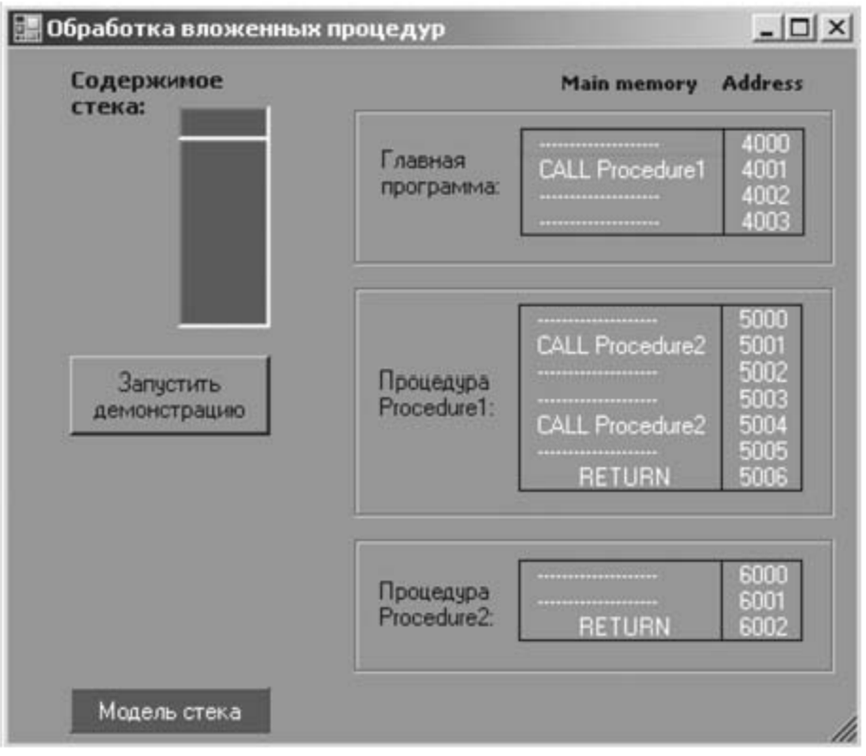


Рис. 2.29

2.6.2. Модель мультипрограммного вычислительного процесса

В учебных материалах авторов на сайте www.hse.ru приведена программа анимационной модели алгоритма планирования потоков, в которой организуются две очереди потоков: с высшими (от 16 до 31) и низшими (от 1 до 15) приоритетами. Также имеются две очереди потоков, ожидающих ввода-вывода или других событий соответственно для каждой из этих очередей потоков высшего и низшего приоритетов. Для простоты время событий ввода-вывода неизменно, в модели оно равно 20 единицам модельного времени. Квант времени процессора, выделяемого потокам, равен 50 единицам модельного времени. Время тайм-аута — двум единицам модельного времени. Время модели связано с системным временем через коэффициент timeRate = 0,05, т.е. время модели протекает в 20 раз медленнее реального, чтобы можно было увидеть происходящие в модели процессы.

После запуска сразу начинается процесс моделирования выполнения потоков, которые генерируются со случайным значением приоритета и времени выполнения. На рисунке 2.30 приведен один из вариантов возможной ситуации очередей, выполненных потоков и полученных временны́х характеристик.

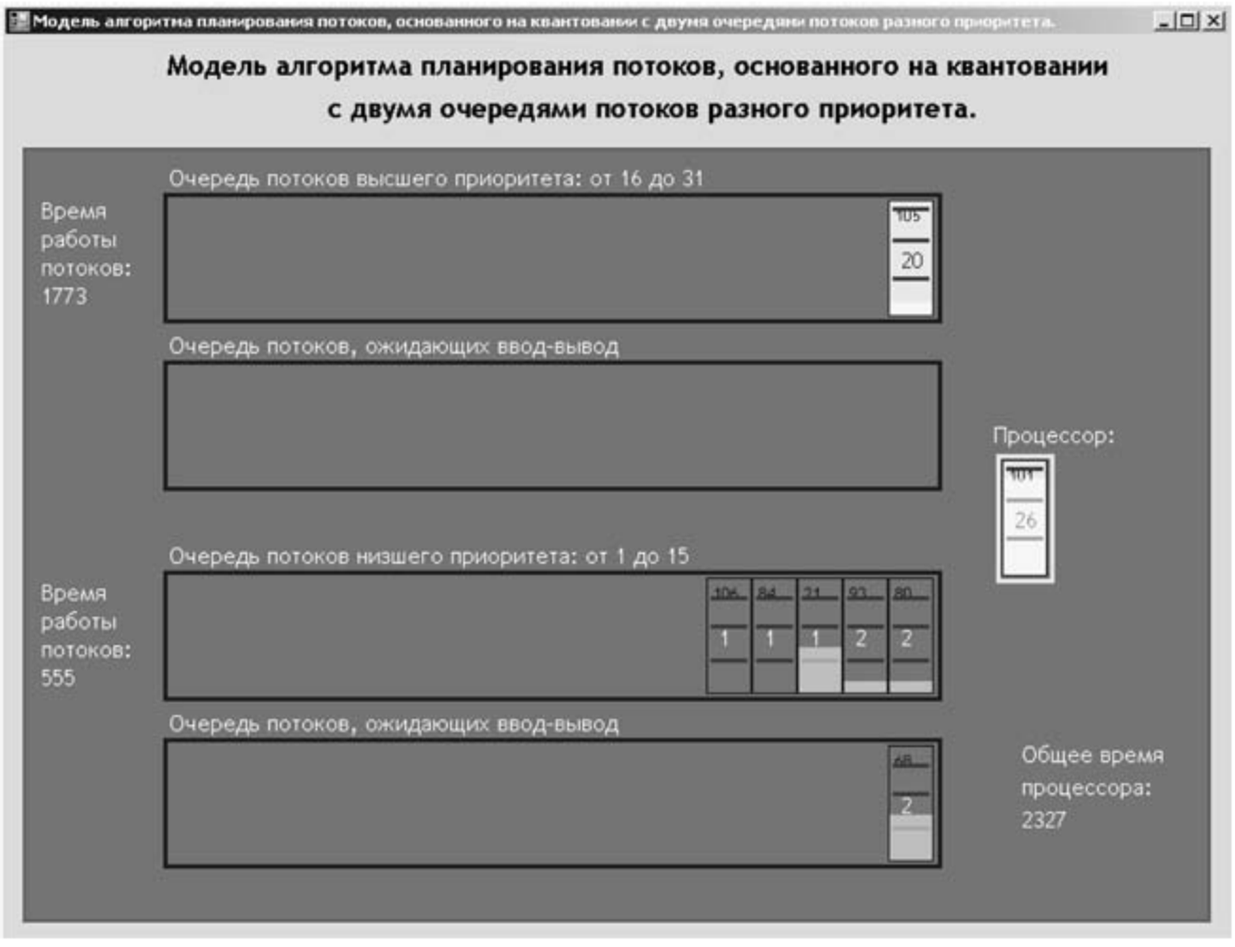


Рис. 2.30

На диаграмме видно, что на процессоре выполняется 101 поток, имеющий приоритет 26. Общее время выполнения потоков составляет 2327 единиц. В очереди потоков высшего приоритета находится поток с номером 105 и приоритетом 20. В очереди потоков низшего приоритета находятся пять потоков с номерами 80, 93, 21, 84 и 106. Поток с номером 68 находится в очереди, ожидающей завершения ввода-вывода. Процессор затратил 1773 единицы системного времени на обработку потоков высшего приоритета и 555 единицы системного времени на обработку потоков низшего приоритета.

Обратите внимание на два факта: потоки выстраиваются в очередь в порядке убывания приоритета, на каждом потоке светлая часть показывает время работы потока, которое уже им получено от процессора.

2.6.3. Планирование вычислительного процесса

В учебных материалах авторов на сайте www.hse.ru приведена программа, которая позволяет моделировать следующие алгоритмы распределения времени центрального процессора для очереди выполняе-

мых процессов: RR (Round Robin) — «круговая дисциплина обслуживания»; FCFS (First Com —First Served) — «первый пришел — первый обслужен»; SJF (Short Job First) — «короткая задача — первая» в двух вариантах: 1) вытесняющий алгоритм, т.е. с прерыванием процесса по истечении кванта времени; 2) невытесняющий алгоритм, т.е. с выполнением короткой задачи до ее завершения.

Окно программы после ее запуска показано на рис. 2.31. В верхней части окна представлены возможные алгоритмы для исследования. Среднее поле программы предназначено для ввода исходных данных. Здесь задается количество обрабатываемых процессов, причем для каждого процесса задаются время перехода в состояние «Готовность» (например, по причине ожидания ввода-вывода) и время выполнения в условных единицах.

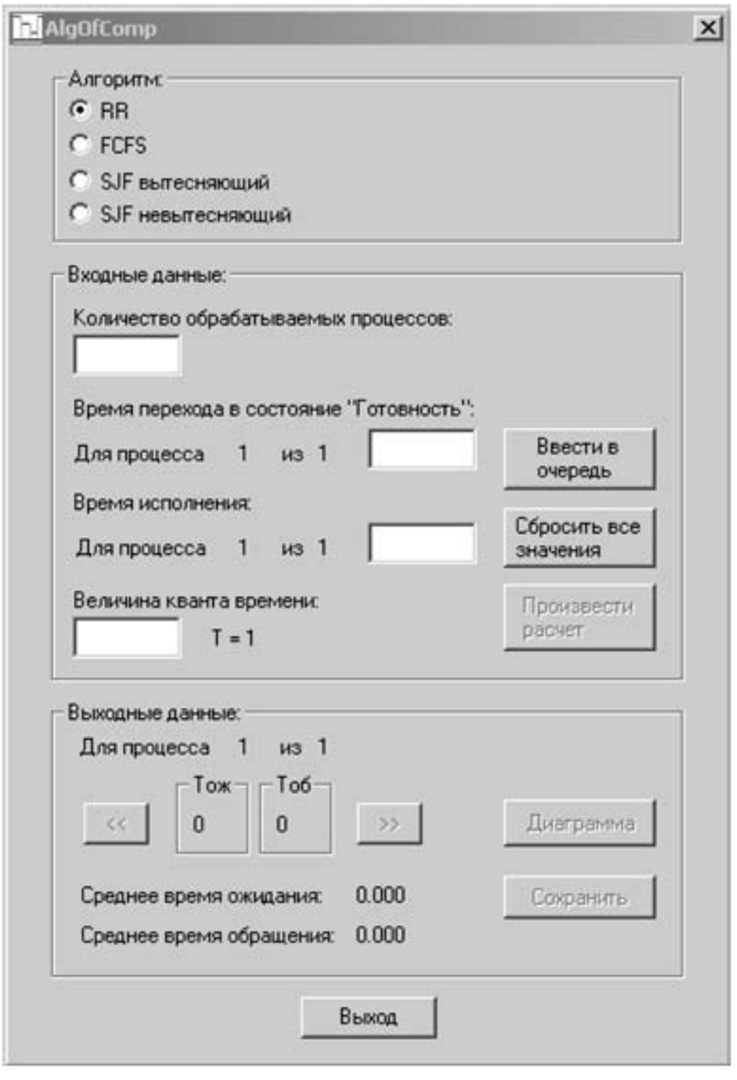


Рис. 2.31

После ввода исходных данных по всем процессам и задания величины кванта времени процессора в таких же условных единицах следу-

ет нажать кнопку Произвести расчет. В нижней части окна представляются полученные результаты. По каждому процессу даются среднее время ожидания и среднее время обслуживания (выполнения). Полученные результаты можно просмотреть в виде диаграммы, нажав кнопку Диаграмма (рис. 2.32), а также сохранить в файле, нажав кнопку Сохранить. Файл можно просмотреть в программе Блокнот (рис. 2.33). В правой части диаграммы даются обобщенные данные по результатам выполнения процессов.

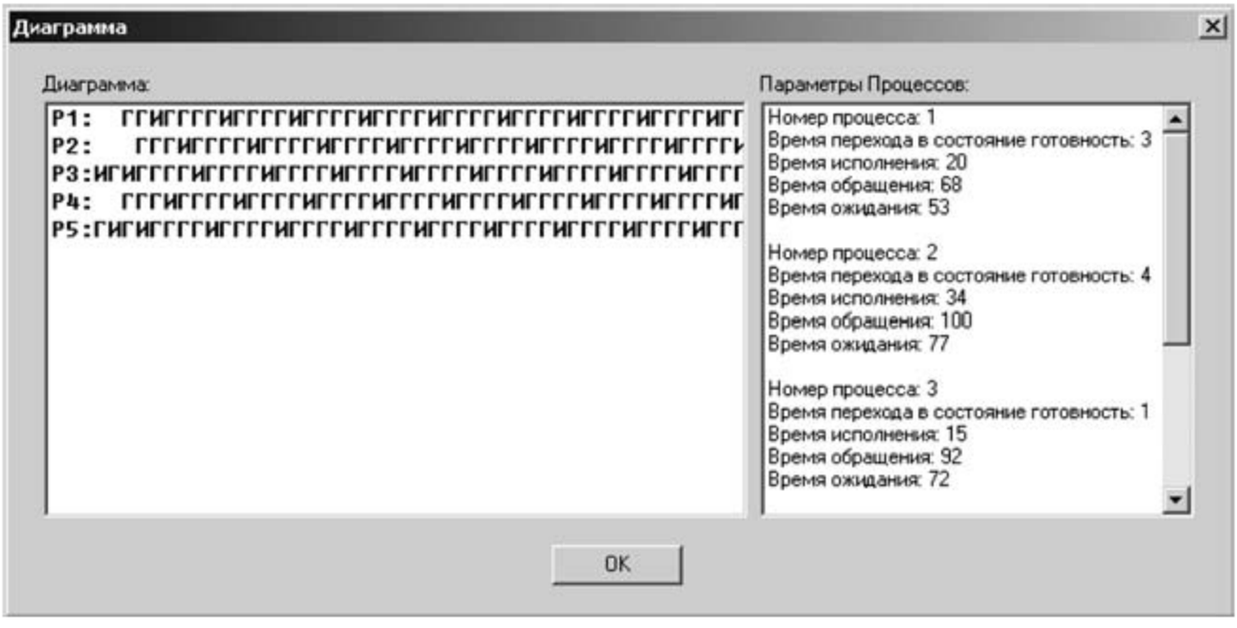


Рис. 2.32

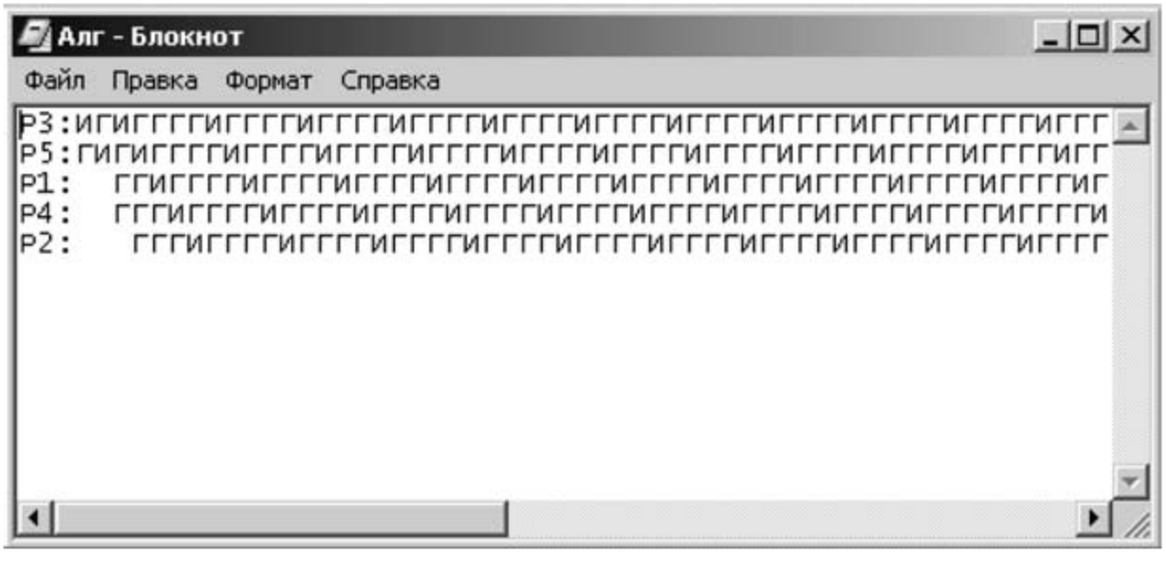


Рис. 2.33

Как отмечено выше, в основе многих вытесняющих алгоритмов планирования лежит концепция квантования. В соответствии с этой кон-

цепцией каждому потоку поочередно для выполнения предоставляется ограниченный непрерывный период процессорного времени — квант. Поток, который исчерпал свой квант, переводится в состояние готовности и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый поток из очереди готовых к обработке потоков.

Задания для самостоятельной работы

1. Провести исследование различных алгоритмов планирования процессов, используя программу AlgOfComp. В качестве исходных данных сформировать 10 процессов со временем перехода в состояние «Готовность» в диапазоне 1—5 усл. ед. и временем исполнения в диапазоне 10—50 усл. ед. Исследования провести для различных значений квантов времени процессора 1, 2, 3, 4, 5. По полученным результатам построить графики в электронной таблице Excel. Провести анализ эффективности того или иного алгоритма в зависимости от величины кванта времени процессора и характеристик процессов.
2. Разработать программу, с помощью которой можно сравнить эффективность обработки потоков на основе квантования и дисциплины FIFO. Пользовательский интерфейс оформить в соответствии с рис. 2.34.

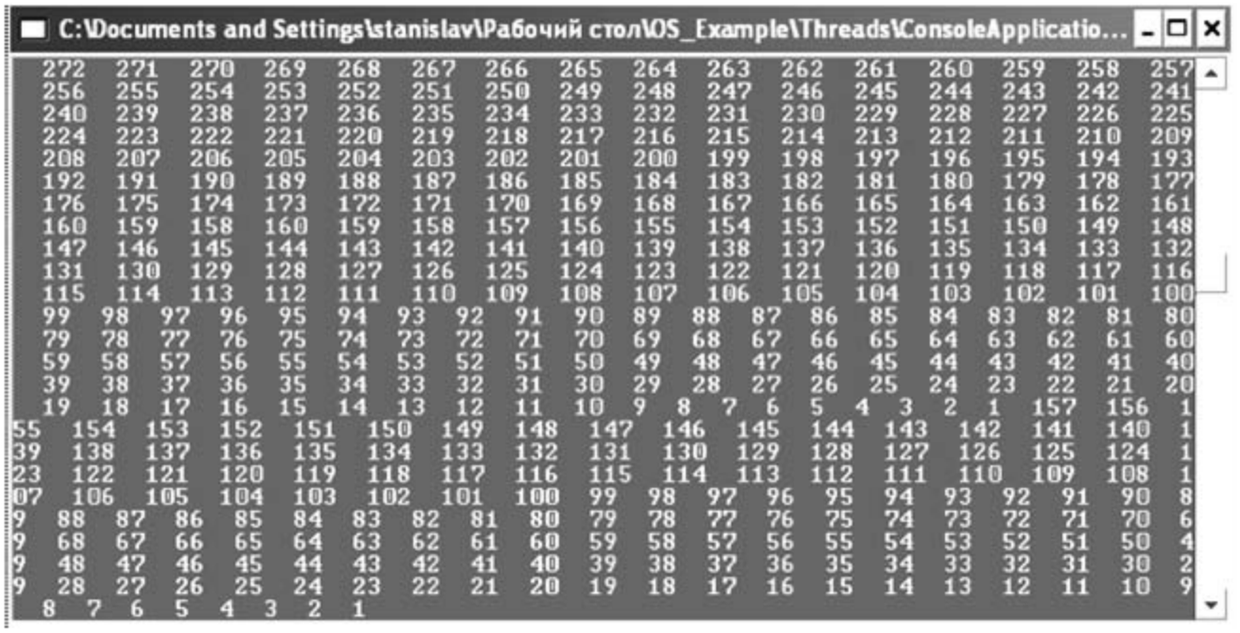


Рис. 2.34

2.7. Управление потоками

2.7.1. Создание потоков в приложении

Все потоки мультипрограммного вычислительного процесса должны иметь доступ к системным ресурсам — кучам, портам, файлам, окнам и т.д. Потоки взаимодействуют друг с другом в двух основных случаях: совместно используя разделяемый ресурс и уведомляя друг друга о завершении каких-либо операций. С вопросами синхронизации потоков связана обширная тематика, которую мы частично рассмотрим. Предварительно нужно научиться создавать многопоточные процессы. Наиболее просто это сделать, используя современные системы программирования, например MS Visual Studio 2005 и языковые средства (например, C#), предоставляемые этой системой.

Для создания нового потока в среде .NET нужно создать экземпляр класса Thread. Конструктор этого класса принимает один параметр — экземпляр делегата ThreadStart. Этот делегат представляет метод, который будет вызываться для выполнения операций созданного потока. Шаблон для создания потока выполнения выглядит следующим образом [10]:

```
public static void Method ()
    {
        {
            // Тело метода
        }
    }
public static void Main ()
{
    Thread t2 = new Thread (new ThreadStart (Method ()));
    ...
// Пример создания потоков
using System;
using System.Threading;

namespace ConsoleApplication1
{
    class Create_Thred
    {
        // Метод Coundown считает от 1000 до 1
```

```
        public static void Coundown()
        {
            for (int counter = 1000; counter > 0; counter--)
                Console.Write(counter.ToString() + " ");
        }
        static void Main()
        {
            // Создание второго потока
            Thread t2 = new Thread(new ThreadStart(Coundown));
            // Запуск потока
            t2.Start();
            //
            // В это же время метод Coundown вызывается в
        // основном потоке
            Coundown();
            Console.ReadLine();
        }
    }
}
```

В этой программе создается поток с именем t2 и вызывается метод Start. После того как метод Start вызван, начинается выполнение метода Countdown. В то же время метод Countdown вызывается из основного потока. Основная идея этого примера — заставить два метода Countdown выполняться одновременно.

На рисунке 2.34 приведен возможный результат выполнения этой программы. Если внимательно рассмотреть рисунок, то можно заметить, что информация, выводимая разными потоками, перемешана. Это происходит потому, что, хотя процессор выполняет потоки раздельно, они используют одни и те же ресурсы — дисплей компьютера. Если выполнить программу несколько раз, можно получить различные результаты. На самом деле невозможно предсказать, когда процессор переключится с одного потока на другой. Это связано с тем, что потоки имеют один и тот же приоритет, т.е. равноправны для системы.

2.7.2. Приоритеты потоков

Один из основных принципов многопоточной архитектуры заключается в том, что не все потоки имеют одинаковые приоритеты. Несмотря на то что все потоки создаются одинаковыми (с приорите-

том Normal), не всегда целесообразно оставлять приоритеты потоков, установленные при их создании.

Например, если производится фоновая печать одновременно с редактированием документа, то ясно, что отдавать половину вычислительных ресурсов процессу печати невыгодно. Намного лучше отдавать большее количество ресурсов основному приложению — в этом случае оно будет быстрее реагировать на запросы пользователя, а печать все равно закончится, может быть несколько позже.

Чтобы выполнить подобное перераспределение ресурсов, используется свойство Priority, которое имеет пять допустимых значений:

1) Lowest (низший);
2) BelowNormal (ниже нормального);
3) Normal (нормальный);
4) AboveNormal (выше нормального);
5) Highest (наивысший).

Следует иметь в виду, что среда .NET изначально проектировалась в расчете на перенос на несколько платформ, поэтому перечисленные значения приоритетов могут не совпадать со значениями конкретной ОС. Например, в Windows 2000 существует шесть основных приоритетов, а в Windows CE 3.0 — 256 градаций приоритета, обозначаемых числами от 0 до 255.

Если в следующей далее программе установить потокам одинаковый приоритет, то результат работы программы будет таким, как это показано на рис. 2.35, *а*. Если этого не делать, а оставить программу такой, как она приведена ниже, то поток более высокого приоритета завершается раньше (рис. 2.35, *б*).

а *б*

```
// Пример потоков с приоритетами
using System;
using System.Threading;
namespace ConsoleApplication2
{
    class Prior_Thread
    {
        // Метод Coundown считает от 1000 до 1
        public static void Coundown()
        {
            for (int counter = 1000; counter > 0; counter--)
                Console.Write(counter.ToString() + " ");
        }
```

```
            static void Main()
            {
            // Создание второго потока
            Thread t2 = new Thread(new ThreadStart(Coundown));
            // Установка второму потоку высшего приоритета
            t2.Priority = ThreadPriority.Highest;
            // Установка текущему потоку низшего приоритета
            Thread.CurrentThread.Priority = ThreadPriority.Lowest;
            // Запуск второго потока
            t2.Start();
            // В то же время несколько измененный
// метод Coundown
            // выполняется в первом потоке
            for (int counter = 1000; counter > 0; counter--)
                  Console.Write(counter.ToString() + "a");
                  Console.ReadLine();
            }
      }
}
```

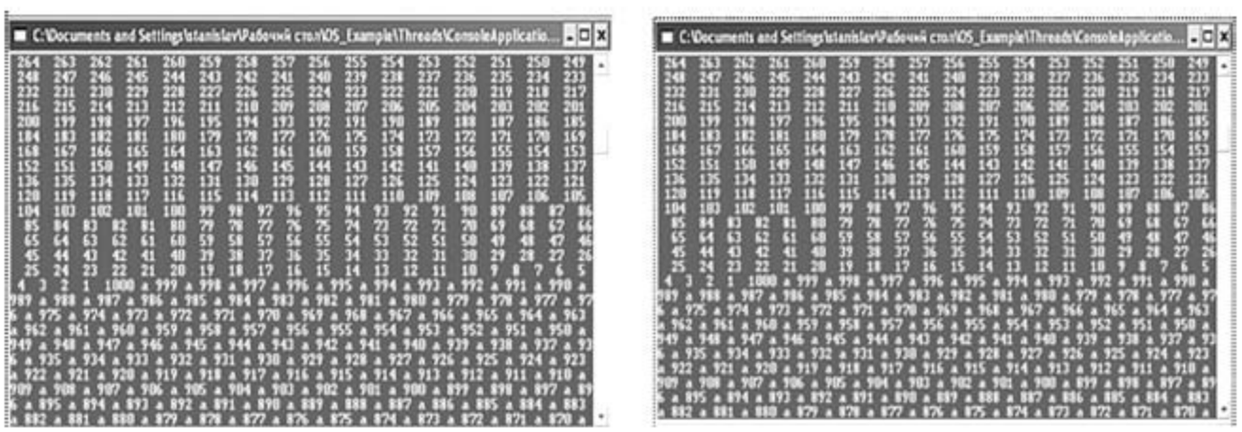


Рис. 2.35

2.7.3. Состояния потоков

Объекты класса Thread за время своего существования могут пребывать в разных состояниях. Для получения текущего состояния потока используется свойство ThreadState. Оно возвращает одно из десяти возможных значений:

1) Unstarted — не запущен;
2) Running — выполняется;
3) Background — фоновый;

4) WaitSleepJoin — заблокирован в результате вызова методов Wait, Sleep или Join;
5) SuspendRequested — запрос на приостановку;
6) Suspended — приостановлен;
7) StopRequested — запрос на остановку;
8) Stopped — остановлен;
9) AbortRequested — запрос на отмену (был вызван метод Abort, но поток еще не получил исключение ThreadAbortException;
10) Aborted — отменен.

Обычно сразу после создания объект класса Thread находится в состоянии Unstarted. После вызова метода Thread.Start он переходит в состояние Running. Если после этого свойству Is Background присвоить значение true, то поток перейдет в состояние Background, и т.д.

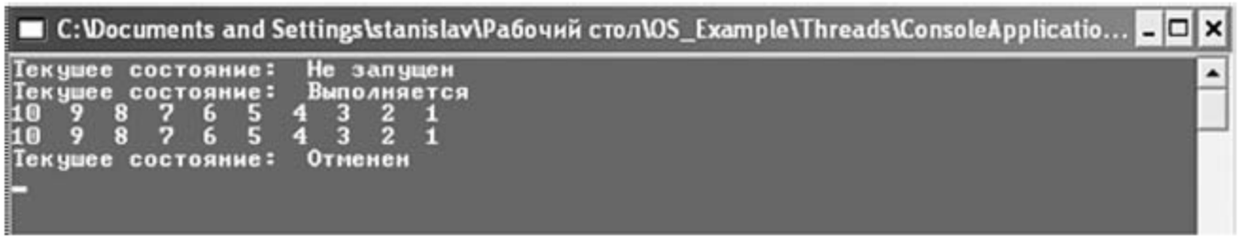
Поток может находиться в нескольких состояниях в один и тот же момент времени. Например, поток, ждущий освобождения ресурса, может быть одновременно в состоянии WaitSleepJoin и, если в ходе ожидания был вызван метод Abort, в состоянии AbortRequested. Поэтому для проверки состояния потока часто приходится использовать логические конструкции. Такой подход показан в следующей программе:

```
// Анализ состояния потока
using System;
using System.Threading;
namespace ConsoleApplication3
{
class StateThreads
    {
        // Метод Countdown считает от 10 до 1
        public static void Countdown()
        {
            for (int counter = 10; counter > 0; counter--)
                Console.Write(counter.ToString() + " ");
            Console.WriteLine();
        }
        // Метод DumpThreadState выводит текущее состояние потока
        // ThreadState — битовая маска состояния потока
public static void DumpThreadState (Thread t)
{
        Console.Write("Текушее состояние: ");
```

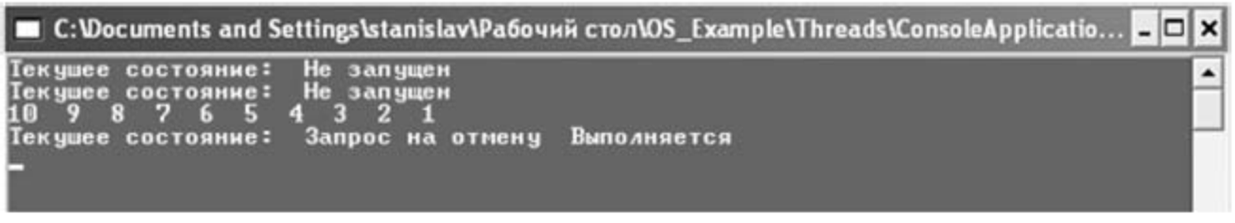
```
        if ((t.ThreadState & ThreadState.Aborted) = = ThreadState.Aborted)
            Console.Write ("Отменен ");
        if ((t.ThreadState & ThreadState.AbortRequested) = =
            ThreadState.AbortRequested)
            Console.Write ("Запрос на отмену ");
        if ((t.ThreadState & ThreadState.Background) = =
            ThreadState.Background)
            Console.Write ("Выполняется в фоновом режиме ");
        if ((t.ThreadState & (ThreadState.Stopped | ThreadState.Unstarted |
            ThreadState.Aborted)) = = 0)
            Console.Write ("Выполняется ");
        if ((t.ThreadState & ThreadState.Stopped) = = ThreadState.Stopped)
            Console.Write ("Остановлен ");
        if ((t.ThreadState & ThreadState.StopRequested) = =
            ThreadState.StopRequested)
            Console.Write ("Запрос на остановку ");
        if ((t.ThreadState & ThreadState.Suspended) = =
            ThreadState.Suspended)
            Console.Write ("Приостановлен ");
        if ((t.ThreadState & ThreadState.SuspendRequested) = =
            ThreadState.SuspendRequested)
            Console.Write ("Запрос на приостановку ");
        if ((t.ThreadState & ThreadState.Unstarted) = =
            ThreadState.Unstarted)
            Console.Write ("Не запущен ");
        if ((t.ThreadState & ThreadState.WaitSleepJoin) = =
             ThreadState.WaitSleepJoin)
            Console.Write ("Ожидает освобождения ресурсов ");
            Console.WriteLine();
}
        static void Main()
        {
            // Создание еще одного потока
            Thread t2 = new Thread(new ThreadStart(Countdown));
            DumpThreadState(t2);
            // Запуск нового потока
            t2.Start();
            DumpThreadState(t2);
            // В это же время метод Countdown вызывается в
            // основном потоке
```

```
            Countdown();
            // Отмена второго потока
            t2.Abort();
            DumpThreadState(t2);
            Console.ReadLine();
        }
    }
}
```

Эта программа может вывести строки, изображенные на рис. 2.36.



а



б

в

Рис. 2.36

2.7.4. Порядок выполнения потоков

Как было показано в предыдущих примерах, обычно порядок выполнения потоков не определен. Однако существуют способы управления потоками, позволяющие сделать их поведение более предсказуемым.

Класс Timer позволяет выполнять потоки через повторяющиеся промежутки времени, *класс Join* позволяет одному потоку ждать завершения другого. *Классы Lock, Interlocked, Monitor и Mutex* позволяют ко-

ординировать действия нескольких потоков и использование ресурсов несколькими потоками.

При помощи класса Timer можно добавить в программу поток, работающий по принципу «вызвали и забыли». При создании экземпляра класса Timer указывается четыре параметра:

1) Callback. Делегат типа TimerCallback, представляющий метод, который будет вызываться таймером;
2) State. Объект, который передается методу TimerCallback. Его можно использовать для того, чтобы таймеру было доступно некоторое постоянное состояние. Если такой необходимости нет, то этот параметр может быть равным null;
3) DueTime. Количество миллисекунд перед первым срабатыванием таймера;
4) Period. Количество миллисекунд между срабатываниями таймера.

В следующем листинге показан пример использования класса Timer:

```
// Использование класса Timer
using System;
using System.Threading;

namespace ConsoleApplication4
{

    class ManagementThreads
    {
        // Метод CheckTime вызывается по таймеру
        public static void CheckTime(Object state)
        {
            Console.WriteLine(DateTime.Now);
        }
        static void Main()
        {
            // Создание делегата, который будет вызываться объектом
            Timer
            TimerCallback tc = new TimerCallback(CheckTime);
            // Создание таймера, срабатывающего дважды в секунду
            // Первый запуск произойдет через одну секунду
            Timer t = new Timer(tc, null, 1000, 500);
            // Ожидание ввода пользователя
```

```
            Console.WriteLine("Нажмите Enter для выхода");
            Console.Read();
            // Освобождение ресурсов
            t.Dispose();
            t = null;
        }
    }
}
```

Эта программа может вывести строки, показанные на рис. 2.37.

Рис. 2.37

2.7.5. Последовательное выполнение потоков

Метод, указанный в конструкторе объекта Timer, выполняется в отдельном потоке, созданном системой, а не в потоке, создавшем объект Timer. Обратите внимание на вызов метода Dispose объекта Timer. Это делается для того, чтобы корректно освободить уже ненужные ресурсы, выделенные отдельному потоку.

Метод Thread.Join позволяет «прикрепить» один поток к другому. Это означает, что первый, например, поток, будет ждать завершения второго, после чего будет запущен. Использование этого метода показано в следующем листинге.

```
// Использование метода Join
using System;
using System.Threading;
namespace ConsoleApplication5
```

```
{
    class Waite_End_Thread
    {
        // Метод Countdown считает от 100 до 1
        public static void Countdown()
        {
            for (int counter = 100; counter > 0; counter--)
                Console.Write(counter.ToString() + " ";
        }
        // Метод Countdown1 считает от 100 до 1 с буквой A
        public static void Countdown1()
        {
            for (int counter = 100; counter > 0; counter--)
                Console.Write(counter.ToString() + "A");
        }

        static void Main()
        {
            // Создание второго потока
            Thread t2 = new Thread(new ThreadStart(Countdown));
            // Запуск второго потока
            t2.Start();
            // Блокировка первого потока до завершения второго
            t2.Join();
            // Вызов метода Countdown1 из первого потока
            Countdown1();
            Console.Read();
        }
    }
}
```

Если выполнить эту программу (рис. 2.38), то можно заметить, что вывод двух методов Countdown не пересекается, несмотря на то, что оба потока выполняются с нормальным приоритетом. Это произошло потому, что вызов t2.Join приостанавливает основной поток до тех пор, пока не завершится выполнение метода Countdown во втором потоке.

Метод Join имеет перегруженную версию, которая позволяет указать максимальное время ожидания. Например, для того чтобы ждать

завершения второго потока не дольше 5 с, можно использовать оператор t2.Join (5000).

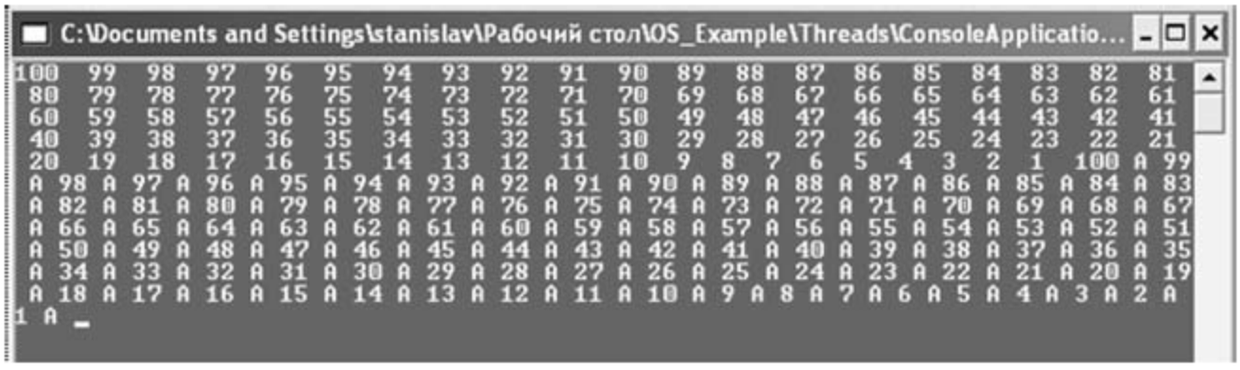


Рис. 2.38

Задания для самостоятельной работы

1. Модифицировать рассмотренные в этом разделе программы, организовав в них три и более потоков.
2. Подумайте, как изменится время выполнения многопоточной программы, увеличивающей значение глобальной переменной, при увеличении количества потоков. Не забывайте, что каждое переключение контекста требует определенных расходов системных ресурсов.
3. Продумайте параллельный алгоритм решения какой-либо задачи и составьте по нему многопоточную программу.

2.8. Проблемы многопоточных программ

2.8.1. Критические секции

При использовании многопоточной программы возникает ряд проблем, которые не существуют в однопоточных программах. Например, как поведет себя программа, если два потока используют одну и ту же переменную? Результат может отличаться от ожидаемого. В следующем листинге показан многопоточный код с проблемой такого рода — критической секцией, которая представляет собой участок кода, требующий монопольного доступа к каким-то общим данным.

```
// Многопоточная работа с проблемами критической секции
using System;
```

```
using System.Threading;
namespace ConsoleApplication6
{
    class CriticalSection
    {
        // Общий счетчик
        int Runs = 0;
        // Метод CountUp увеличивает значение общего счетчика
        public void CountUp()
        {
        while (Runs <= 10)
        {
            int Temp = Runs;
        Temp++;
            Console.WriteLine(Thread.CurrentThread.Name +
        " "+ Temp);
            Thread.Sleep(1000);
            Runs = Temp;
        }
        }
        public void RunThreads()
        {
            // Создание и запуск двух потоков
            Thread t2 = new Thread(new ThreadStart(CountUp));
            t2.Name = "t2";
            Thread t3 = new Thread(new ThreadStart(CountUp));
            t3.Name = "t3";
            t2.Start();
            t3.Start();
        }
        static void Main()
        {
            // Создание экземпляра текущего класса
            CriticalSection Section = new CriticalSection();
            // Выполнение тестов
            Section.RunThreads();
            Console.Read();
        }
    }
}
```

Эта программа выводит строки, изображенные на рис. 2.39. Если вы ожидали увидеть десять повторений цикла в методе CountUp, то очень удивитесь, увидев результат. Проблема состоит в том, что увеличение переменной Runs производится не за один шаг. Дело в том, что в процессе получения значения переменной, увеличения его на единицу и присвоения нового значения могут происходить другие действия. Метод Thread.Sleep указывает объекту Thread освободить процессор на указанное количество миллисекунд.

Что же произошло в нашем случае? Приведем последовательность выполняемых действий:

1) поток t2 присваивает переменной Temp значение переменной Runs и увеличивает Temp на единицу;
2) поток t2 приостанавливает свое выполнение на 1 с;
3) планировщик передает управление потоку t3;

Рис. 2.39

4) поток t3 присваивает переменной Temp значение переменной Runs и увеличивает Temp на единицу. Здесь используются те же значения, что и в потоке t2, так как поток t2 еще не успел изменить значение общего счетчика (он «заснул» на 1000 мс);
5) поток t3 приостанавливает свое выполнение на 1 с;
6) поток t2 возобновляет выполнение и присваивает новое значение переменной Runs;
7) поток t3 возобновляет выполнение и присваивает новое значение переменной Runs, изменяя значение, присвоенное потоком t2;
8) цикл повторяется.

Заметим, что в этом случае метод Sleep используется для того, чтобы сделать борьбу за ресурсы очевидной. Подобная ситуация может сложиться и без использования метода Sleep, так как планировщик может переключиться на выполнение другого потока в любом месте цикла, перед тем как переменной Runs присвоено новое значение.

2.8.2. Блокировка

Решить проблему критической секции можно различными средствами [8, 16]. Например, использованием оператора lock. Образно говоря, оператор lock означает «не позволять другим потокам выполнять эту часть кода до тех пор, пока я ее выполняю». Оператор позволяет заблокировать любой объект. В приведенном далее листинге оператор lock (this) блокирует код всего класса:

```
// Выделение критической секции с помощью Lock
using System;
using System.Threading;

namespace ConsoleApplication7
{
    class SectionLock
    {
        // Общий счетчик
        int Runs = 0;
        // Метод CountUp увеличивает значение общего счетчика
        public void CountUp()
        {
            while (Runs <= 10)
            {
                lock(this)
                {
                    int Temp = Runs;
                    Temp++;
                    Console.WriteLine(Thread.CurrentThread.Name +
                            " " + Temp);
                    Thread.Sleep(1000);
                    Runs = Temp;
                }
            }
        }
```

```
        public void RunThreads()
        {
            // Создание и запуск двух потоков
            Thread t2 = new Thread(new ThreadStart(CountUp));
            t2.Name = "t2";
            Thread t3 = new Thread(new ThreadStart(CountUp));
            t3.Name = "t3";
            t2.Start();
            t3.Start();
        }
        static void Main()
        {
            // Создание экземпляра текущего класса
            SectionLock Section = new SectionLock();
            // Выполнение тестов
            Section.RunThreads();
            Console.Read();
        }
    }
}
```

Использование оператора lock позволило обеспечить правильную работу программы, приведенной в предыдущем примере (рис. 2.40).

Рис. 2.40

2.8.3. Монитор

Ключевое слово lock, рассмотренное в предыдущем примере, это на самом деле сокращенный способ использования класса Monitor. Класс Monitor предоставляет гибкие способы синхронизированного доступа к любому объекту кода программы. В следующем листинге

предыдущий пример переписан с явным использованием класса Monitor:

```
// Использование класса монитор
using System;
using System.Threading;
namespace ConsoleApplication8
{
    class Section_Monitor
    {
        // Общий счетчик
        int Runs = 0;
        // Метод CountUp увеличивает значение общего счетчика
        public void CountUp()
        {
            while (Runs < 10)
            {
                Monitor.Enter(this);
                int Temp = Runs;
                Temp++;
                Console.WriteLine(Thread.CurrentThread.Name +
                    " "+ Temp);
                Thread.Sleep(1000);
                Runs = Temp;
                Monitor.Exit(this);
            }
        }
        public void RunThreads()
        {
            // Создание и запуск двух потоков
            Thread t2 = new Thread(new ThreadStart(CountUp));
            t2.Name = "t2";
            Thread t3 = new Thread(new ThreadStart(CountUp));
            t3.Name = "t3";
            t2.Start();
            t3.Start();чясячсячс
        }
        static void Main()
        {
            // Создание экземпляра текущего класса
            Section_Monitor Section = new Section_Monitor();
```

```
            // Выполнение тестов
            Section.RunThreads();
            Console.Read();
        }
    }
}
```

Результат работы этой программы (рис. 2.41) в данном случае не отличается от предыдущего.

Рис. 2.41

Методы Monitor.Enter и Monitor.Exit работают точно так же, как и ключевое слово lock. Но класс Monitor имеет значительно больше возможностей, чем простая блокировка. Далее перечислены методы класса Monitor:

Метод	Описание
Enter()	Блокирует объект, переданный монитору. Если другой поток уже блокировал этот объект, то выполнение текущего пока будет приостановлено до тех пор, пока другой поток не освободит объект
Exit ()	Снимает блокировку с объекта
Pulse ()	Информирует следующий ждущий поток, что монитор временно закончил работу с объектом и поток может продолжить выполнение
PulseAll ()	Сигнализирует всем потокам о том, что объект скоро будет освобожден
TryEnter ()	Пытается заблокировать переданный объект. Если объект может быть заблокирован, возвращает true, в противном случае возвращает false
Wait ()	Освобождает все блокировки и приостанавливает выполнение текущего потока до тех пор, пока другой поток не вызовет метод Pulse

2.8.4. Семафоры

Другой вариант решения рассматриваемой проблемы связан с использованием объектов ядра ОС — семафоров. Семафоры используются для синхронизации потоков разных процессов. Это обусловлено тем, что семафор как объект реализован не в каком-либо приложении, а в ядре ОС Windows. Если два процесса создают объекты Mutex, передав конструкторам одинаковое имя, то они получают один и тот же семафор.

В определенный момент времени семафор может принадлежать только одному потоку. В данном варианте конструктор класса Mutex принимает два параметра (существуют также перегруженные версии конструктора). Первый параметр булевого типа определяет, должен ли объект изначально принадлежать потоку, создавшему его. Второй параметр представляет имя семафора.

Поток может вызвать метод WaitOne для получения семафора. Если никакой другой поток не использует семафор, тот его получает поток, вызвавший метод WaitOne. Если семафор используется другим потоком, то поток, вызвавший метод WaitOne, блокируется до тех пор, пока семафор не освободится. Когда работа, связанная с семафором, завершена, поток может вызвать метод ReleaseMutex, что приведет к освобождению семафора для другого ожидающего потока.

Класс Mutex также имеет метод WaitAll, ждущий до тех пор, пока все семафоры группы освободятся, и метод WaitAny, ожидающий до тех пор, пока освободится какой-либо семафор из группы:

```
// Использование семафора
using System;
using System.Threading;
namespace ConsoleApplication8
{
    class Semafor
    {
        // Общий счетчик
        static int Runs = 0;
        // Семафор
        static Mutex mtx;
        // Метод CountUp увеличивает значение общего счетчика
        public static void CountUp()
        {
            while (Runs < 10)
            {
                // Получение семафора
```

```
                mtx.WaitOne();
                int Temp = Runs;
                Temp++;
                Console.WriteLine(Thread.CurrentThread.Name +
                    " " + Temp);
                Runs = Temp;
                // Освобождение семафора
                mtx.ReleaseMutex();
                Thread.Sleep(1000);
            }
        }
        static void Main()
        {
            // Создание семафора
            mtx = new Mutex(false, "RunsMutex");
            // Создание и запуск двух потоков
            Thread t2 = new Thread(new ThreadStart(CountUp));
            t2.Name = "t2";
            Thread t3 = new Thread(new ThreadStart(CountUp));
            t3.Name = "t3";
            t3.Start();
            t2.Start();
            Console.Read();
        }
    }
}
```

Результат работы этой программы приведен на рис. 2.42.

Задание для самостоятельной работы

Объяснить, почему результаты работы программ, изображенных на рис. 2.41, 2.42 несколько отличаются. Нет ли тут ошибки?

Рис. 2.42

2.8.5. Взаимоблокировка

Серьезной проблемой многопоточных программ является возможность взаимной блокировки потоков. Взаимная блокировка может произойти в том случае, если каждый из двух потоков ждет освобождения ресурса, заблокированного другим потоком. Предположим, в одном потоке выполняются следующие строки:

```
lock (a)
{
    lock (b)
    {
        // Здесь что-то выполняется
    }
}
```

В то же время в другом потоке выполняются такие строки:

```
lock (b)
{
    lock (a)
    {
        // Здесь что-то выполняется
    }
}
```

Если процессор переключится с первого потока на второй после того, как первый поток заблокировал объект a, то второй поток заблокирует объект b и будет ожидать освобождения объекта a. После того как управление перейдет к первому потоку, он будет ожидать освобождения объекта b. В результате ни один поток не сможет продолжить выполнение.

В программе, листинг которой следует далее, создается два объекта Section и Section1, которые используют общие данные (Runs и Runs1) и два метода для работы с ними:

```
using System;
using System.Threading;
namespace ConsoleApplication10
{
    class SectionDeadlock
    {
        // Объявление полей — общие строки
```

```
        private static string Runs; // строка цифр от 0 до 9
        private static string Runs1; // строка букв от a до к

        // Объявление конструктора
        SectionDeadlock (string A, string B)
        {
            Runs = A;
            Runs1 = B;
        }

        // Метод CountUp увеличивает значение в строке Runs от 0
до  10
        public void CountUp()
        {
            lock (Runs1)
            {
                for (int i = 1; i <= 10; i++)
                {
                    lock(Runs)
                    {
                            string Res = Runs.Substring(i-1, 1);
                            Console.WriteLine(Thread.CurrentThread.
                            Name + " "+ Res);
                            Thread.Sleep(1000);

                    }
                }
            }
        }
        // Метод Char выдает 10 букв в порядке, записанном в строке
Runs1
        public void Char()
        {
            lock (Runs)
            {
                for (int i = 1; i <= 10; i++)
                {
                    lock(Runs1)
                    {
```

```
                    string Res1 = Runs1.Substring(i-1, 1);
                    Console.WriteLine(Thread.CurrentThread.
                    Name + " "+ Res1);
                    Thread.Sleep(1000);
                }
            }
        }
    }
    static void Main()
    {
        // Создание экземпляра текущего класса
    SectionDeadlock Section = new SectionDeadlock("0123456789",
    "абвгдежзик");
    SectionDeadlock Section1 = new SectionDeadlock("0123456789",
    "абвгдежзик");
        // Создание и запуск двух потоков
        Thread t2 = new Thread(new ThreadStart(Section.
        CountUp));
        t2.Name = "t2";
        Thread t3 = new Thread(new ThreadStart(Section1.Char));
        t3.Name = "t3";
        t2.Start();
        t3.Start();

    }
  }
}
```

В этой программе поток t2 использует метод CountUp, в котором предварительно блокируется Runs1, и производится работа с Runs, а поток t3 использует метод Char, в котором блокируется Runs, и производится работа с Runs1. Первым получает управление поток t2. Он блокирует Runs1, печатает цифру 1 и засыпает. Управление получает процесс t3. Он блокирует Runs, но не печатает первую букву Runs1, поскольку поток t2 заблокировал Runs1. Управление переходит к потоку t2. Однако он не может выполняться, так как Runs заблокирован потоком t3. Налицо тупиковая ситуация. Результат работы программы показан на рис. 2.43.

Рис. 2.43

Задание для самостоятельной работы

На сайте авторов дан текст Windows-приложения, позволяющего моделировать взаимоблокировку потоков. При правильном написании программы потоки работают параллельно и взаимоблокировки не наступает. Окно приложения представлено на рис. 2.44.

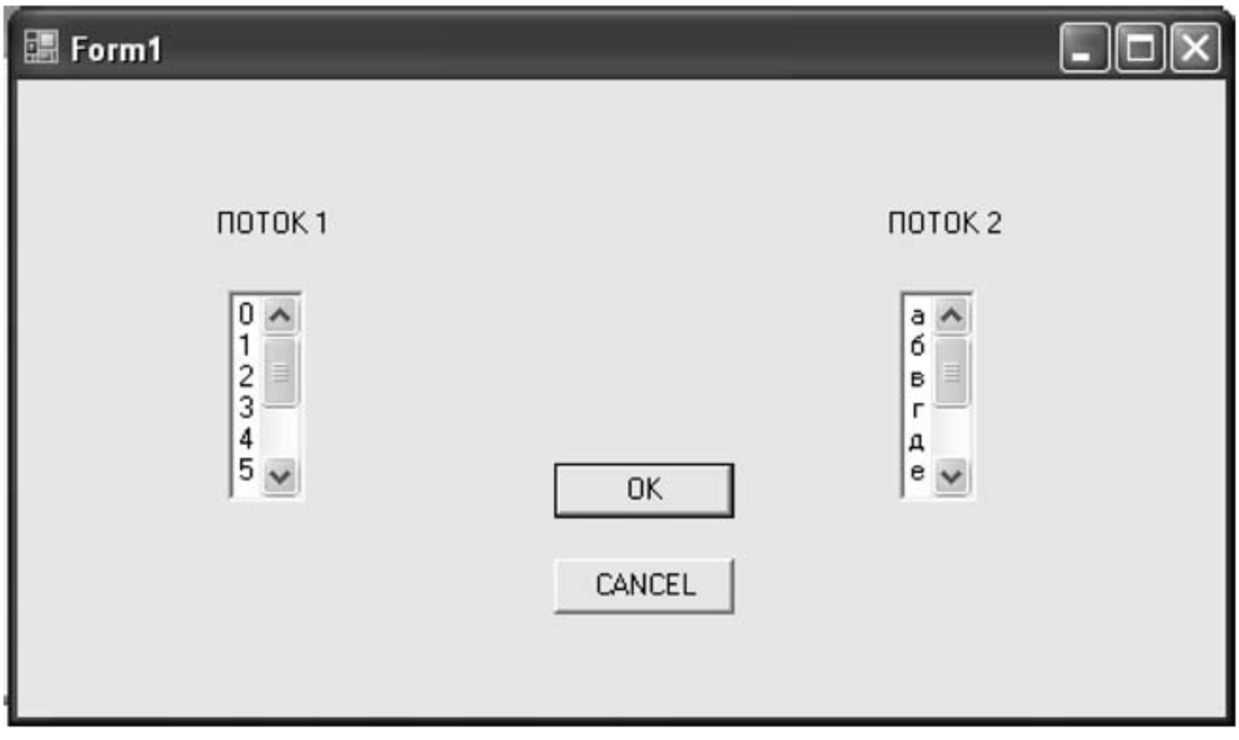


Рис. 2.44

При неправильном написании программы наступает взаимоблокировка. Окно приложения представлено на рис. 2.45.

Изменить программу таким образом, чтобы можно было проиллюстрировать обе возможные ситуации.

Рис. 2.45

2.9. Обнаружение взаимоблокировок

2.9.1. Неразделяемые ресурсы

Рассмотрим простейший вариант для случая, когда в системе существует только один ресурс каждого типа [16]. Например, пусть система из семи процессов (A, B, C, D, E, F, G) и шести ресурсов (R, S, T, V, W, U) в некоторый момент соответствует следующему списку:

- процесс A занимает ресурс R и хочет получить ресурс S;
- процесс B ничего не использует, но хочет получить ресурс T;
- процесс C ничего не использует, но хочет получить ресурс S;
- процесс D занимает ресурс U и хочет получить ресурсы S и T;
- процесс E занимает ресурс T и хочет получить ресурс V;
- процесс F занимает ресурс W и хочет получить ресурс S;
- процесс G занимает ресурс V и хочет получить ресурс U.

Вопрос: заблокирована ли эта система, и если да, то какие процессы в этом участвуют? Ответ можно получить, построив граф ресурсов и процессов.

Решение этой задачи можно получить, используя модель, программа которой приведена на сайте авторов. После запуска программы в открывшемся окне (рис. 2.46) нужно отметить ресурсы и процессы, которые используются в системе.

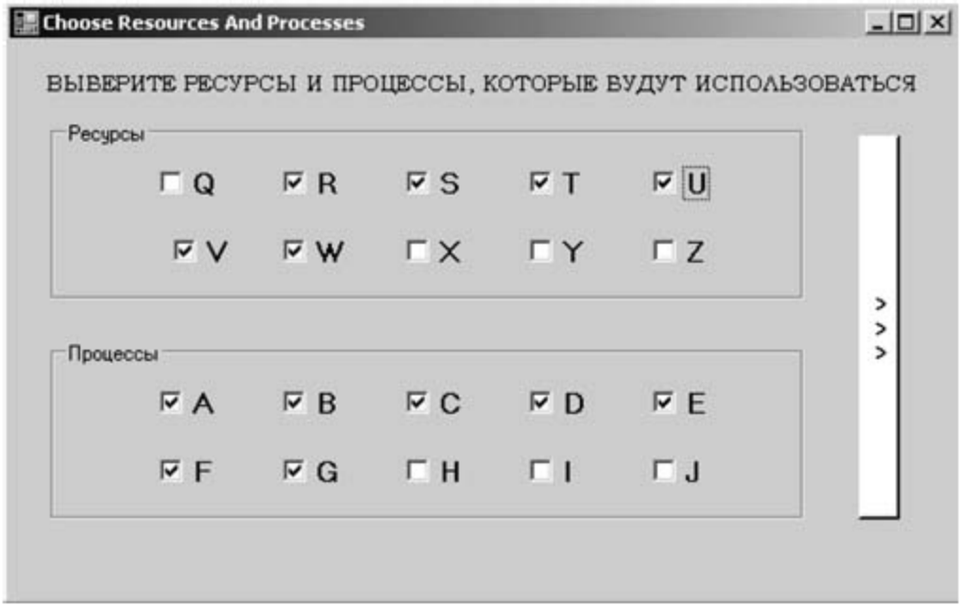


Рис. 2.46

Далее, нажав на клавишу со значком >, нужно распределить ресурсы по процессам в соответствии со списком, приведенным выше (рис. 2.47).

После распределения ресурсов на экране появляется окно (рис. 2.48), которое сообщает о наличии или отсутствии цикла в графе ресурсов и процессов. В этом окне можно построить граф ресурсов и процессов, позволяющий установить процессы, которые попали в тупиковую ситуацию (рис. 2.49).

Рис. 2.47

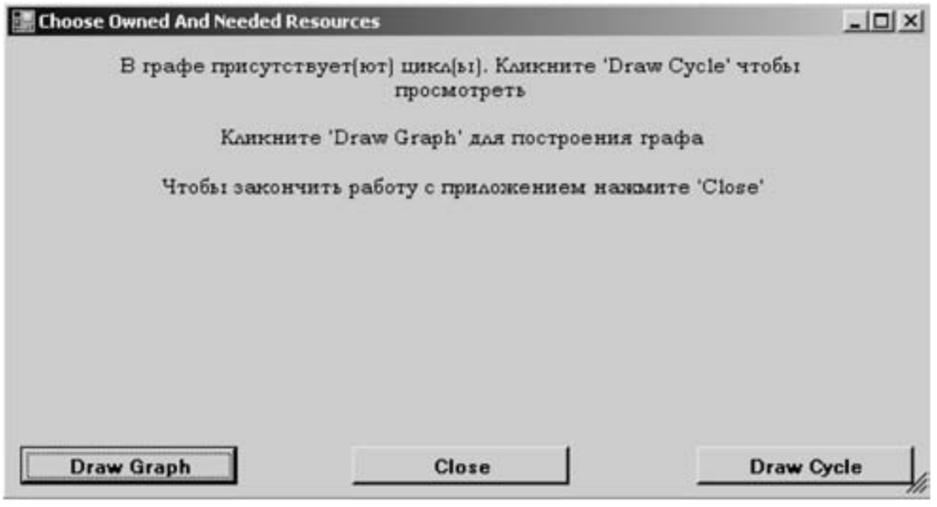


Рис. 2.48

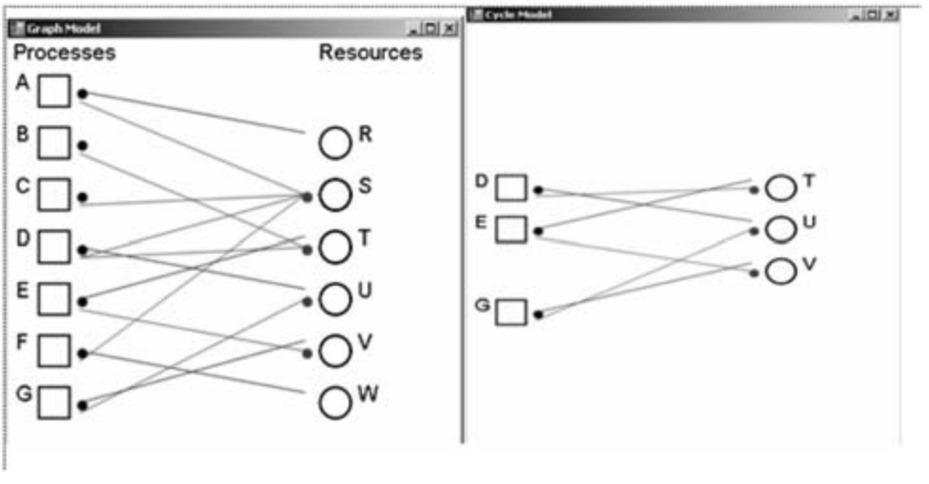


Рис. 2.49

2.9.2. Разделяемые ресурсы

Рассмотрим возможность обнаружения взаимоблокировок при наличии нескольких ресурсов каждого типа. Пусть имеется множество процессов $P=\{P_1, P_2, ..., P_n\}$, всего n процессов, и множество ресурсов $E=\{E_1, E_2, ..., E_m\}$, где m — число классов ресурсов. В любой момент времени некоторые из ресурсов могут быть заняты и соответственно недоступны. Пусть A — вектор доступных ресурсов $A=\{A_1, A_2, ..., A_m\}$. Очевидно, что $A_j \le E_j$, $j = 1, 2, ..., m$.

Введем в рассмотрение две матрицы:

$C=\{c_{i,j|}\ i = 1, 2, ..., n;\ j = 1, 2, ..., m\}$ — матрица текущего распределения ресурсов, где $c_{i,j}$ -количество ресурсов j-ого класса, которые занимает процесс P_i;

$R=\{r_{i,j|}\ i = 1, 2, ..., n; j = 1, 2, ..., m\}$ — матрица требуемых (запрашиваемых) ресурс $r_{i,j}$ —количество ресурсов j-го класса, которые хочет получить процесс $P_{i.}$

Справедливо m соотношений по ресурсам:

$$\sum_{i=1}^{n} C_{ij} + A_j = E_j, j= 1, 2, ..., m.$$

Алгоритм обнаружения взаимоблокировок основан на сравнении векторов доступных и требуемых ресурсов. В исходом состоянии все процессы не маркированы (не отмечены). По мере реализации алгоритма на процессы будет ставиться отметка, служащая признаком того, что они могут закончить свою работу и, следовательно, не находятся в тупике. После завершения алгоритма любой немаркированный процесс находится в тупиковой ситуации.

Алгоритм обнаружения тупиков состоит из следующих шагов:

1) ищется процесс P_i, для которого i-я строка матрицы R меньше вектора A, т.е. $R_i \le A$, или $r_{j,I} < A_j$, $j = 1, 2, ..., m$;
2) если такой процесс найден, это означает, что он может завершиться, а следовательно освободить занятые ресурсы. Найденный процесс маркируется, и далее прибавляется i-я строка матрицы C к вектору A, т.е. $A_j = A_j + c_{i,j}$, $j = 1, 2, ..., m$. Возвращаемся к шагу 1;
3) если таких процессов не существует, работа алгоритма заканчивается. Немаркированные процессы попадают в тупик.

В учебных материалах авторов на сайте www.hse.ru дана Windows-программа, реализующая изложенный алгоритм (ограничения на число ресурсов — до 10 и число процессов — до 16 для наглядного отображения всех элементов алгоритма). После запуска программы на экране появляется окно (рис. 2.50), позволяющее заполнять матрицы распределенных и требуемых ресурсов по каждому из 16 возможных процессов. В нижней части окна задается количество процессов и ресурсов, а также отображается вектор свободных и занятых ресурсов. Результат решения показан на рис. 2.51.

Задания для самостоятельной работы

1. Изучить Windows-программу и использовать ее для решения задач обнаружения тупиковых ситуаций и заблокированных процессов в системах с единичными ресурсами по заданиям, приведенным в разделе «Задачи».

2. Изучить Windows-программу и использовать ее для решения задач обнаружения тупиковых ситуаций и заблокированных процессов в системах с множественными ресурсами по заданиям, приведенным в разделе «Задачи».

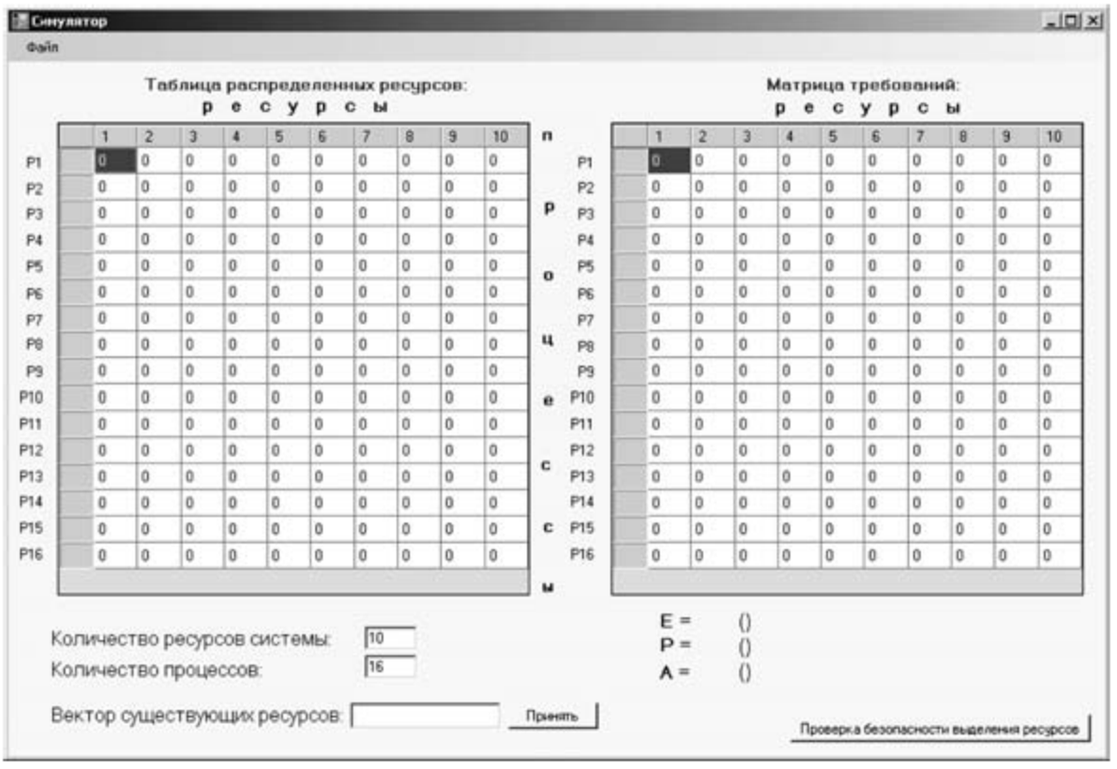


Рис. 2.50

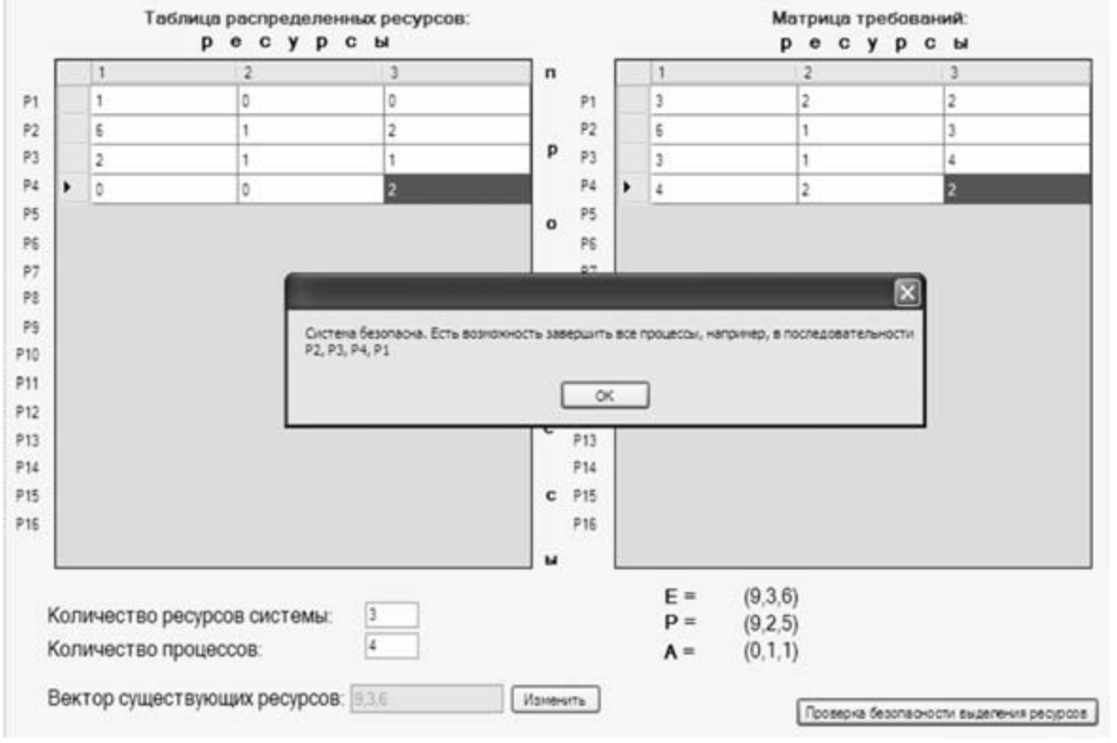


Рис. 2.51

2.10. Синхронизация потоков

Во многих приложениях возникает одна и та же проблема синхронизации, о которой часто говорят как о сценарии «один писатель — группа читателей» [11, 16]. Суть проблемы заключается в том, что произвольное число потоков пытается получить доступ к некоторому разделяемому ресурсу. Потокам-писателям нужно модифицировать данные, а потокам-читателям — прочитать эти данные. Синхронизация такого процесса необходима, поскольку должны соблюдаться следующие правила:

1) если один поток-писатель что-то пишет в область общих данных, то другие потоки-писатели этого делать не должны;
2) когда один поток-писатель что-то пишет в область общих данных, другие потоки (читатели) не могут ничего считывать оттуда;
3) если один поток-читатель считывает что-то из области общих данных, другие потоки (писатели) не должны туда ничего записывать;
4) когда один поток-читатель считывает что-то из области общих данных, другие потоки-читатели тоже могут это делать.

В учебных материалах авторов на сайте www.hse.ru приведена модель, с помощью которой можно исследовать рассматриваемую задачу. Пользователь вводит количество потоков читателей и писателей через окно формы (рис. 2.52) и нажимает кнопку Запуск. Начинается процесс моделирования случайных действий потоков-писателей и потоков-читателей с соблюдением указанных правил работы.

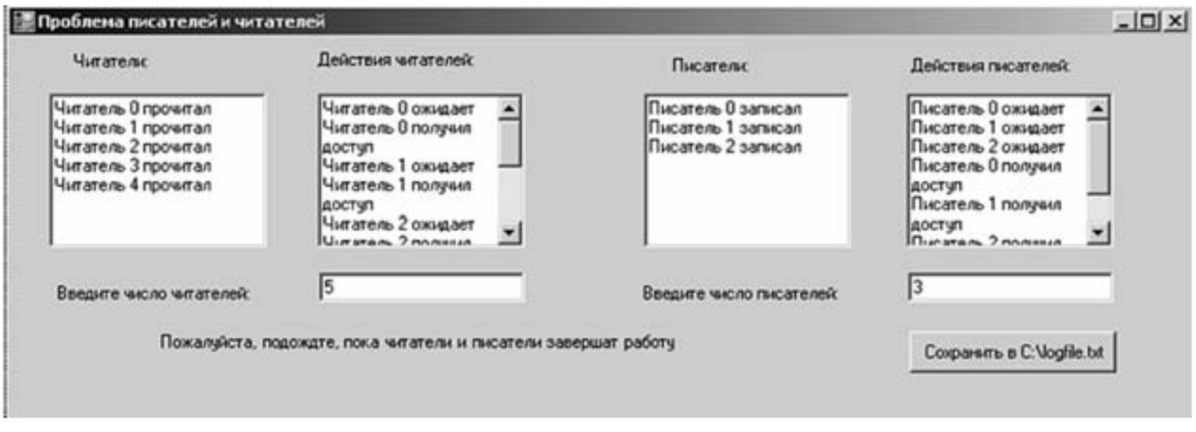


Рис. 2.52

Пользователь может видеть последовательность доступа читателей и писателей к файлу, с которым они работают. Для этого писатели и читатели регистрируют себя в нем.

Окно формы показывает пользователю четыре списка: первый последовательно формируется при доступе к данным читателями; второй — при входе читателей на пропускной пункт, это означает, что читатель объявил намерение получить доступ к данным, а также в этот список записывается информация о том, что читателю открыт доступ. Третий список последовательно формируется при доступе к данным писателями, четвертый — при входе писателей на пропускной пункт, это означает, что писатель объявил намерение получить доступ к данным, а также в этот список записывается информация о том, что писателю открыт доступ.

Если имеется тупиковая ситуация, то никакие списки не формируются и пользователь не может сохранить файл с информацией о последовательности доступа, поскольку ни один поток не может получить к нему доступ: каждый ожидает открытия доступа другим потоком. Если появляется возможность сохранить этот файл, то на экране отображается кнопка, предлагающая его сохранить и указывающая, где его найти. После сохранения данных можно очистить списки, просмотреть файл и вновь запустить модель.

Выход из программы осуществляется нажатием на кнопку выхода, заданную по умолчанию.

Нижняя часть окна Проблема писателей и читателей (на рис. 2.52 не показана) предназначена для вывода графика, иллюстрирующего количество читателей и писателей, стоящих в очереди и закончивших свою работу. Для вывода графика нужно нажать кнопку Сохранить в ... (место файла на диске). Для пояснения графических результатов следует нажать кнопку Условные обозначения (рис. 2.53). Сохраненный файл можно просмотреть в Блокноте.

С помощью программы можно анализировать ход выполнения потоков писателей и читателей и постановку их в очереди. При этом возможны следующие случаи развития процесса. Когда нет потоков-писателей, претендующих на ресурс, читатели оповещают любых писателей, что получают доступ к ресурсу и не задерживаются на семафорах. Можно наблюдать, что окна писателей пустые, а читатели, едва зарегистрировавшись в очередь, получают доступ к ресурсу.

Когда появляются писатели, они устанавливают семафор читателей, оповещая о своем приходе. Сами при этом ожидают на своем семафоре, так как им было объявлено, что читатели используют ресурс. Следующий читатель остановится в очереди на семафоре читателей, а следующие за ним читатели займут отдельную общую очередь, чтобы писатели могли пройти вперед них.

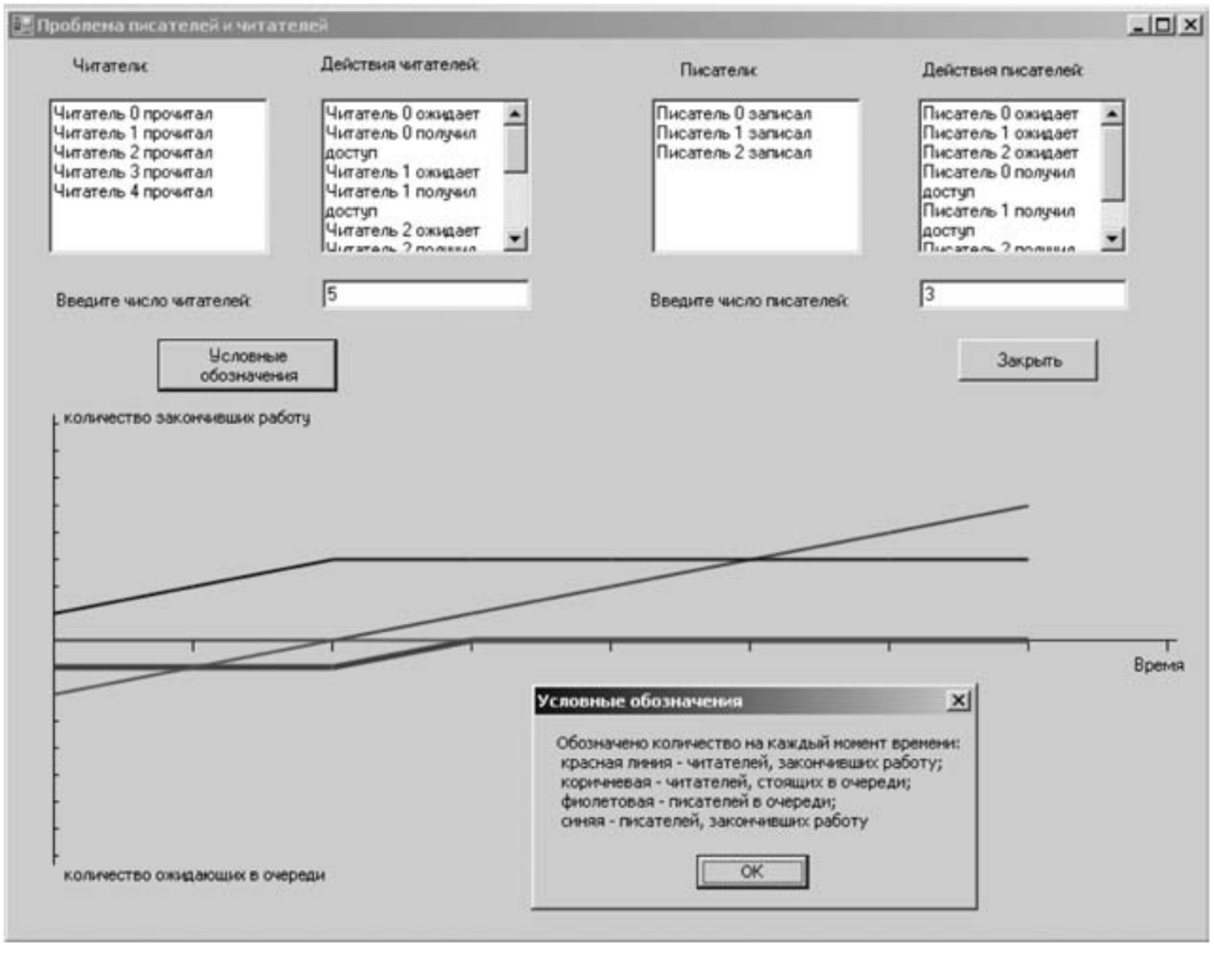


Рис. 2.53

Если читателей нет, то писатель устанавливает семафоры и для писателей, и для читателей. Остальные писатели ждут, пока он закончит работу на семафоре для писателей. Можно наблюдать пустые списки читателей и ожидающих в очередях писателей.

Задание для самостоятельной работы

1. Изучить Windows-программы и использовать их для моделирования задач синхронизации процессов.

Задачи

2.1. Требуется сравнить считывание файла через однопоточный и многопоточный файловые серверы. Получение запроса, его диспетчеризация и обработка занимают 15 мс при условии наличия данных в блочном кэше. В каждом третьем случае требуется обращение к диску, занимающее 75 мс, в течение которых поток находится в состоянии ожидания. Сколько запросов в 1 с обработает однопоточный сервер? А многопоточный?

2.2. Покажите, как можно реализовать считающие семафоры при помощи только бинарных семафоров и обычных машинных команд. Нарисуйте блок-схему алгоритма.

2.3. Одновременно запускаются два задания, каждому из которых нужно 10 мин работы процессора. Сколько времени потребуется для завершения их работы, если они работают последовательно? А сколько, если они работают параллельно? Предположим, ожидание ввода-вывода составляет 50%.

2.4. Три задачи A, B, C поступают в компьютерный центр практически одновременно. Ожидается, что время их выполнения составит 4, 2, и 7 мин соответственно. Требуется определить среднее время выполнения задач, считая, что время переключения между процессами (время смены контекста) 2 мс, а время кванта процессора — 20 мс. Планирование циклическое (каждой задаче достается справедливая доля процессорного времени).

2.5. Имеется многозадачный компьютер, в котором каждое задание имеет идентичные характеристики. В течение цикла вычисления одного задания T первую половину времени занимает ввод-вывод, а вторую половину — работа процессора. Для выполнения каждого задания требуется N циклов. Для планирования используется простой алгоритм циклического обслуживания, а ввод-вывод выполняется одновременно с работой процессора.

Определить значение реальное время, затрачиваемое на выполнение задания, если в компьютере одновременно выполняется четыре задания.

2.6. Измерения показали, что время выполнения среднестатистического процесса до блокировки ввода-вывода равно T. На переключение между процессами уходит время S, которое теряется впустую. Напишите формулу расчета эффективности для циклического планирования с квантом Q, принимающим следующие значения:

а) $S < Q < T$;

б) $Q = S$.

2.7. Запуска ожидают пять задач. Предполагаемое время их выполнения составляет 9, 6, 3, 5 и X. В каком порядке их следует запустить, чтобы минимизировать среднее время отклика? (Ответ должен зависеть от X).

2.8. Пять пакетных задач A, B, C, D, E поступают в однопроцессорный компьютер с интервалом в 1 мин. Ожидаемое время их выполнения составляет 10, 6, 2, 4 и 8 мин. Их установленные приоритеты — 3,

5, 2, 1 и 4, причем 5 — высший приоритет. Определите среднее оборотное время для алгоритма приоритетного планирования (запущенная задача работает до конца), пренебрегая временем, теряющимся при переключении между процессами.

2.9. Четыре пакетных задачи A, B, C, D поступают в двухпроцессорный компьютер практически одновременно. Ожидаемое время их выполнения составляет 10, 6, 2 и 8 мин. Их установленные приоритеты — 3, 2, 1 и 4, причем 4 — высший приоритет. Определите среднее оборотное время для алгоритма «первым пришел — первым обслужен» (запущенная задача работает до конца), пренебрегая временем, теряющемся при переключении между процессами.

2.10. В гибкую систему реального времени поступают четыре периодических сигнала с периодами 50, 100, 200, и 250 мс. На обработку каждого сигнала требуется 35, 20, 10 и X времени центрального процессора. Требуется определить максимальное значение X, при котором система остается поддающейся планированию.

2.11. В оперативной памяти центрального процессора может содержаться три программы. Эти программы простаивают в ожидании ввода-вывода одну треть времени. Какая часть времени процессора пропадает? Как изменится эта часть, если число программ удвоить?

2.12. В системе есть пять процессов и четыре ресурса, которые можно предоставить процессам. Текущее распределение ресурсов и максимальное их количество, необходимое процессам, приведено в табл. 2.1.

Таблица 2.1

Процесс	Предоставлено R_1 R_2 R_3 R_4	Максимальные требования R_1 R_2 R_3 R_4	Требуется R_1 R_2 R_3 R_4	Доступно R_1 R_2 R_3 R_4
A	0 0 1 2	0 0 1 2		2 1 0 0
B	2 0 0 0	2 7 5 0		
C	0 0 3 4	6 6 5 6		
D	2 3 5 4	4 3 5 6		
E	0 3 3 2	0 6 5 2		

Необходимо:

а) заполнить столбец «Требуется»;

б) определить, безопасно ли немедленное удовлетворение запроса (0, 1, 0, 0) процесса C.

ГЛАВА 3

УПРАВЛЕНИЕ ПАМЯТЬЮ

3.1. Общие сведения об использовании памяти

Диспетчер задач Windows позволяет просматривать общее использование памяти на вкладке Быстродействие (рис. 3.1). Здесь отображается информация в трех разделах: Выделение памяти, Физическая память и Память ядра.

В первом разделе содержится три статистических параметра виртуальной памяти:

1) параметр Всего — общий объем виртуальной памяти, используемой как приложениями, так и ОС;
2) параметр Предел — объем доступной виртуальной памяти;
3) параметр Пик — наибольший объем памяти, использованный в течение сессии с момента последней загрузки.

В разделе Физическая память содержатся параметры, несущие информацию о текущем состоянии физической памяти машины. Эта статистика не имеет никакого отношения к файлу подкачки, следовательно, может являться хорошим индикатором ситуаций, когда его увеличение не даст эффекта. Параметр Всего — объем памяти, определенный ОС на компьютере. Параметр Доступно отражает память, доступную для использования процессами. Эта величина не включает память, доступную приложениям за счет файла подкачки. Каждое приложение требует определенный объем физической памяти и не может использовать только ресурсы файла подкачки. Параметр Системный кэш сообщает объем, доступный кэш-памяти системы. Это объем физической памяти, не задействованный ОС для удовлетворения своих потребностей.

В разделе Память ядра отображается информация о потребностях компонентов ОС, обладающих наивысшим приоритетом. Эти компоненты обычно работают с сервисом низкого уровня, типа прямого доступа к жесткому диску. Параметры Память ядра отображают потребности ключевых служб ОС. Параметр Всего — объем виртуальной

памяти, необходимой ОС. Параметр Выгружаемая несет информацию об общем объеме памяти, использованной системой за счет файла подкачки. Параметр Невыгружаемая — объем физической памяти, потребляемой ОС.

Необходимо помнить, что эти параметры относятся лишь к привилегированным службам, а не ко всему сервису системы в целом. Многие компоненты ОС работают как приложения. В большинстве случаев параметры Память ядра должны оставаться без изменений, если не меняется что-либо в ядре ОС (например, устанавливается новое устройство в компьютер). Глобальные изменения в этом разделе обычно являются сигналом возможного возникновения проблем [6].

С помощью Диспетчера задач можно также узнать объемы памяти, используемые процессами. Для этого нужно перейти на вкладку Процессы, которая показывает список исполняемых процессов и занимаемую ими память (рис. 3.2), в том числе физическую память, пиковую (максимальное использование памяти) и виртуальную память.

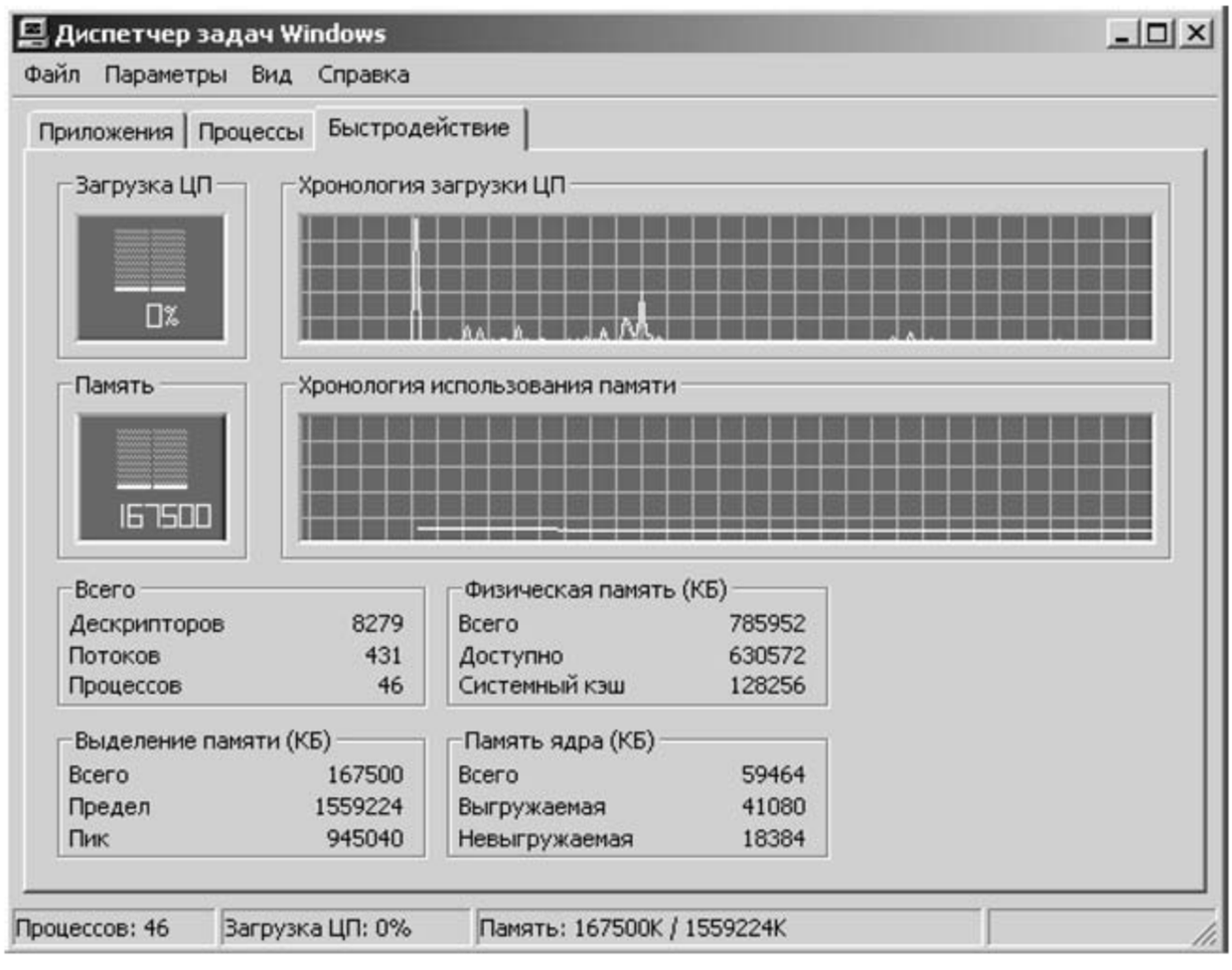


Рис. 3.1

Однако конкретное размещение процесса в виртуальной памяти с помощью Диспетчера задач узнать невозможно, нельзя также увидеть свободные, занятые страницы и блоки памяти, их размер и атрибуты защиты.

Информация, которую способен вывести диспетчер задач, не является полной [6]. В ряде случаев ее достаточно для оптимизации системы, но есть несколько ограничений, характерных для Диспетчера задач:

1) список процессов не полон. В окне Диспетчера задач представлены только процессы, зарегистрированные в Windows. В частности, в этот список не включаются драйверы устройств и некоторые системные службы;
2) требования к памяти отражают текущее состояние процесса. В списке отражены объемы памяти, занимаемые приложениями в текущий момент времени, а не их максимальные значения;
3) отсутствуют статистические данные. Поскольку в Диспетчере задач не выводятся временны́е характеристики, а только мгновенная картина потребления памяти, невозможно отследить ее изменение.

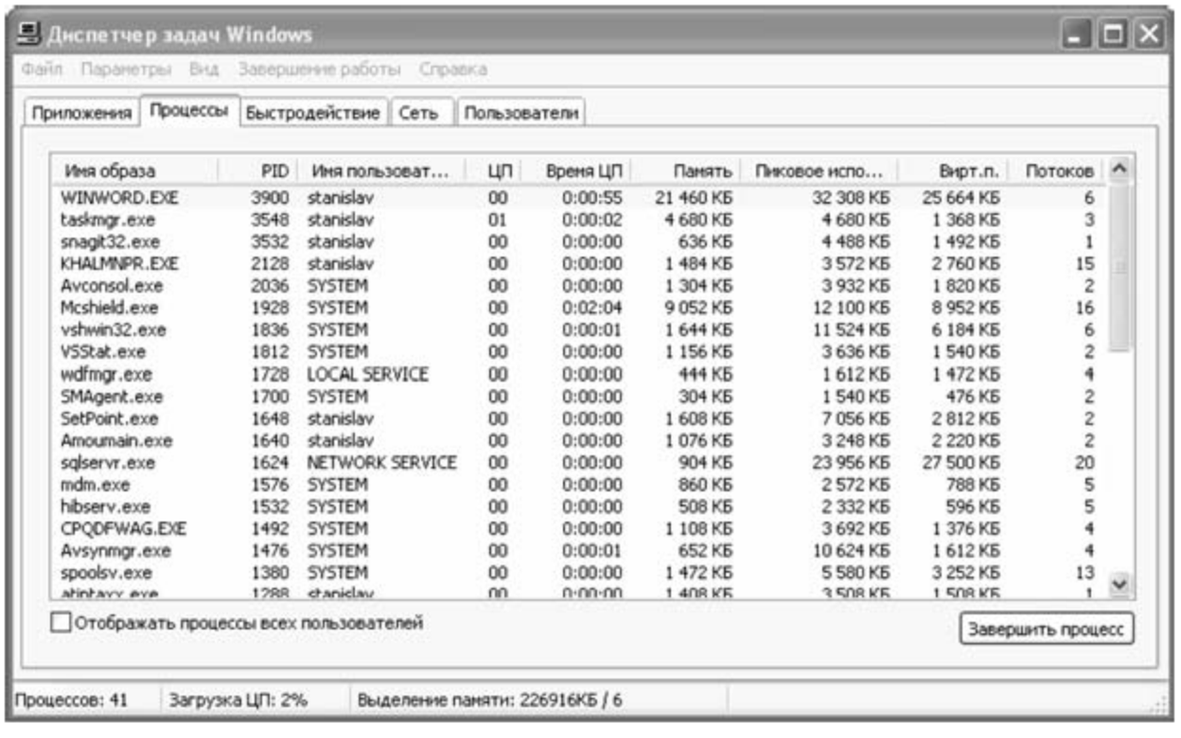


Рис. 3.2

Утилита TaskList предоставляет более обширную информацию по сравнению с Диспетчером задач, но пользоваться ею сложнее. Запускается утилита из окна командной строки (рис. 3.3).

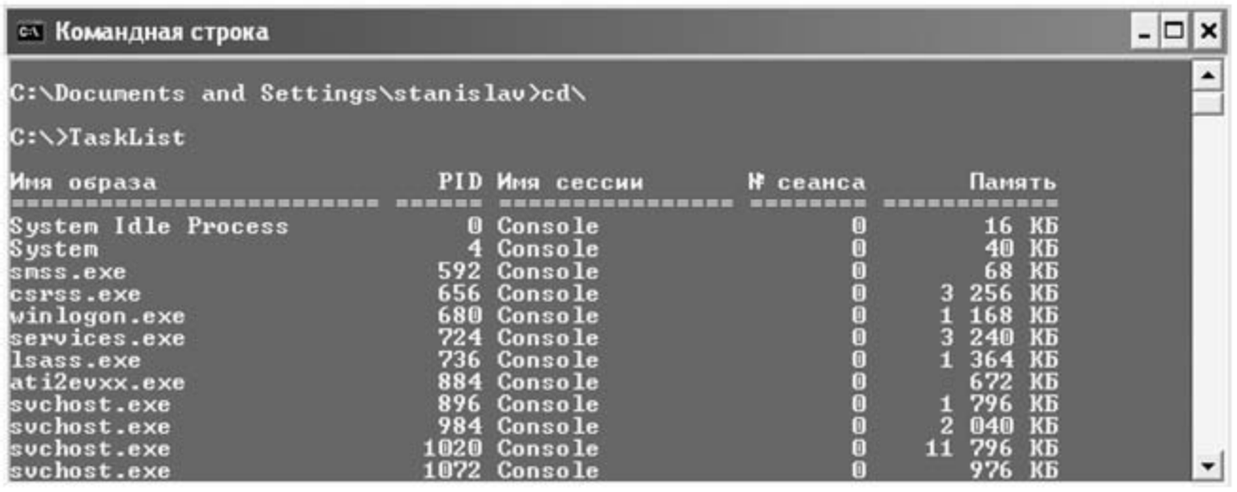


Рис. 3.3

Вызов утилиты с аргументами позволит получить более полезную информацию [6]. Например, параметр /M (модуль) позволит отобразить модули (обычно DLL), задействованные приложением. Параметр /FI обеспечит фильтрацию информации, выводимой утилитой, чтобы можно было видеть только интересующие записи и т.д. Получить информацию о параметрах утилиты TaskList можно обычным образом в окне командной строки (рис. 3.4).

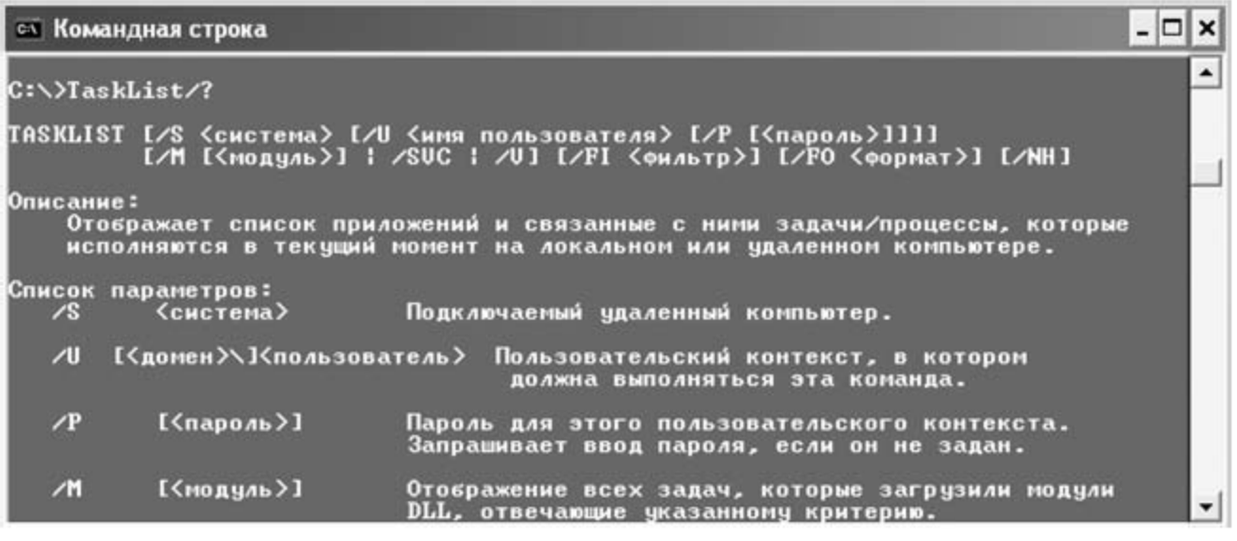


Рис. 3.4

Операционные системы Windows в Служебных программах содержат программу Сведения о системе, с помощью которой можно получить сведения об основных характеристиках организации памяти в компьютере (рис. 3.5).

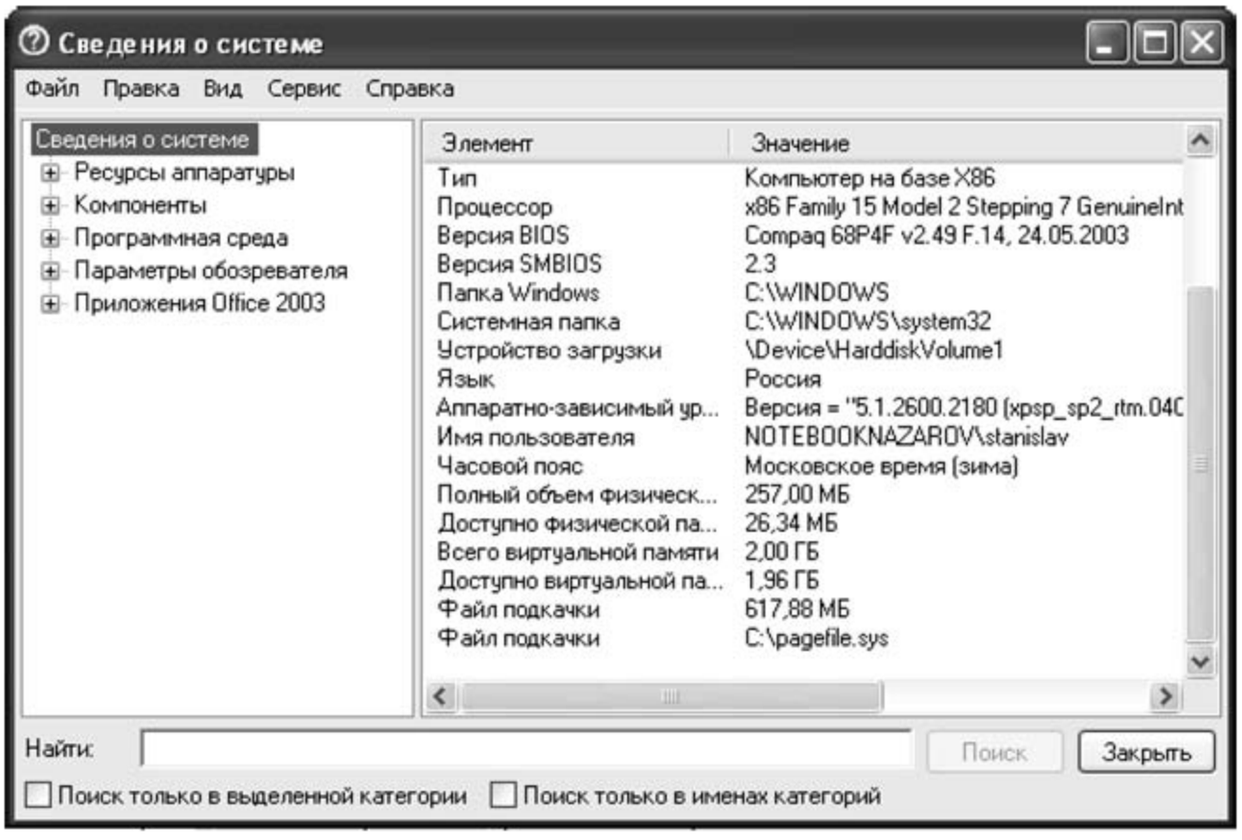


Рис. 3.5

В частности, здесь можно узнать полный объем установленной в компьютере физической памяти, общий объем виртуальной памяти и доступной (свободной) в данный момент времени виртуальной памяти, размещение и объем файла подкачки. Щелкнув по кнопке Ресурсы аппаратуры, а затем Память, можно получить сведения об использовании физической памяти аппаратными компонентами компьютера (рис 3.6).

3.2. Архитектура памяти в Windows

В операционной системе Windows 2000 реализована крайне сложная система виртуальной памяти [11, 16]. Предусмотрено множество функций Win32 для использования виртуальной памяти: часть исполняющей системы (менеджер памяти) плюс шесть выделенных потоков ядра для управления памятью. Каждый пользовательский процесс имеет собственное виртуальное адресное пространство размером в 4 Гбайт (адрес имеет 32 двоичных разряда). Конфигурация адресного пространства приведена на рис. 3.7. Нижние 2 Гбайт за вычетом примерно 256 Кбайт (системные данные — указатели и таймеры — ис-

пользуются совместно в режиме «только чтение») доступны для программы и данных процесса. Верхние 2 Гбайт защищенным образом отображаются на память ядра ОС. Страницы виртуального адресного пространства имеют фиксированный размер (4 Кбайт для процессора Pentium) и загружаются по требованию. Белым цветом на рис. 3.7 изображена область приватных данных пользовательского процесса (на рис. 3.7 его имя — A). Затененные области представляют память, которая совместно используется всеми процессами.

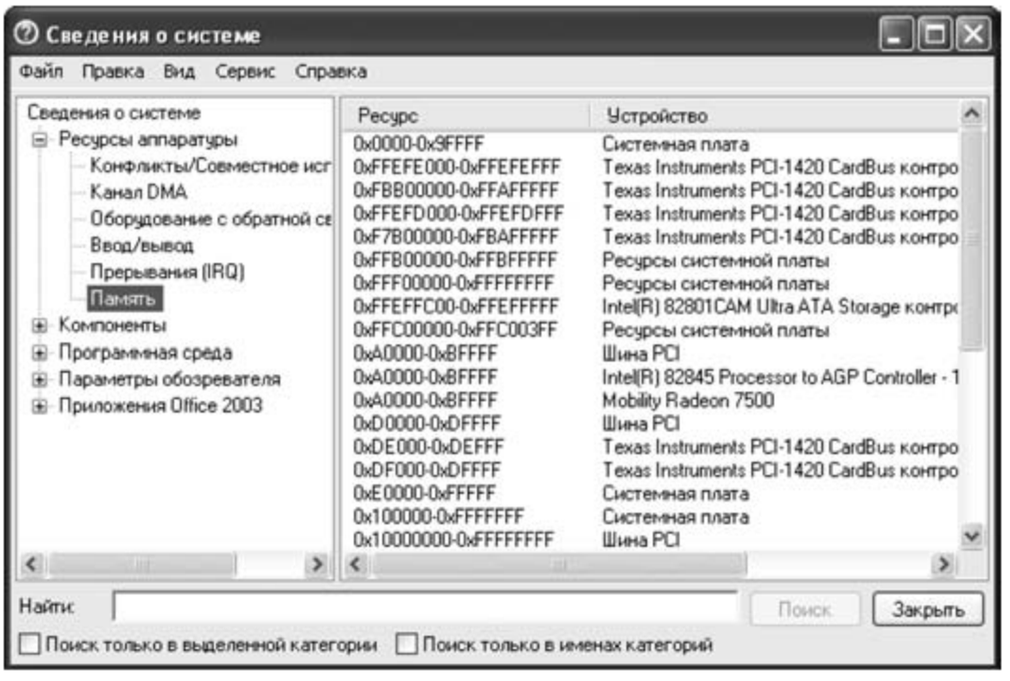


Рис. 3.6

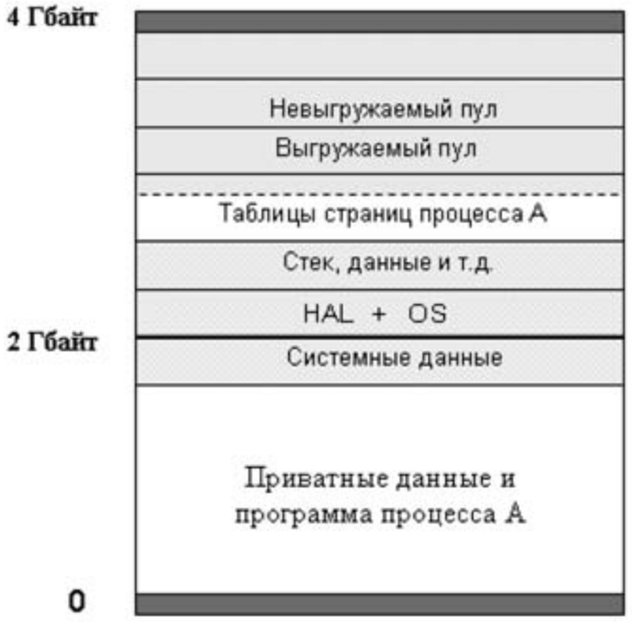


Рис. 3.7

Нижние и верхние 64 Кбайт каждого виртуального адресного пространства в обычном состоянии не отображаются на физическую память. Это сделано для облегчения перехвата программных ошибок (выявления недействительных указателей, имеющих значения 0 или –1).

Верхние 2 Гбайт виртуального адресного пространства предназначены для ОС, включая код ядра, драйверы устройств, кеш-буферы ввода-вывода, данные, выгружаемые и невыгружаемые пулы (используемые для объектов, создаваемых операционной системой) и т.д. Эта область используется совместно всеми процессами, кроме таблиц страниц, которые являются индивидуальными для каждого процесса. Верхняя часть адресного пространства запрещена для записи в режиме пользователя и, по большей части, в режиме чтения.

Причина, по которой потокам доступна эта область, заключается в том, что когда поток обращается к системному вызову, он переключается в режим ядра, но остается все тем же потоком. Если сделать всю ОС и все ее структуры данных (как и весь пользовательский процесс) видимыми в адресном пространстве потока, когда он переключается в режим ядра, то отпадает необходимость в изменении карты памяти или выгрузке кеша при входе в ядро. Все что нужно сделать, это переключиться на стек режима ядра. Платой за более быстрые системные вызовы при данном подходе является уменьшение приватного адресного пространства для каждого процесса.

Адресное пространство, выделенное процессу в момент создания, практически все свободно (не зарезервировано). Поэтому, чтобы воспользоваться какой-нибудь его частью, нужно выделить в нем определенные области, используя функцию Win32 API VirtualAlloc. Выделенные области всегда начинаются с 64-килобайтных страниц (предполагается, что это сведет к минимуму проблемы переноса системы на компьютеры будущего с большим размером страниц). Количество выделенного адресного пространства может быть меньше, чем 64 Кбайт, но оно должно состоять из целого числа страниц.

Для использования зарезервированного региона (области) адресного пространства ему нужно выделить физическую память и спроецировать ее на этот регион. Такая операция называется передачей физической памяти. Делается это с помощью той же функции VirtualAlloc. Передавая физическую память, нет необходимости отводить ее целому региону. Например, зарезервировав регион размером 64 Кбайт, можно передать физическую память только его второй и пятой страницам. Если физическая память, переданная зарезер-

вированному региону, больше не нужна, ее освобождают вызовом функции VirtualFree.

Рассматривая физическую память современных ОС, нужно помнить, что они «умеют» имитировать память за счет дискового пространства, создавая страничный файл (paging file). С точки зрения прикладной программы страничный файл просто увеличивает объем памяти, которой она может пользоваться. Следовательно, физическую память следует рассматривать как данные, хранимые в дисковом файле со страничной структурой. Поэтому когда приложение передает физическую память какому-нибудь региону адресного пространства, она на самом деле выделяется из файла, размещенного на жестком диске. Таким образом, размер страничного файла — главный фактор, определяющий количество физической памяти, доступной приложению. Реальный объем оперативной памяти имеет меньшее значение. Однако надо помнить о том, что процессор проецирует виртуальный адрес на физический только тогда, когда данные находятся в оперативной памяти.

Чем чаще системе приходится копировать страницы памяти в страничный файл и обратно, тем больше нагрузка на жесткий диск и тем медленнее работает ОС. Получается так, что система будет тратить все свое время на подкачку страниц вместо выполнения программы. Поэтому, увеличив объем оперативной памяти, можно снизить частоту обращения к диску и тем самым увеличить общую производительность системы.

Не следует думать, что страничный файл сильно увеличивается при одновременном выполнении нескольких программ за счет того, что система при каждом запуске приложения резервирует регионы адресного пространства, передает им физическую память, а затем копирует код и данные из файла программы (расположенного на диске) в физическую память, переданную из страничного файла. Операционная система действует не так, иначе на загрузку и подготовку программы к запуску уходило бы слишком много времени.

На самом деле при запуске приложения система открывает его исполняемый файл и определяет объем кода и данных. Затем резервирует регион адресного пространства и помечает, что физическая память, связанная с этим регионом, — сам EXE-файл, т.е. вместо выделения какого-то пространства из страничного файла система использует образ EXE-файла как зарезервированный регион адресного пространства программы. Благодаря этому приложение загружается очень быстро, а размер страничного файла удается существенно уменьшить.

Образ исполняемого файла (т.е. EXE- или DLL-файл), размещенный на диске и применяемый как физическая память для того или иного региона адресного пространства, называется проецируемым в память файлом (memory-mapped file). При загрузке EXE- или DLL-файлов система автоматически резервирует регион адресного пространства и проецирует на него образ файла. Помимо этого система позволяет проецировать на регион адресного пространства и файлы данных.

В заключение этого раздела рассмотрим типовую структуру адресного пространства прикладного процесса, выполняемого в операционной среде Windows 2000. Для этого можно использовать программу VMMap [11], предназначенную для детального просмотра карты виртуальной памяти (http://files.zipsites.ru/books/computers/windows/windows2000/richter4ru.zip). Она просматривает свое адресное пространство и показывает содержащиеся в нем регионы и блоки, присутствующие в регионах. При ее запуске открывается окно, изображенное на рис. 3.8.

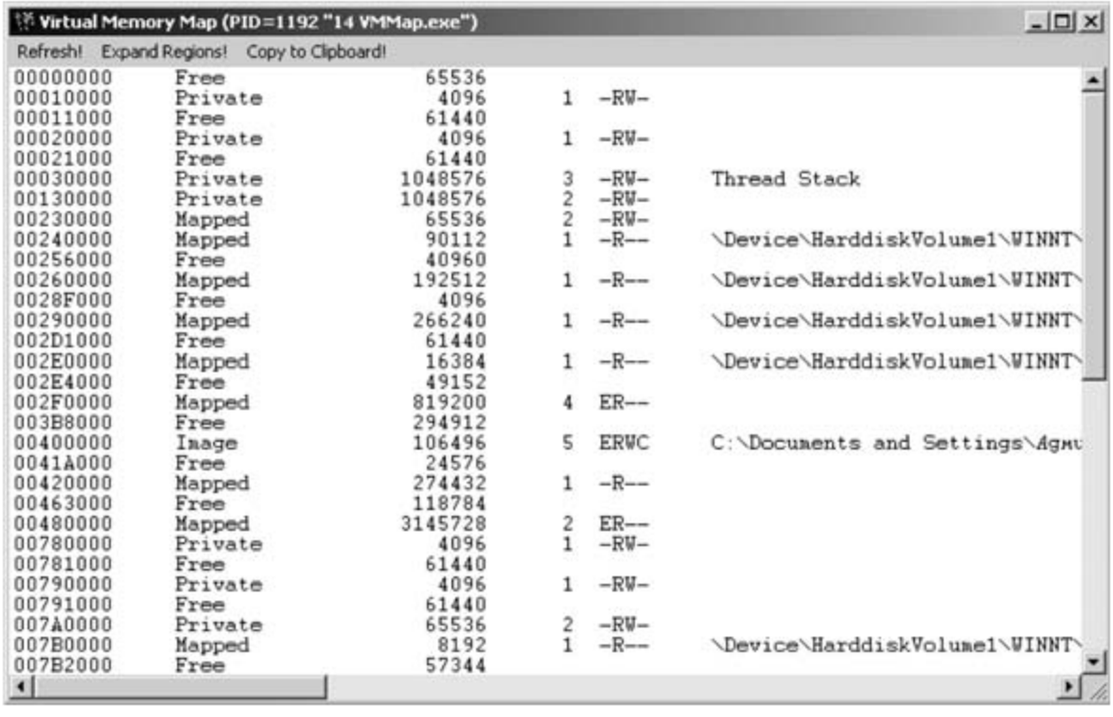


Рис. 3.8

В первом (левом) столбце отображается базовый адрес региона. Просмотр адресного пространства начинается с региона по адресу 0x00000000 и заканчивается последним регионом используемого адресного пространства, который начинается по адресу 0x7FFE0000. Все регионы непрерывны. Почти все эти базовые адреса начинаются со значений, кратных 64 Кбайт. Это связано с гранулярностью выделения памяти в адресном пространстве. Если какой-то участок памяти

начинается с адреса, не кратного 64 Кбайт, значит он выделен ОС для управления пользовательским процессом.

Во втором столбце указывается тип региона: Free (свободный), Private (закрытый), Image (образ) и Mapped (проецируемый). Характеристика типов дана ниже:

Тип	Описание
Free	Этот диапазон виртуальных адресов не сопоставлен ни с каким типом физической памяти. Его адресное пространство не зарезервировано
Private	Этот диапазон виртуальных адресов сопоставлен со страничным файлом
Image	Этот диапазон виртуальных адресов изначально был сопоставлен с образом EXE- или DLL-файла, проецируемого в память, но теперь, возможно, нет. Например, при записи в глобальную переменную из образа модуля механизм поддержки «копирования при записи» выделяет соответствующую страницу памяти из страничного файла, а не исходного образа файла
Mapped	Этот диапазон виртуальных адресов изначально был сопоставлен с файлом данных, проецируемым в память, но теперь, возможно, нет. Например, файл данных мог быть спроецирован с использованием механизма поддержки «копирования при записи». Любые операции записи в этот файл приведут к тому, что соответствующие страницы памяти будут выделены из страничного файла, а не из исходного файла данных

В третьем столбце дается размер региона в байтах. В четвертом столбце фиксируется количество блоков (группа страниц) с одинаковыми атрибутами защиты, связанных с одним и тем же типом физической памяти. Для свободных регионов это значение всегда равно нулю, так как им не передается физическая память. Поэтому в четвертом столбце никаких данных для свободных регионов не приводится. Для занятых регионов это значение может колебаться в пределах от 1 до максимума (его вычисляют делением размера региона на размер страницы).

Пятый столбец отображает атрибуты защиты региона: E = execute (исполнение), R = read (чтение), W = write (запись), C = copy-on-write (копирование при записи). Последний столбец описывает содержимое региона (для свободных регионов оно пустое, а для закрытых — обычно пустое, так как у VMMap нет возможности выяснить, зачем приложение зарезервировало данный закрытый регион).

Для детализации карты памяти нужно нажать в меню программы Expand Regions! Тогда внутри каждого региона будут отображаться блоки. Значения Private, Mapped и Image говорят о том, что блок поддерживается физической памятью соответственно из страничного файла, файла данных, загруженного EXE- или DLL-модуля. Значения Free и Reserve говорят о том, что блок вообще не связан с физической памятью.

Задания для самостоятельной работы

1. Используя рассмотренные средства ОС, определить объем установленной физической памяти, объем виртуальной памяти, величину файла подкачки и его размещение в компьютере.
2. Определить, какие области физической памяти использует системная плата.
3. Запустить программу VMMap и просмотреть карту виртуальной памяти процесса на своем компьютере. Обратить внимание на свободные участки памяти на «концах» адресного пространства.

3.3. Исследование виртуальной памяти

В работе [11] рассмотрена программа SystemInfo (files.zipsites.ru/effectivnye_win32_pril/richter4ru/txt). Назначение программы заключается в выводе системной информации о процессоре, ее количестве и виртуальной памяти. Для этого программа вызывает функцию GetSystemInfo. После запуска программы на экране открывается окно, показанное на рис. 3.9.

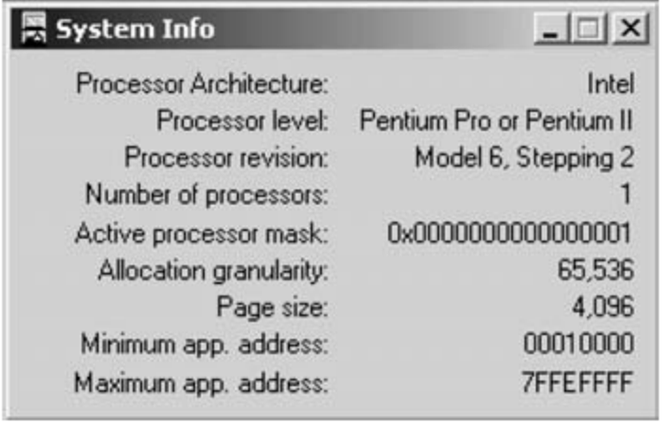


Рис. 3.9

По полученной информации можно определить основные характеристики процессора компьютера, число процессоров в компьютере, активные процессоры компьютера, дискретность (granularity) выделения памяти процессу, размер страницы и размер виртуального адресного пространства процесса (по значениям минимального и максимального адреса приложения).

Программа VMStat [11], в отличие от программы SysInfo, дает динамику изменения основных данных о физической и виртуальной памяти, обновляемых каждую секунду, что делает ее пригодной для мониторинга памяти в системе. Для этого программа вызывает функцию GlobalMemoryStatus. После запуска программы на экране открывается окно, показанное на рис. 3.10. Элемент Memory load позволяет оценить, насколько занята подсистема управления памятью. Это число может быть любым в диапазоне от 0 до 100 (на практике от значения этого элемента толку немного).

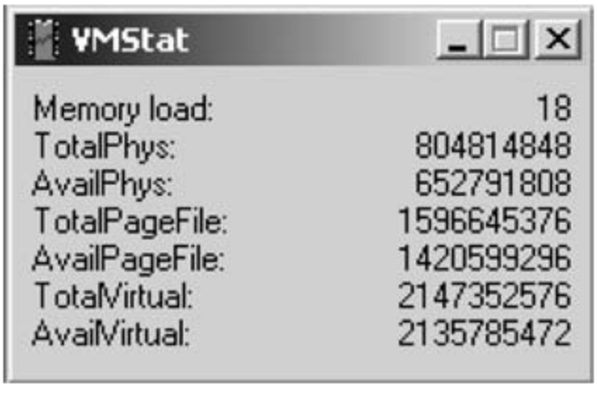


Рис. 3.10

Элемент TotalPhys отражает общий объем физической (оперативной) памяти в байтах. Однако реально память компьютера несколько больше (проверить экспериментально на своем компьютере). Причина, по которой GlobalMemoryStatus не сообщает полный объем памяти, заключается в том, что система при загрузке резервирует небольшой участок оперативной памяти, не доступный даже ядру. Этот участок никогда не сбрасывается на диск.

Элемент AvailPhys сообщает число байтов свободной физической памяти. Элемент TotalPageFile дает максимальное количество байтов, которое может содержаться в страничном файле (файлах) на жестком диске (дисках). Свободное число байтов в страничном файле, которое может быть передано любому процессу, показывает элемент AvailPageFile.

Элемент TotalVirtual отражает общее количество байтов, отведенных под закрытое адресное пространство процесса. Значение

2 147 352 576 ровно на 128 Кбайт меньше 2 Гбайт. Два раздела недоступного адресного пространства — от 0x00000000 до 0x0000FFFF и от 0x7FFF0000 до 0x7FFFFFFF — как раз и составляют эту разницу в 128 Кбайт.

Последний элемент, отображаемый на рис. 3.10, — AvailVirtual — единственный элемент структуры, специфичный для конкретного процесса, вызывающего GlobalMemoryStatus (остальные элементы относятся исключительно к самой системе и не зависят от того, какой именно процесс вызывает эту функцию). При подсчете значения AvailVirtual функция суммирует размеры всех свободных регионов в адресном пространстве вызывающего процесса. В данном случае его значение говорит о том, что в распоряжении программы VMStat имеется 2 135 785 472 байтов свободного адресного пространства. Вычтя из значения TotalVirtual величину AvailVirtual, получим 11 567 104 байтов — такой объем памяти VMStat зарезервировала в своем виртуальном адресном пространстве.

Задание для самостоятельной работы

1. Используя программу SystemInfo, определить объем виртуальной памяти, доступной процессу. Сравнить эти данные с результатом программы VMStat. Совпадают ли эти значения? Если нет, то почему?

3.4. Использование виртуальной памяти

Операционная система Windows реализует три механизма работы с памятью:

1) виртуальную память — для операций с большими массивами объектов или структур;
2) проецируемые в память файлы — для операций с большими потоками данных (обычно из файлов) и для совместного использования данных несколькими процессами на одном компьютере;
3) кучи — для работы с множеством малых объектов.

Функции, работающие с виртуальной памятью, позволяют напрямую резервировать регион адресного пространства, передавать ему физическую память из страничного файла и присваивать любые допустимые атрибуты защиты.

Программа VMAlloc [11] демонстрирует применение механизма виртуальной памяти для управления массивом структур. После ее запуска открывается диалоговое окно, показанное на рис. 3.11.

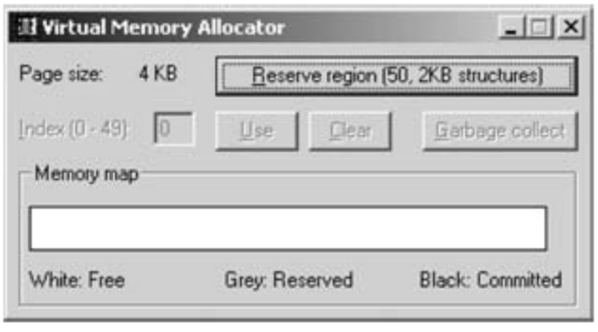


Рис. 3.11

Изначально для массива не резервируется никакого региона, все адресное пространство свободно, что отображается на карте памяти (Memory map) белым цветом. При нажатии кнопки Reserve region (50,2KB structures) программа VMAlloc вызовет функцию VirtualAlloc для резервирования региона, что сразу отразится на карте памяти (рис. 3.12), которая окрасится серым цветом. Активными становятся все кнопки и теперь можно ввести индекс и нажать кнопку Use. Например, введем индексы 5, 15 и 41 (рис. 3.13). При этом по адресу, где должен располагаться указанный элемент массива, предается физическая память. Это отображается в окне программы черным цветом на карте памяти.

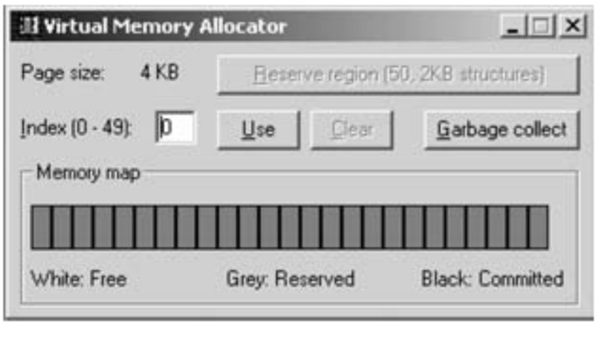


Рис. 3.12

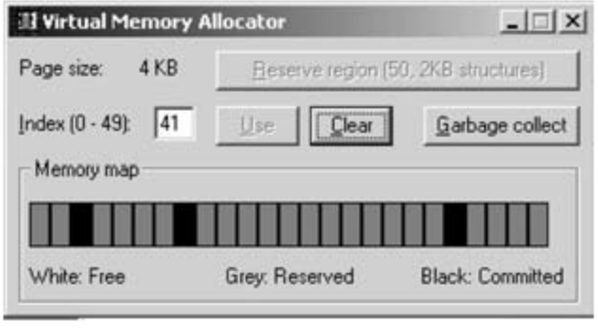


Рис. 3.13

Аналогично можно занять следующие элементы зарезервированной памяти. Любой элемент массива, помеченный как занятый, можно освободить щелчком по кнопке Clear. Однако это не приведет к возврату физической памяти, переданной под элемент массива. Дело в том, что каждая страница содержит несколько структур и освобождение одной структуры не влечет за собой освобождения других. Если бы память была возвращена, то пропали бы и данные, содержащиеся в остальных структурах.

Освобождение структуры приводит к тому, что ее элемент sInUse (см. листинг 15-1 в [11]) принимает значение False. Это нужно для того, чтобы функция сбора мусора могла вернуть не используемую больше физическую память. Делается это с помощью кнопки Garbage collect (Сборка мусора). Чтобы посмотреть, как это работает, очистим элемент массива с индексом 41. Карта памяти пока не изменится. Если щелкнуть по кнопке Сборка мусора, то карта памяти обновится, как показано на рис. 3.14.

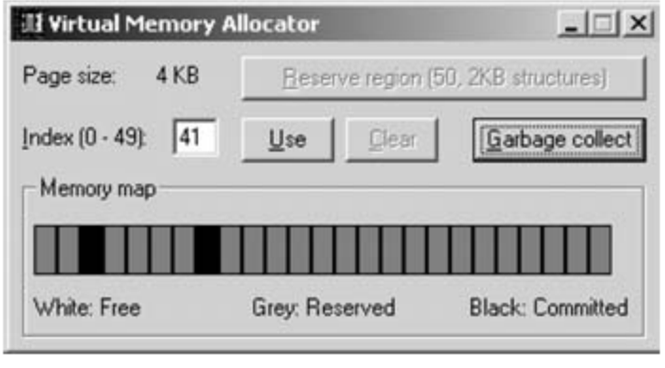


Рис. 3.14

3.5. Проецируемые в память файлы

Как и виртуальная память, проецируемые файлы позволяют резервировать регион адресного пространства и передавать ему физическую память. Различие между этими механизмами заключается в том, что в последнем случае физическая память не выделяется из страничного файла, а берется из файла, уже находящегося на диске. Как только файл спроецирован в память, к нему можно обращаться так, будто он целиком в нее загружен.

Проецируемые файлы применяются в следующих случаях [11]:

■ загрузка и выполнение EXE- и DLL-файлов. Это позволяет существенно экономить как на размере страничного файла, так и на времени, необходимом для подготовки приложения к выполнению;

■ организация доступа к файлу данных, размещенному на диске. При этом можно исключить операции файлового ввода-вывода и буферизацию его содержимого;

■ разделение данных между несколькими процессами, выполняемыми на одной машине.

Рассмотрим процесс загрузки и выполнения EXE- и DLL-файлов. При выполнении функции CreateProcess (создать процесс) операционная система действует следующим образом:

1) отыскивается EXE-файл, указанный в вызове функции, и, если он не найден, новый процесс не создается, а функция возвращает значение FALSE;
2) если файл найден, создается новый объект ядра — процесс;
3) создается адресное пространство нового процесса;
4) резервируется такой регион адресного пространства, чтобы в него поместился данный EXE-файл. Желательное расположение этого региона указывается внутри самого EXE-файла. По умолчанию базовый адрес EXE-файла — 0x00400000. При создании исполняемого файла приложения базовый адрес может быть изменен параметром компоновщика /BASE;
5) система отмечает, что физическая память, связанная с зарезервированным регионом, — EXE-файл — на диске, а не страничный файл.

Спроецировав EXE-файл на адресное пространство процесса, система обращается к разделу EXE-файла, содержащему список DLL, которые обеспечивают программе необходимые функции. После этого система, вызывая функцию LoadLibrary, поочередно загружает указанные DLL-модули (а при необходимости и дополнительные). Каждый раз когда для загрузки DLL вызывается LoadLibrary, ОС выполняет действия, аналогичные описанным ранее в п. 4 и 5.

Если система по какой-либо причине не свяжет EXE-файл с необходимым ему DLL-модулем, на экране появится соответствующее сообщение, а адресное пространство процесса и объект «процесс» будут освобождены. При этом функция CreateProcess вернет значение FALSE.

После увязки EXE- и DLL-файлов с адресным пространством процесса начинает исполняться стартовый код EXE-файла. Подкачку страниц, буферизацию и кэширование выполняет система. Например, если код в EXE-файле переходит к команде, не загруженной в оперативную память, возникает ошибка. Обнаружив ее, система перекачивает нужную страницу кода из образа файла на страницу оперативной

памяти. Затем отображает страницу оперативной памяти на должный участок адресного пространства процесса, позволяя потоку продолжить выполнение кода. Все эти операции скрыты от приложения и периодически повторяются при каждой попытке процесса обратиться к коду или данным, отсутствующим в оперативной памяти.

Операционная система позволяет проецировать на адресное пространство процесса и файл данных. Для этого нужно выполнить три операции:

1) создать или открыть объект ядра «файл», идентифицирующий дисковый файл, который предполагается использовать как проецируемый в память. Для этого используется функция CreateFile [11];
2) создать объект ядра «проекция файла» с помощью функции CreateFileMapping, чтобы сообщить системе размер файла и способ доступа к нему;
3) указать системе, как спроецировать в адресное пространство процесса объект «проекция файла» — целиком или частично. Это делает функция MapViewOfFile.

Закончив работу с проецируемым в память файлом, следует выполнить тоже три операции:

1) сообщить системе об отмене проецирования на адресное пространство процесса объекта ядра «проекция файла». Для этого предусмотрена функция UnmapViewOfFile;
2) закрыть объект «проекция файла»;
3) закрыть объект файл. Последние два действия осуществляются с помощью функции CloseHandle.

Остановимся на возможности разделения данных между несколькими процессами, выполняемыми на одной машине. Создавать файл на диске и хранить там данные только с этой целью неудобно. Поэтому в Windows предусмотрена возможность проецирования файлов непосредственно на физическую память из страничного файла, а не из специально создаваемого дискового файла. Этот способ проще стандартного, основанного на создании дискового файла, проецируемого в память.

Создав объект «проекция файла» и спроецировав его представление на адресное пространство своего процесса, его можно использовать так же, как и любой другой регион памяти. Для того чтобы данные стали доступны другим процессам, нужно вызвать функцию CreateFileMapping и передать в параметре pszName строку с нулевым символом в конце. Тогда посторонние процессы, если им понадо-

бится доступ к этому же файлу, смогут вызвать CreateFileMapping или OpenFileMapping и передать ей то же имя.

Когда необходимость в доступе к объекту «проекция файла» отпадет, процесс должен вызвать функцию CloseHandle. Как только все описатели объекта будут закрыты, система освободит память, переданную из страничного файла. Программа MMFShare, приведенная в [11], иллюстрирует, как происходит обмен данными между двумя и более процессами с помощью файлов, проецируемых в память. Для проведения эксперимента нужно запустить минимум две копии программы (рис. 3.15). Каждый экземпляр программы создаст свое диалоговое окно.

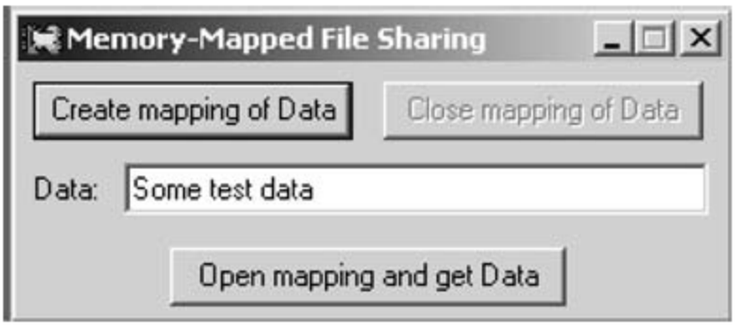


Рис. 3.15

Чтобы переслать данные из одной копии программы в другую, нужно набрать какой-нибудь текст в поле Data, а затем щелкнуть кнопку Create mapping of Data. Программа вызовет функцию CreateFileMapping, чтобы создать объект «проекция файла» размером 4 Кбайт и присвоить ему имя MMFSharedData (ресурсы выделяются объекту из страничного файла). Увидев, что объект с таким именем существует, программа выдаст сообщение, что не может создать объект. А если такого объекта нет, спроецирует представление файла на адресное пространство процесса и скопирует данные из поля Data в проецируемый файл.

Далее программа прекратит проецировать представление файла, отключит кнопку Create mapping of Data и активизирует кнопку Close Mapping of Data. На этот момент проецируемый в память файл с именем MMFSharedDATA будет находиться где-то в системе, и никакие процессы не проецируют представление на данные, содержащиеся в файле.

Если теперь перейти в другую копию программы и там щелкнуть кнопку Open mapping and get Data, то программа попытается найти объект «проекция файла» через функцию OpenFileMapping. Если ей не удастся найти объект с таким именем, она выдаст соответствую-

щее сообщение. В ином случае она спроецирует представление объекта на адресное пространство своего процесса и скопирует данные из проецируемого файла в поле Data. Таким образом, данные пересланы из одного процесса в другой.

Кнопка Close Mapping Of Data служит для закрытия объекта «проекция файла», что приводит к освобождению физической памяти, занимаемой им в страничном файле. Если же объект «проекция файла» не существует, никакой другой экземпляр программы MMFShare не сможет открыть этот объект и получить от него данные. Кроме того, если один экземпляр программы создал объект «проекция файла», то остальным повторить его создание и тем самым перезаписать данные, содержащиеся в файле, уже не удастся.

Задания для самостоятельной работы

1. Изучить возможности использования виртуальной памяти для операций с большими массивами объектов или структур на примере программы VMAlloc.
2. Изучить возможности применения проецируемых файлов, достоинства и недостатки этой технологии Windows.
3. Запустить два и более экземпляров программы MMFShare и провести эксперимент по обмену данными между процессами.

3.6. Изменение размера файла подкачки

Файл подкачки — область жесткого диска, используемая Windows для хранения данных оперативной памяти. Он создает иллюзию, что система располагает бóльшим объемом оперативной памяти, чем это есть на самом деле. Единой стратегии работы с файлом подкачки не существует. Многое определяется назначением и настройкой компьютера.

По умолчанию Windows удаляет файл подкачки после каждого сеанса работы и создает его в процессе загрузки ОС. Размер файла постоянно меняется по мере выполнения приложений и контролируется ОС. Обычно используется единственный файл подкачки, расположенный на том же диске, что и ОС. Такой подход не является лучшим, и более того, практически всегда плох. В этом случае возникает несколько проблем:

1) память может неожиданно оказаться исчерпанной, из-за того что приложение создало на жестком диске файл большого объема или операционная система без предупреждения увеличила потребности файла подкачки. Большой файл подкачки приводит к дефициту дискового пространства и к увеличению непроизводительных затрат на организацию страничного обмена;
2) файл подкачки фрагментируется, что приводит не только к медленному считыванию жесткого диска, но и к дополнительным перемещениям считывающей головки диска, а в итоге — к существенному снижению производительности;
3) файл подкачки фрагментируется сам по себе, и очень быстро, причем так, что одна и та же область памяти может оказаться в разных местах жесткого диска. В этом случае даже отдельные приложения не могут получить доступ к памяти без нескольких обращений к диску;
4) производительность системы снижается и в том случае, если ОС установлена не на самом быстром из жестких дисков, имеющихся в компьютере.

Наличие двух жестких дисков может дать значительное преимущество при настройке файла подкачки. Для максимально эффективного использования файла подкачки нужно так его настроить, чтобы он располагался на жестком диске в виде достаточно протяженных фрагментов (это уменьшает количество перемещений считывающей головки, радикально влияющих на производительность). Кроме того, файл подкачки необходимо периодически удалять, чтобы избежать его фрагментации.

Для установки размера файла подкачки нужно выполнить такую последовательность действий. Щелкнуть правой клавишей мыши на значке Мой компьютер и выбрать в контекстном меню строку Свойства. На экране появится окно Свойства системы (рис. 3.16). Перейти на вкладку Дополнительно (рис. 3.17) и нажать кнопку Параметры в рамке Быстродействие. В появившемся окне Параметры быстродействия (рис. 3.18) нажать кнопку Изменить. Предварительно следует выбрать принцип распределения времени процессора (для оптимизации работы программ, если это пользовательский компьютер, или служб, работающих в фоновом режиме, если это сервер). Кроме того, следует задать режим использования памяти: для пользовательского компьютера — оптимизировать работу программ, для сервера — работу системного кэша.

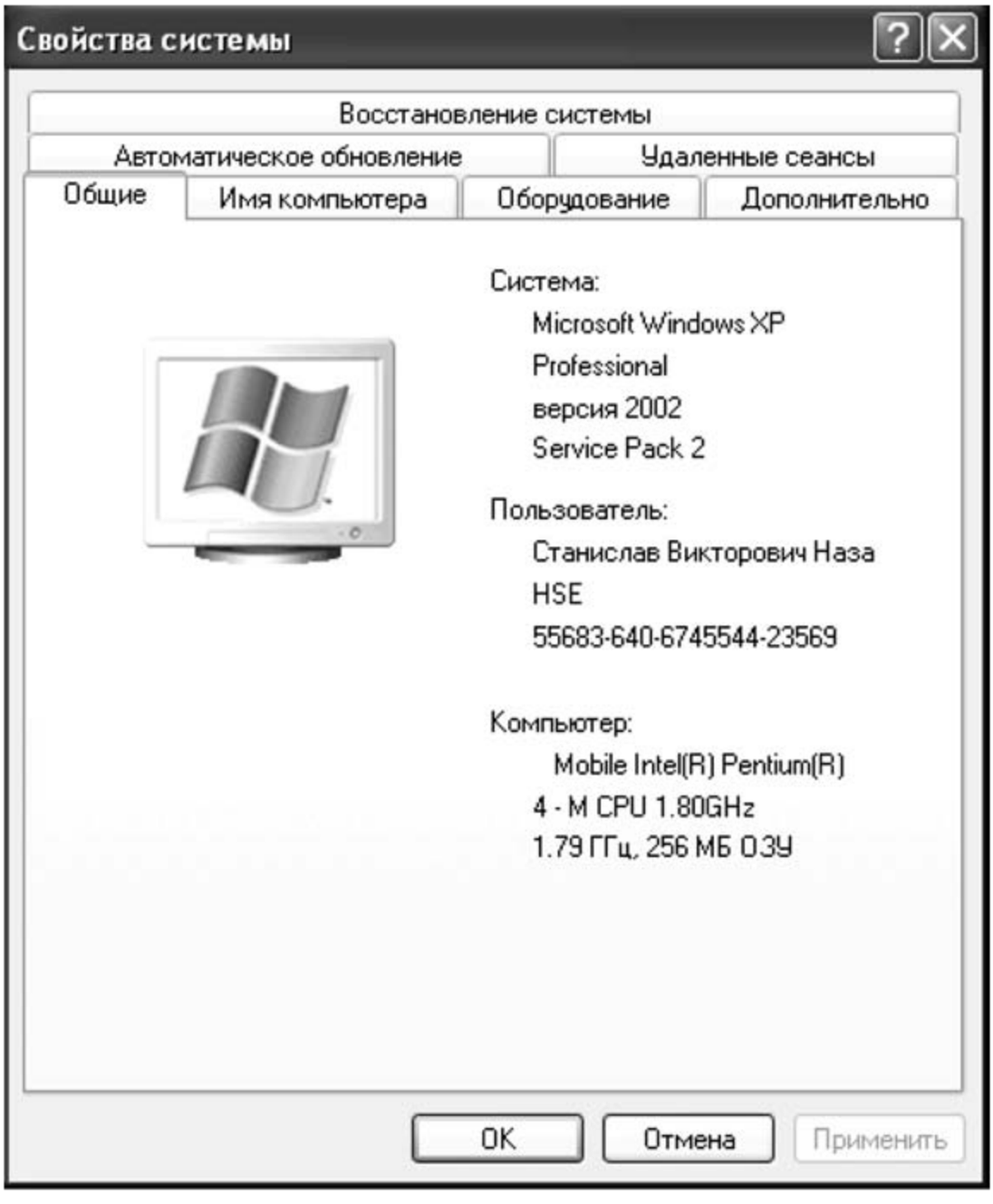


Рис. 3.16

Определение размера файла подкачки до сих пор вызывает многочисленные дискуссии. Основное правило заключается в том, что при небольшом объеме оперативной памяти файл подкачки должен быть достаточно большим. При большом объеме оперативной памяти (512 Мбайт и более) файл подкачки можно уменьшить. Можно вообще ликвидировать файл подкачки, выполнив определенную настройку реестра. Однако в этом случае объем оперативной памяти должен быть достаточно большим, лишь немногие системы обладают подобными ресурсами.

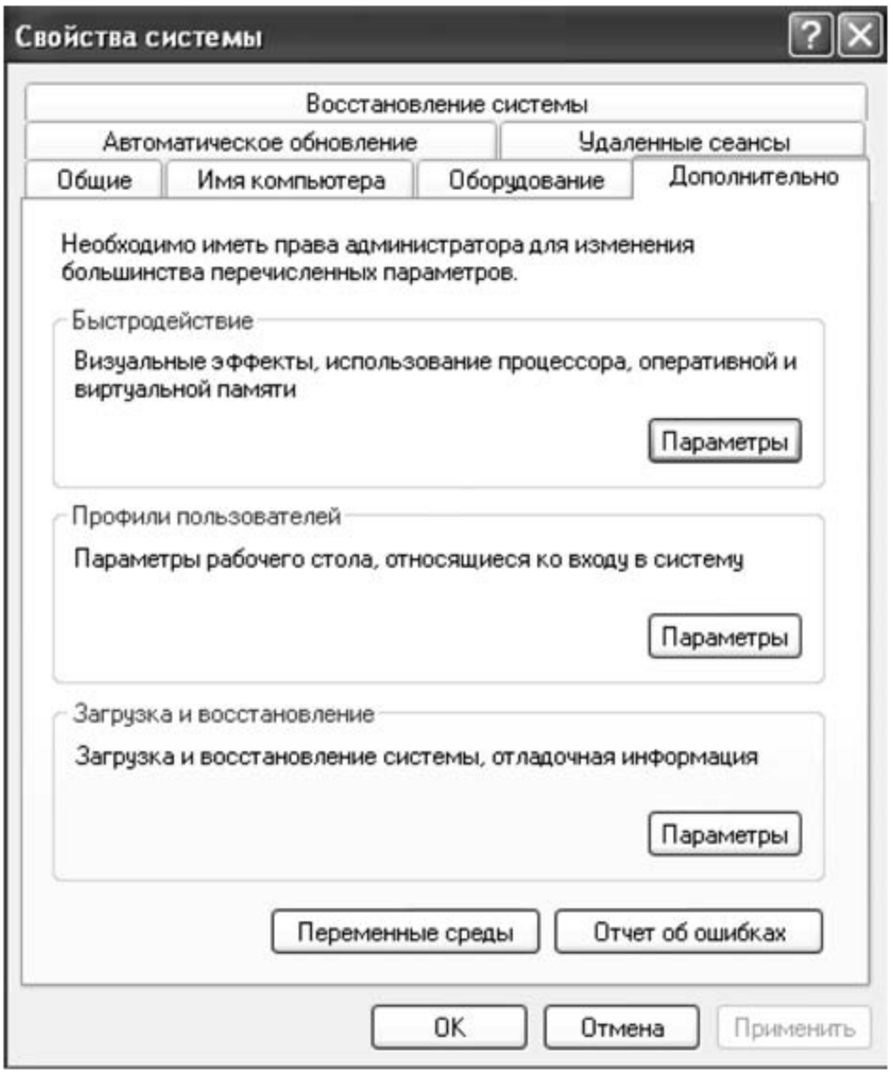


Рис. 3.17

По рекомендации, приведенной в [5, 6], можно установить исходный размер файла подкачки, равный размеру физической памяти, а максимальный размер — не более двух размеров физической памяти. После этого следует нажать кнопку Задать (рис. 3.19) и убедиться в том, что новое значение файла подкачки установлено. Далее щелкнуть на кнопке OK. Windows XP выведет сообщение о том, что данное изменение требует перезагрузки компьютера. Теперь нужно дважды нажать OK, чтобы закрыть окна сообщений и свойств системы и перезагрузить компьютер.

Следует иметь ввиду, что при первом создании файла подкачки жесткий диск, как правило, не готов к его размещению. Это обусловлено фрагментацией жесткого диска. Поэтому вначале нужно выполнить дефрагментацию диска и лишь затем создать файл подкачки, чтобы поместить его в единственную область диска. Последовательность действий может быть, например, такой:

■ если в компьютере имеется единственный жесткий диск, установить минимальный размер файла подкачки (2 Мбайт);

■ если имеются два жестких диска, переместить файл подкачки на более медленный диск;

■ провести дефрагментацию диска (во втором случае — быстрого). Для полной дефрагментации нужно выполнить несколько проходов;

■ присвоить файлу подкачки желаемый размер.

В результате работа с файлом подкачки станет максимально быстрой, а процессорная мощность и дисковое пространство будут использоваться эффективно.

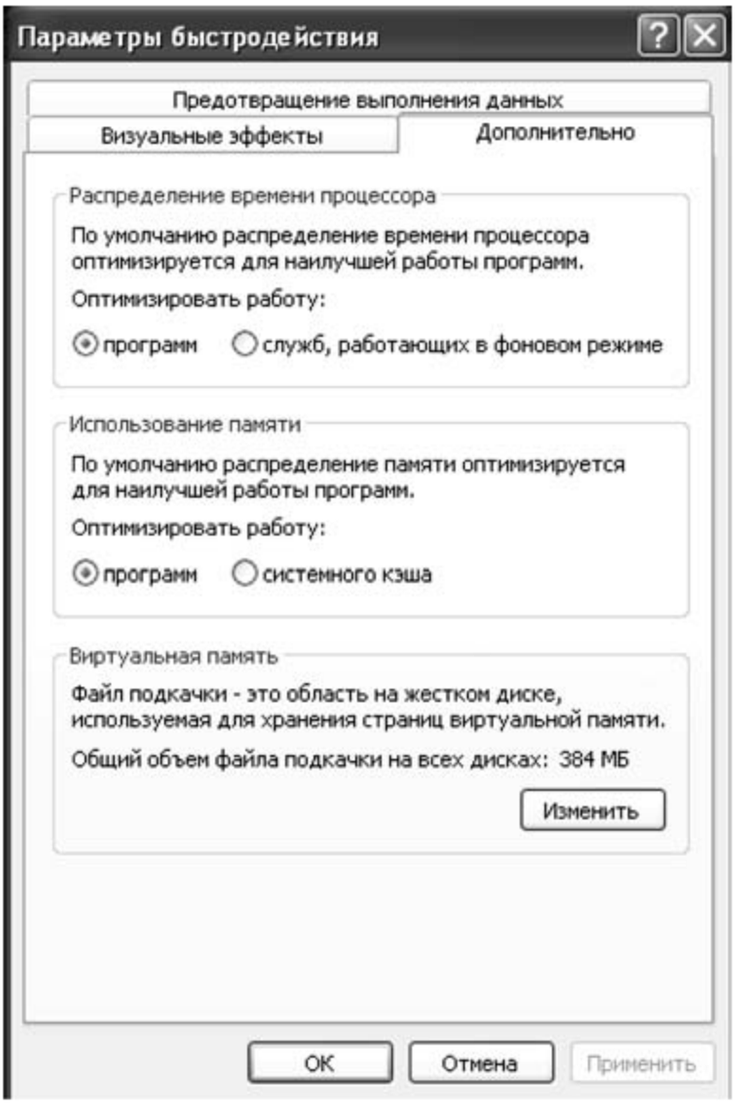


Рис. 3.18

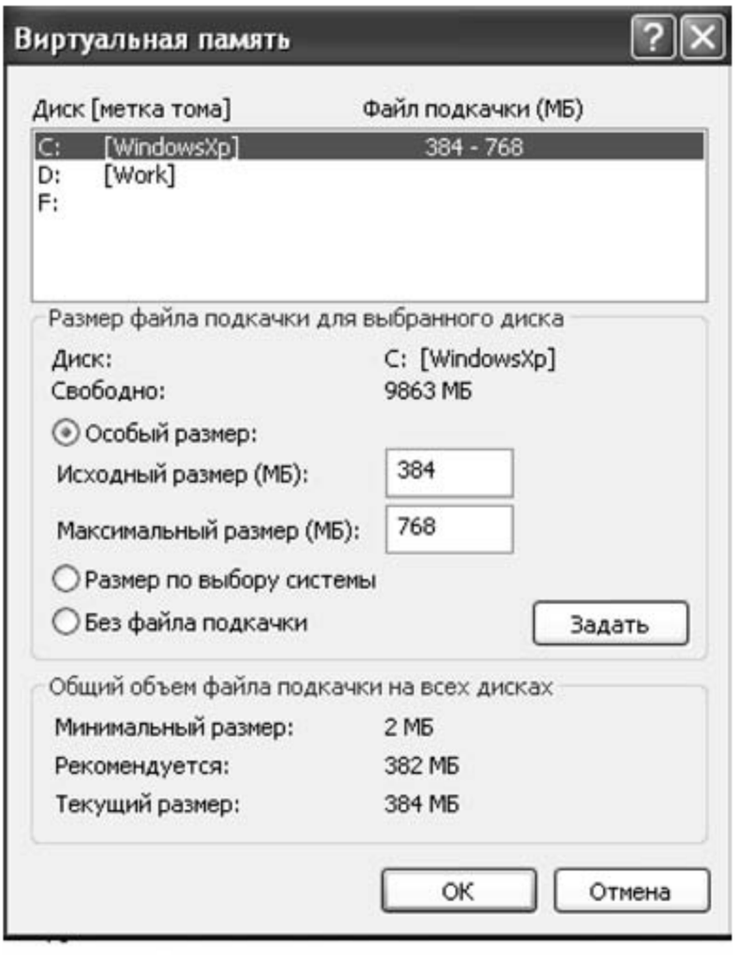


Рис. 3.19

Задания для самостоятельной работы

1. Определить объем оперативной памяти компьютера и рекомендуемый объем файла подкачки.
2. Как изложено ранее, провести дефрагментацию жесткого диска, на который предполагается поместить файл подкачки, установить его желаемое значение и перезагрузить компьютер.
3. Оценить эффект, полученный в результате изменения объема и размещения файла подкачки.

3.7. Исследование алгоритмов замены страниц

При страничном прерывании система должна выбрать страницу для удаления из памяти, чтобы освободить место для страницы, которую нужно перенести в память. Если удаляемая страница была изме-

нена за время своего присутствия в оперативной памяти, ее необходимо переписать на диск, чтобы обновить копию, хранящуюся там. Если же страница не была модифицирована, ее не нужно переписывать, поскольку на диске есть ее копия. В этом случае загружаемая страница записывается в память поверх выгружаемой страницы.

Главная проблема, связанная с такой заменой страниц, — выбор удаляемой страницы, чему посвящена масса работ. Наилучший алгоритм замещения страниц легко описать. Вот как он работает. Когда происходит страничное прерывание, в памяти находится некоторый набор страниц. К одной из этих страниц будет обращаться следующая команда процессора, к другим, возможно, не будет обращений в течение следующих 10, 100 или 1000 команд. Каждую страницу можно пометить количеством команд, которые будут выполняться перед первым обращением к этой странице.

Оптимальный страничный алгоритм определяет, что должна быть выгружена страница с наибольшей меткой. Это отодвинет на возможно максимальный срок страничное прерывание, которое вернет ее назад. Однако такой алгоритм невыполним, поскольку система не может определить, когда произойдет следующее обращение к каждой странице. Тем не менее, выполняя программу на модели и следя за всеми обращениями к страницам, оптимальную замену можно осуществить при втором запуске, используя информацию о ссылках на страницы, собранную во время первого запуска модели.

Рассмотрим одну из моделей исследования алгоритмов замены страниц (автор С. Максимов). В модели приведены четыре самых распространенных алгоритма замещения страниц: FIFO (первым прибыл — первым обслужен), вторая попытка, LRU (страница, не использовавшаяся дольше всего), NRU (не использовавшаяся в последнее время страница). Кроме этих четырех реальных алгоритмов, в модели присутствует пятый, так называемый оптимальный алгоритм. Этот алгоритм производит замещение страниц таким образом, что генерируется наименьшее возможное число прерываний. Он не может быть реализован в реальной системе, но результаты его исследования, т.е. число страничных прерываний, могут помочь в оценке эффективности других алгоритмов. Алгоритм, который показывает более близкие к оптимальному алгоритму результаты, можно считать более эффективным. Такой способ оценки, однако, не учитывает расходы на реализацию того или иного алгоритма. Будем считать, что расходы на реализацию всех алгоритмов равны — как расходы на память, так и расходы на вычислительные ресурсы. Это главное допущение модели.

Модель имеет следующие входные параметры:

■ число виртуальных страниц, т.е. число блоков, на которое поделена виртуальная память системы, $N_в$;

■ число физических страниц, которое содержится в оперативной памяти, $N_ф$;

■ строку обращений к страницам — основной входной параметр. Это модель последовательности обращений к страницам в реальной системе. Она представляет собой последовательность номеров виртуальных страниц, к которым обращаются процессы, а также тип обращения (запись или чтение) — S;

■ длину строки обращений, L_s;

■ для алгоритма NRU важным входным параметром является временной промежуток, по прошествии которого обнуляются биты чтения у страниц, $T_п$.

Выходными результатами модели, позволяющими оценить эффективность различных алгоритмов, являются:

■ число страничных прерываний, когда в процессе обращения к виртуальной странице она не была обнаружена в физической памяти $N_п = F (N_в, N_ф, S, T_п)$;

■ вероятность неудачного обращения к странице, т.е. вероятность обращения с возникновением страничного прерывания. Получить значение вероятности неудачного обращения можно, разделив число прерываний на длину строки обращений. Это более значимый показатель, поскольку он учитывает масштаб, т.е. длину строки обращений. $P_н = N_п / L_s$.

Программа модели приведена на сайте авторов. Ее основное назначение — обучение механизмам работы алгоритмов замещения страниц, а также помощь в исследовании этих алгоритмов. Исходя из этого была выбрана пошаговая форма реализации алгоритмов, т.е. пользователь может посмотреть, что происходит с системой на каждом шаге обращения к виртуальной странице.

Рассмотрим моделируемые алгоритмы. Алгоритм FIFO (First-In, First-Out — первым пришел — первым обслужен) является, пожалуй, самым простым алгоритмом из всех алгоритмов. Реализация этого алгоритма полностью оправдывает его название. В случае страничного прерывания из памяти удаляется та страница, которая первой попала в память. В случае если запрошенная страница существовала в памяти и прерывания не произошло, в памяти ничего не изменяется, и страница, которая стоит первой в списке загруженных страниц, будет удалена при следующем прерывании, даже если предыдущее обращение

было именно к ней. В этом и состоит главный недостаток алгоритма FIFO — он не учитывает того, насколько часто вызывалась страница и насколько давно был последний вызов. Иными словами, пользуясь алгоритмом FIFO, ОС может удалить из памяти важные и часто используемые страницы.

Алгоритм «вторая попытка» несколько совершенствует алгоритм FIFO. К каждой загруженной странице ОС приписывает бит обращений — бит R (от Referenced). Во время страничного прерывания (если у старшей страницы бит R установлен в 1, это означает, что к странице были обращения) страница не удаляется из памяти, а переносится в конец списка, при этом ее значение ее бита обращений устанавливается на 0. Таким образом, если у всех страниц, находящихся в памяти, бит R установлен на 1, то система прокрутит весь список страниц, изменит биты обращений всех страниц на 0 и затем удалит из памяти первую в списке страницу. Этот алгоритм значительно разумнее FIFO, но имеет свой существенный недостаток — затраты времени на передвижение страниц по списку.

Алгоритм LRU — страница, не использовавшаяся дольше всего. Он является неплохой аппроксимацией оптимального алгоритма. В его основе лежит учет того, что страницы, к которым происходило многократное обращение в последних командах, вероятно, будут часто использоваться и в дальнейшем. И наоборот, можно полагать, что страницы, к которым ранее не возникало обращений, не будут использоваться в течение долгого времени. Эта идея приводит к мысли: когда происходит страничное прерывание, нужно выгрузить страницу, которая не использовалась дольше всего.

Алгоритм NRU действует подобно алгоритму LRU, но только удаляет страницы не по мере необходимости, а заблаговременно. С каждым щелчком таймера ОС отслеживает страницы, которые были загружены в память достаточно давно, для того чтобы можно было освободить занимаемое ими место. Таким образом, практически не возникает случаев, когда во время страничного прерывания надо тратить время на то, чтобы переписывать удаляемую страницу на диск, если она была изменена, т.е. время теряется в одном месте, но выигрывается в другом. Как следствие, работа программы получается более сглаженной.

После запуска программы открывается главное окно с меню (рис. 3.20). Модель позволяет пользователю изучить в деталях каждый алгоритм в пошаговом режиме. Для этого используется инструмент Обход алгоритма. В меню Файл можно выбрать, обход какого именно алгоритма производить. На рисунке 3.21 показан обход алгоритма NRU.

Рис. 3.20

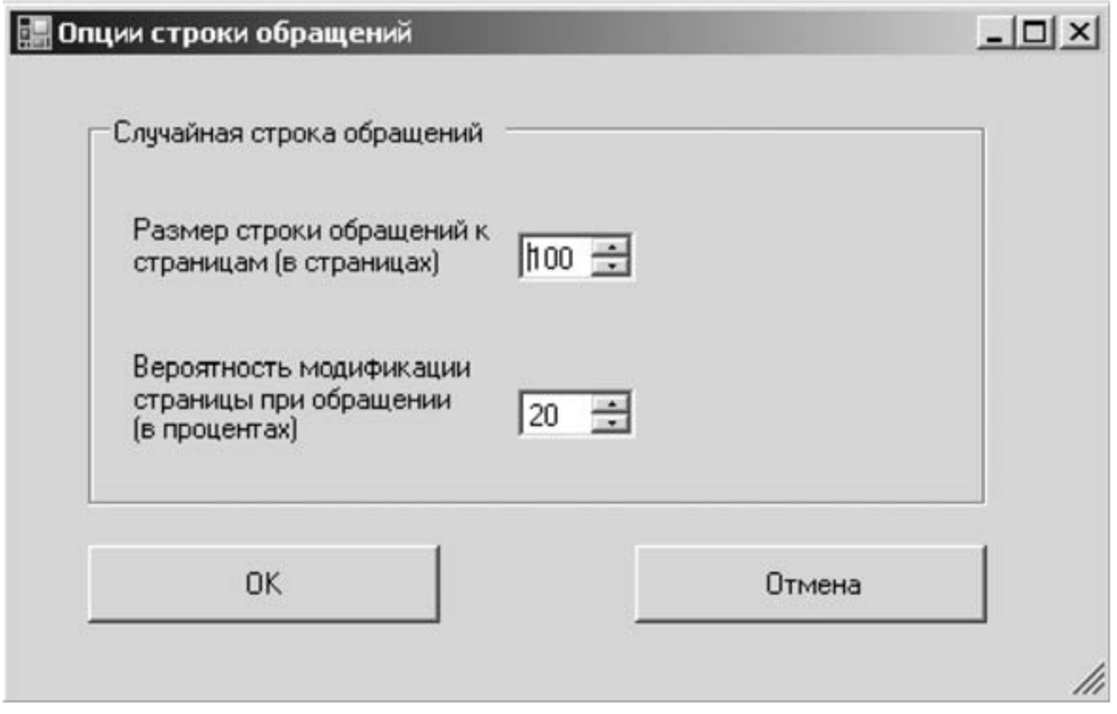


Рис. 3.21

Как видно на рис. 3.21, в окне приложения показаны:

- состояние физической памяти в виде таблицы;
- число страничных прерываний;
- строка обращения к страницам, где отражено, к каким страницам будет обращаться система на следующих шагах и тип обращения;
- информация о произошедшем шаге, где подробно описываются все изменения, произошедшие в физической памяти и изменения счетчика прерываний.

Создание строки сообщений выполняется в двух режимах. Первый режим позволяет создать случайную строку обращений. При этом используются следующие параметры: длина строки обращений и вероятность модификации страницы, количество физических страниц, количество виртуальных страниц, которые можно изменить в опциях строки обращений (рис. 3.22) и опциях алгоритмов (рис. 3.23):

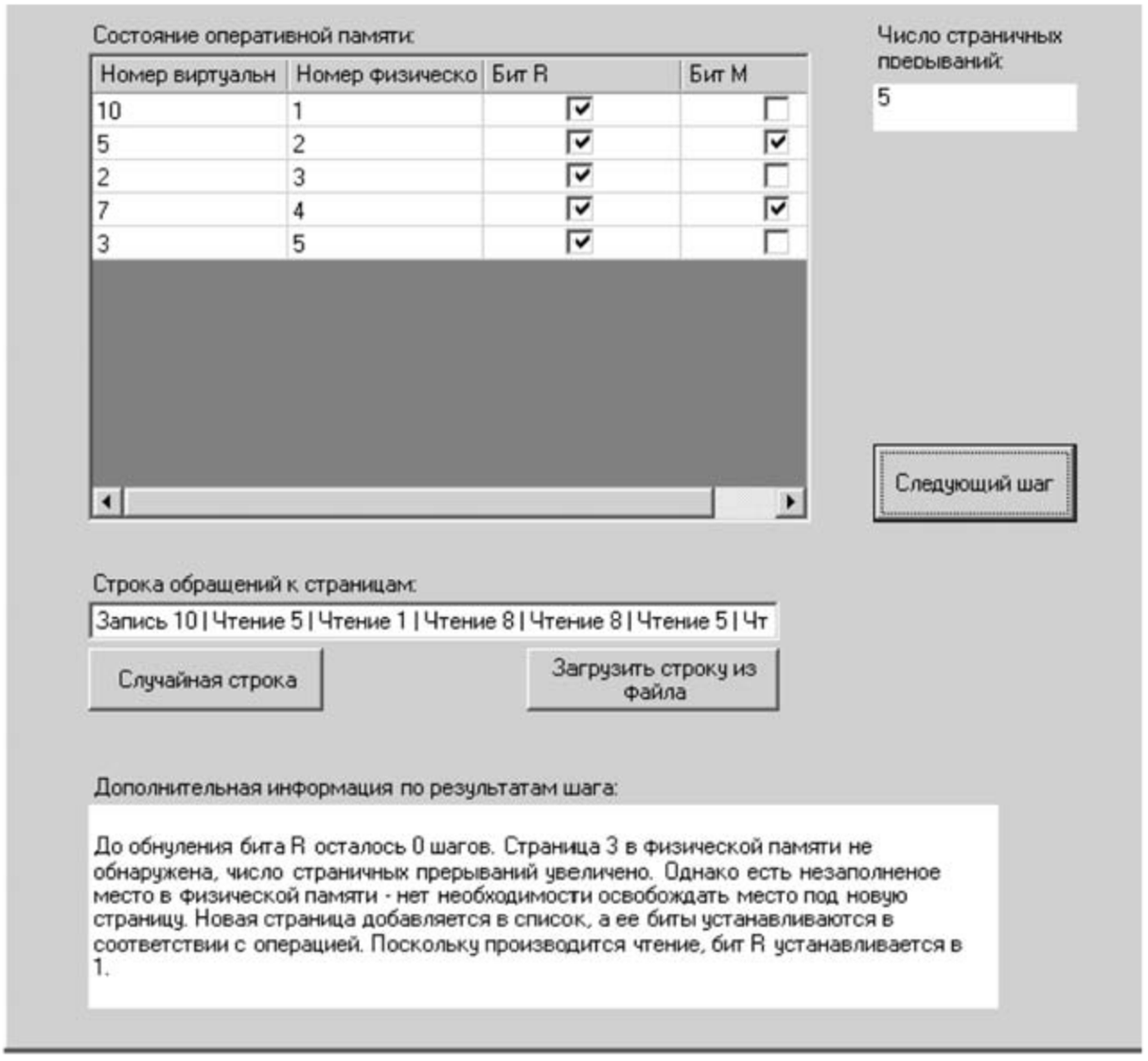


Рис. 3.22

Второй режим позволяет загружать строку обращений из файла с расширением que. Для создания файлов строки обращений можно использовать инструмент Создание очереди (рис. 3.24) из меню Файл.

Кроме пошагового исследования алгоритмов, пользователю предлагается второй способ — построение графиков зависимости количества страничных прерываний от различных параметров модели. Для этого используется инструмент Построение графиков (рис. 3.25).

Для построения графика необходимо выбрать, какие алгоритмы должны присутствовать на графике, и границы изменяемых параметров, а также количество опытов. Чем больше число опытов, тем более точной получится оценка. При расчете графиков для каждой точки в каждом опыте генерируется случайным образом строка обращений. При этом все параметры, кроме изменяемого, определены в опциях алгоритма и опциях строки обращений. Существует возможность обрабатывать данные исследования алгоритмов не только в самой про-

грамме, но и в других программных пакетах. Для этого предусмотрена возможность сохранить данные в формате CSV (кнопка Сохранить данные в файл). Это простой текстовый формат данных, который «понимают» многие программные средства, в том числе Microsoft Excel. Благодаря этой возможности можно проводить исследования и строить графики не только в самой программе.

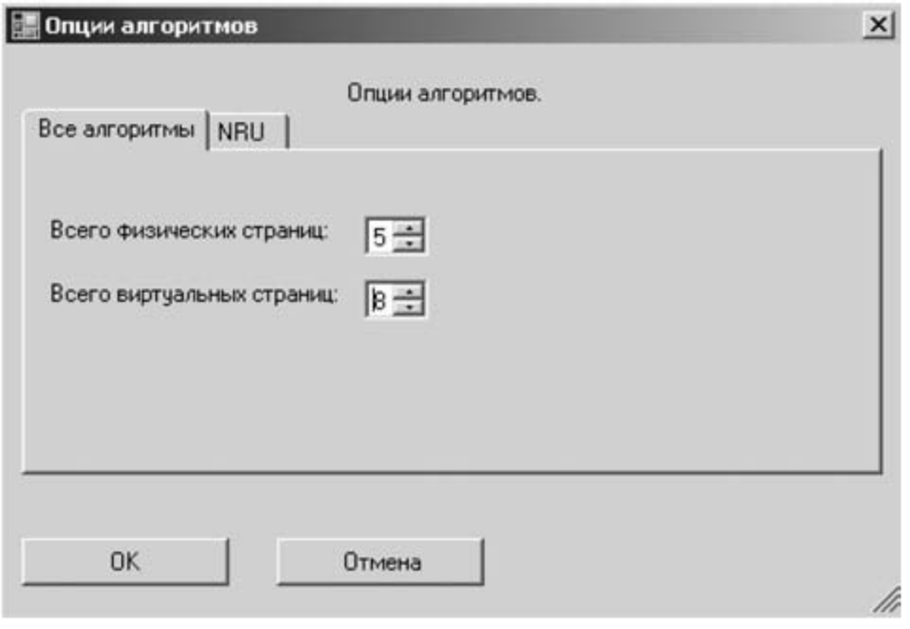


Рис. 3.23

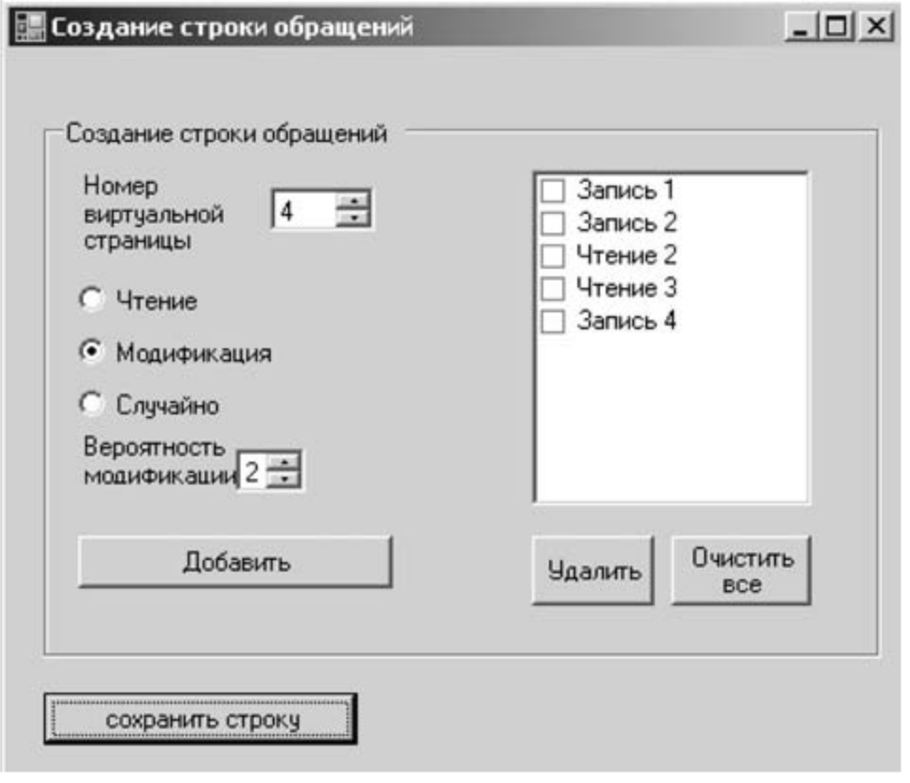


Рис. 3.24

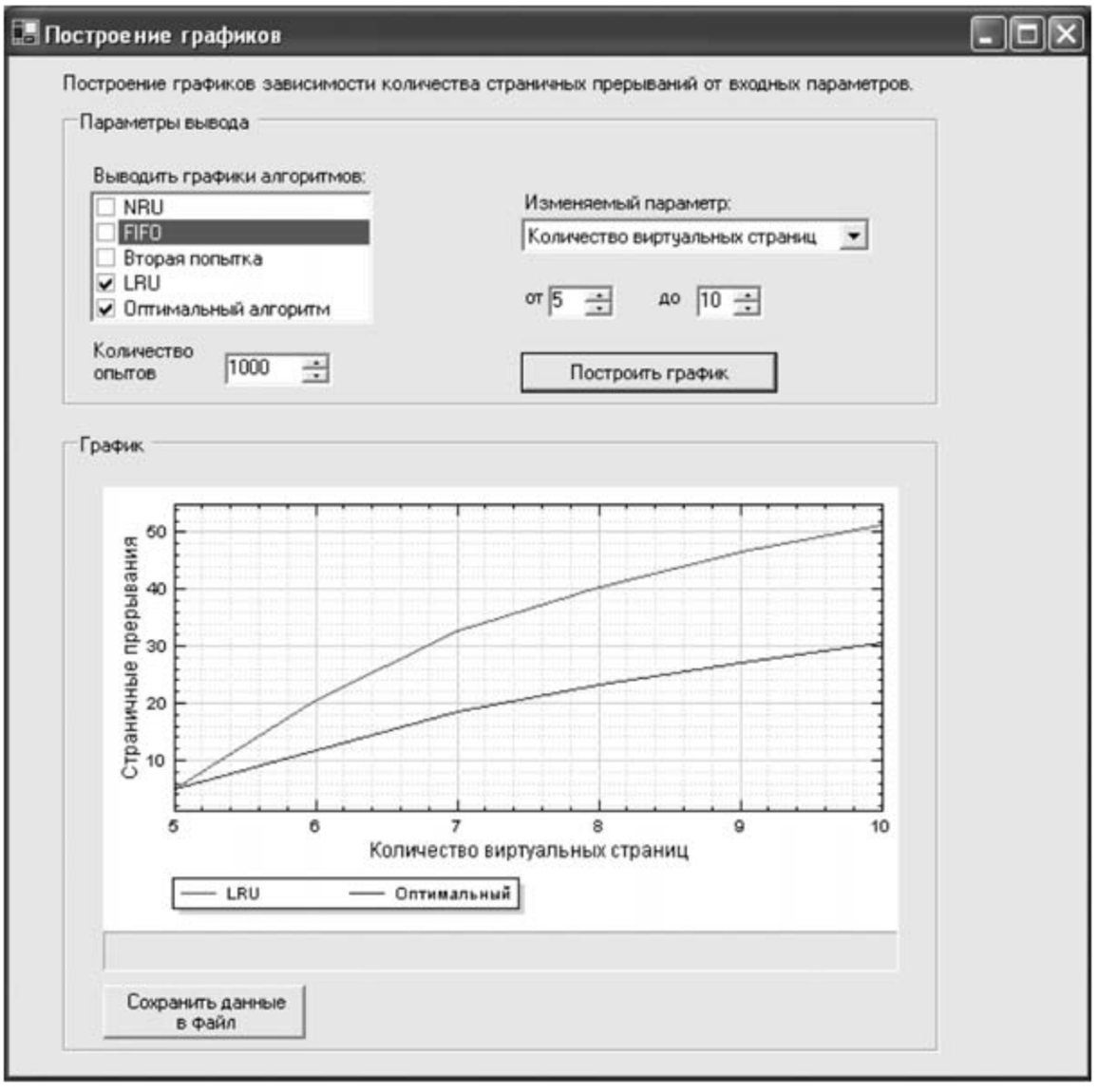


Рис. 3.25

3.8. Трансляция виртуальных адресов

В учебных материалах авторов на сайте www.hse.ru приведены программы, иллюстрирующие преобразование виртуального адреса процесса в физический для разных вариантов организации виртуальной памяти. Программа ProjectOS предназначена для иллюстрации процесса преобразования виртуального адреса в физический при страничной организации памяти. Считается, что каждому выполняющемуся заданию может быть выделено не более 1000 Кбайт виртуальной памяти. Это значение (в заданном диапазоне) устанавливается в поле Размер задания. Размер виртуальной страницы задается кнопочным переключателем и может составлять 1, 2, 4 или 64 Кбайт. Размер физической страницы

равен размеру виртуальной страницы. Программа предусматривает также задание разрядности виртуального адреса (24 или 32 разряда).

При активизации задания (процесса) ОС устанавливает в Регистре таблицы страниц номер задачи. Это активизирует Таблицу страниц текущей задачи. В регистре Таблицы страниц отображается номер задачи, к которому добавляется номер текущей виртуальной страницы P активного задания, поступающего из регистра Виртуального адреса.

Физический адрес формируется из номера физической страницы, считываемой из строки Таблицы страниц, номер которой соответствует значению P, т.е. номеру виртуальной страницы, из смещения, которое переписывается из поля смещения виртуального адреса без изменения. Значение физического адреса дается в десятичной и двоичной системах счисления (рис. 3.26).

Программа MemControl предназначена для иллюстрации процесса преобразования виртуального адреса в физический при сегментной организации памяти (рис. 3.27). Она позволяет задать размер задания (до 1 000 000 байт) и количество сегментов (до 5). После этого необходимо нажать кнопку Применить. При этом случайным образом формируются размеры сегментов, но так, что общая сумма их размеров равна Размеру задания. Таблица сегментов жестко фиксирована в памяти. Виртуальный адрес задается в одноименном поле и состоит из номера сегмента и смещения.

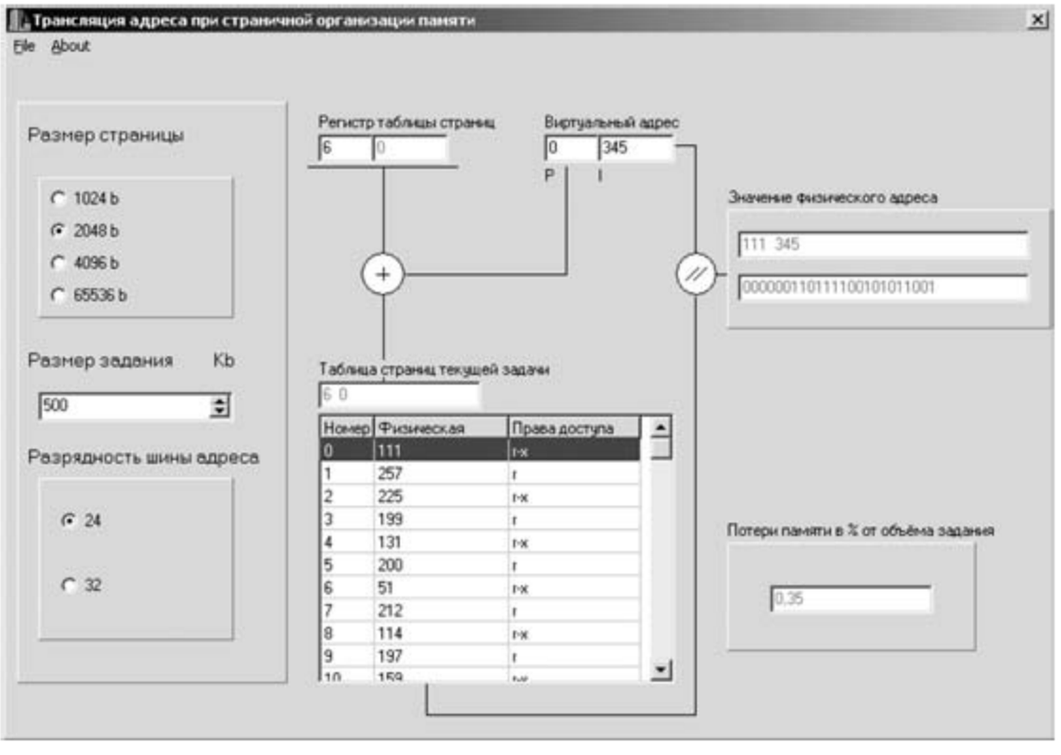


Рис. 3.26

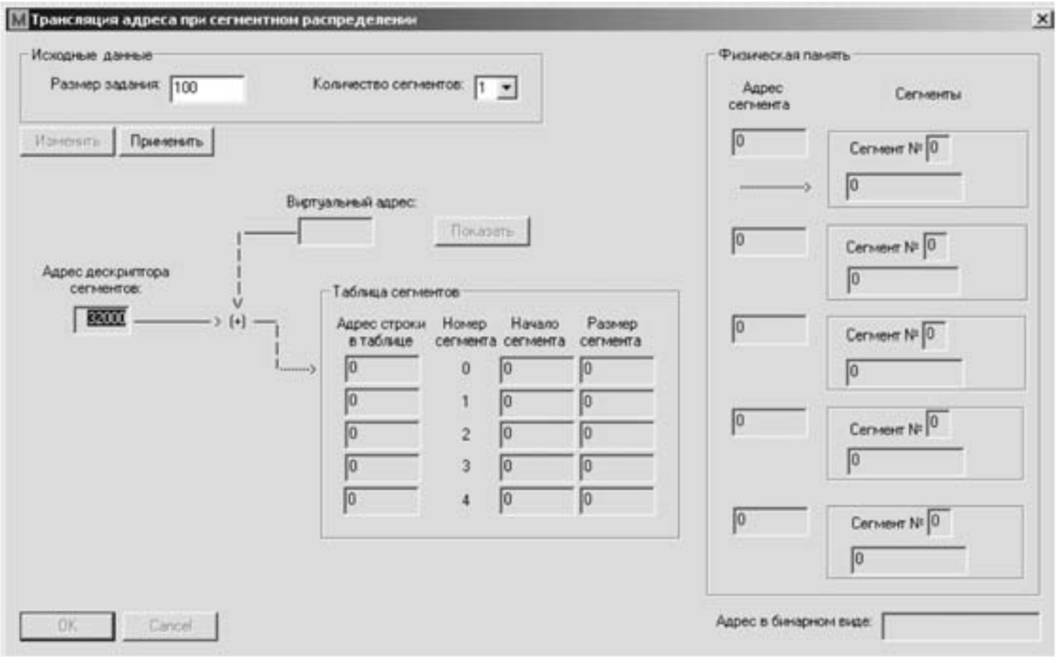


Рис. 3.27

После задания виртуального адреса для его преобразования в физический адрес следует нажать кнопку Показать. Если при этом смещение в виртуальном адресе не превосходит длину сегмента, в правой части окна высвечивается физический адрес в поле соответствующего сегмента. В противном случае выдается сообщение об ошибке задания смещения.

Программа ActualHW (рис. 3.28) подобна MemControl. Она позволяет преобразовать заданный виртуальный адрес в физический в следующих условиях управления памятью с сегментной организацией:

- виртуальный адрес 32-разрядный;
- для каждого процесса выделяется четыре сегмента;
- в системе может быть несколько процессов;
- оперативная память имеет размер 1Гб.

Таблица сегментов хранит четыре записи. Каждая запись представлена номером сегмента (2 бита), признаком наличия сегмента в оперативной памяти (1 бит), длиной сегмента (30 бит; максимум —1Гб), адресом начала в физической памяти (30 бит). Виртуальный адрес содержит 32 бита. Первые 2 бита — номер сегмента. На смещение отводится 30 бит. Сегмент имеет максимальный размер 1 Гб.

Для работы программы необходимо задать:

- количество процессов;
- четыре сегмента каждого процесса, определив их длину;
- виртуальный адрес, подлежащий преобразованию;
- номер активного процесса.

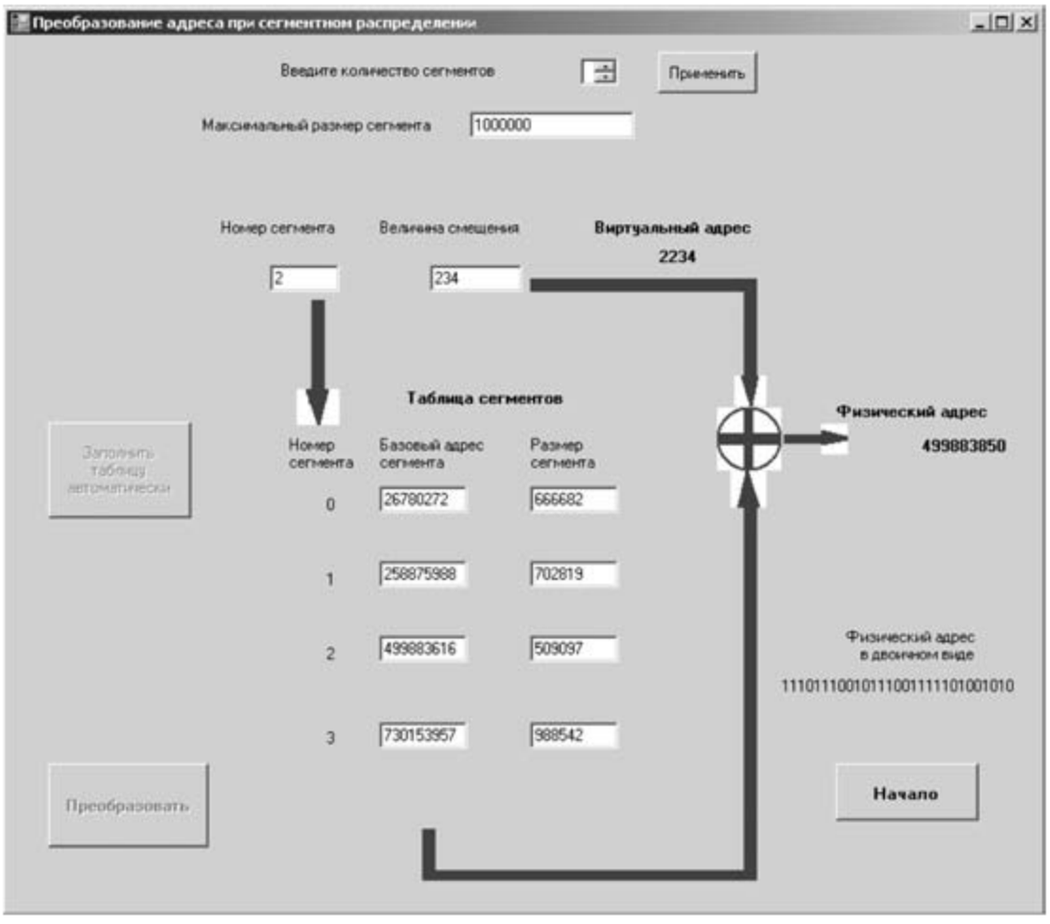


Рис. 3.28

При вводе данных поле для виртуального адреса задается в двоичном коде. Активный (рассматриваемый) процесс можно переключать кнопками < и >. Количество процессов вводится в специальное поле. Размер сегментов задается в байтах. После заполнения любого поля необходимо подтвердить изменения нажатием кнопки Применить. Чтобы получить результат перевода, пользователь должен нажать кнопку Преобразовать. Предусмотрен контроль правильности ввода.

После того как пользователь ввел размеры сегментов, производится запись данных в таблицу. Каждый сегмент представлен одной строкой таблицы.

В качестве результата представляется таблица сегментов, виртуальный адрес и физический адрес.

3.9. Оптимизация виртуальной памяти

Повышение производительности работы виртуальной памяти сводится к определению требуемого объема физической памяти, установлению рациональной интенсивности страничного обмена и оптимиза-

ции размера и размещения файла подкачки. Основным инструментом, который может быть использован для решения этих задач, является оснастка Производительность.

Поскольку производительность компьютера главным образом определяется ресурсами процессора и памяти, следует понимать, как программы используют эти ресурсы. Прежде чем приступить к наблюдению за использованием памяти на локальном компьютере, следует убедиться, что система обладает объемом памяти, рекомендуемым для работы данной ОС (например, Windows XP) и всех служб и программ. Необходимо сравнить имеющийся объем памяти с требованиями ОС и программ.

Чтобы определить требования к памяти, нужно к объему памяти, необходимому для работы ОС, добавить следующие значения:

- число пользователей, умноженное на средний размер файлов данных, открытых пользователем (для клиентского компьютера);
- число программ, запущенных на компьютере-сервере, умноженное на средний размер этих программ.

Если неизвестно, сколько памяти требуется для работы определенного процесса, можно включить отображение его рабочего множества в системном мониторе, затем завершить его работу и понаблюдать за соответствующим изменением использования подкачки на данном компьютере. Объем памяти, освобожденный по завершении работы программы, является объемом физической памяти, который требуется добавить в систему.

Просмотр значений определенных счетчиков на диаграмме системного монитора может помочь в определении объема памяти, используемой программами. Начать можно с наблюдения за счетчиком Процесс\Рабочее множество (рис. 3.29).

Для каждой программы, запускаемой на компьютере, ОС выделяется часть физической памяти, называемой рабочим множеством. Даже если программа не проявляет активности, ОС все равно выделяется память для ее рабочего множества. Эта величина может быть больше, чем минимальный объем памяти, действительно необходимый приложению. Она может отражать физический объем памяти, распределенный между несколькими процессами.

Используя счетчики, в системном мониторе можно построить диаграммы использования памяти. Значение рабочего множества представляет интерес, когда счетчик Память\Доступно байт (рис. 3.30) опускается ниже определенного значения. Windows XP удовлетворяет требования программ к памяти путем использования свободных (до-

ступных) байтов. Когда объем свободной памяти опускается ниже определенного значения, ОС начинает пополнять его, отбирая память у рабочих множеств или менее активных программ. В результате можно заметить, что значение рабочего множества одних программ увеличивается, а других — уменьшается. Если памяти недостаточно для удовлетворения требований всех активных программ, используется файл подкачки, что снижает производительность.

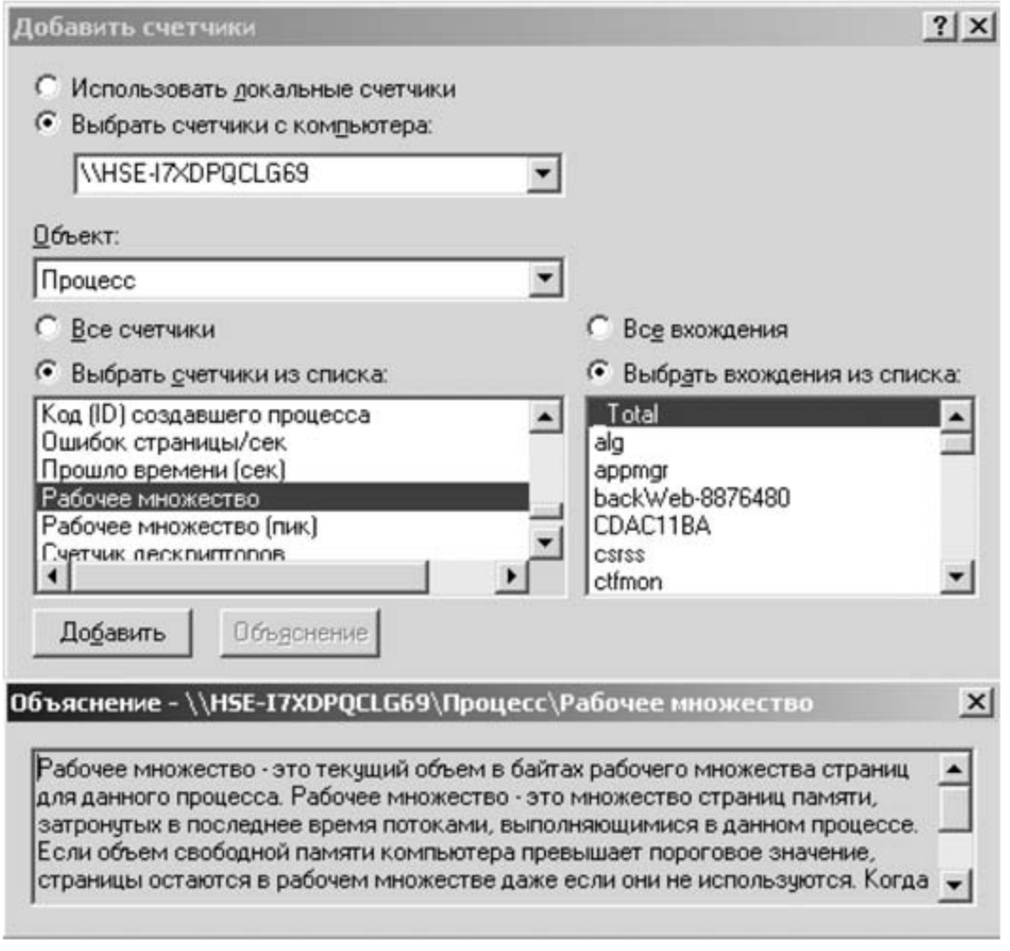


Рис. 3.29

Файл подкачки служит для перемещения страниц памяти, содержащих программы и данные, из оперативной памяти на диск для освобождения памяти для других целей. Использование файла подкачки позволяет увеличить объем памяти, доступный Windows XP. Однако постоянное его использование является причиной значительного снижения производительности.

Наблюдение за ситуациями, порождающими недостаток памяти, рекомендуется начинать со следующих счетчиков:

- Память\Доступно байт (см. рис. 3.30);
- Память\Обмен страниц в сек (рис. 3.31).

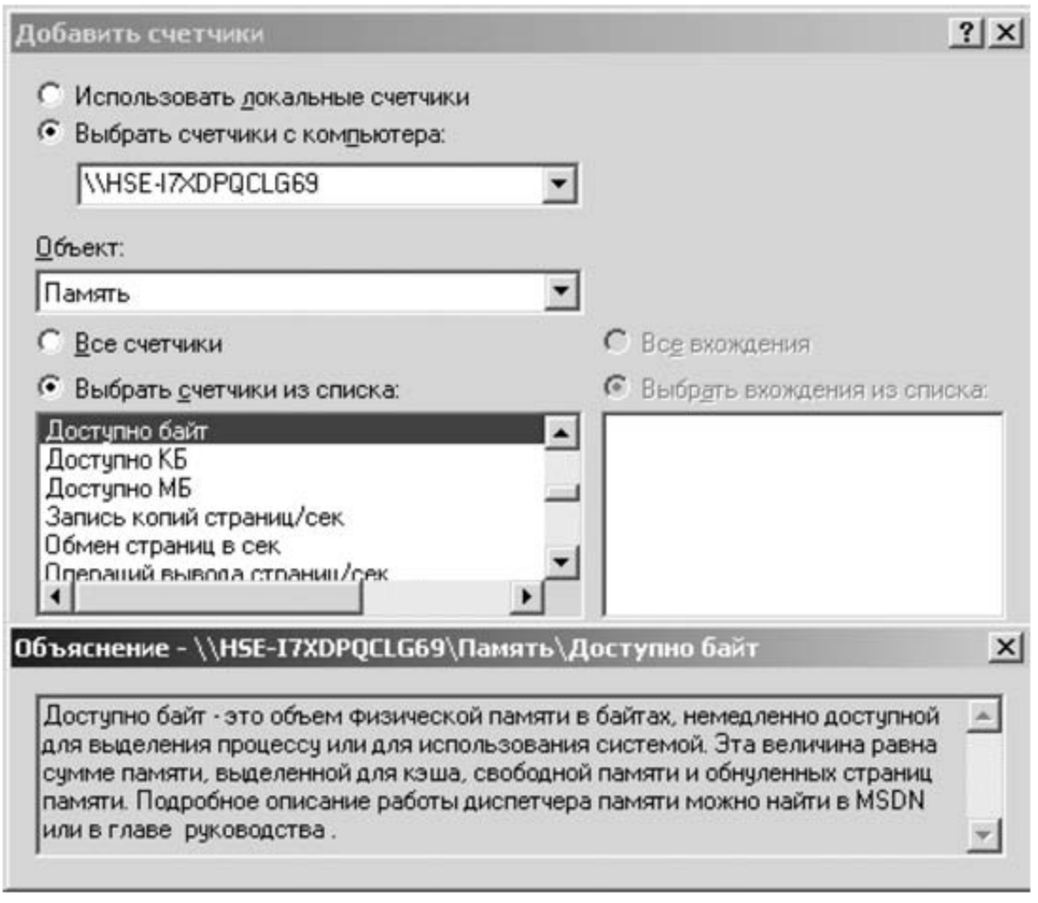


Рис. 3.30

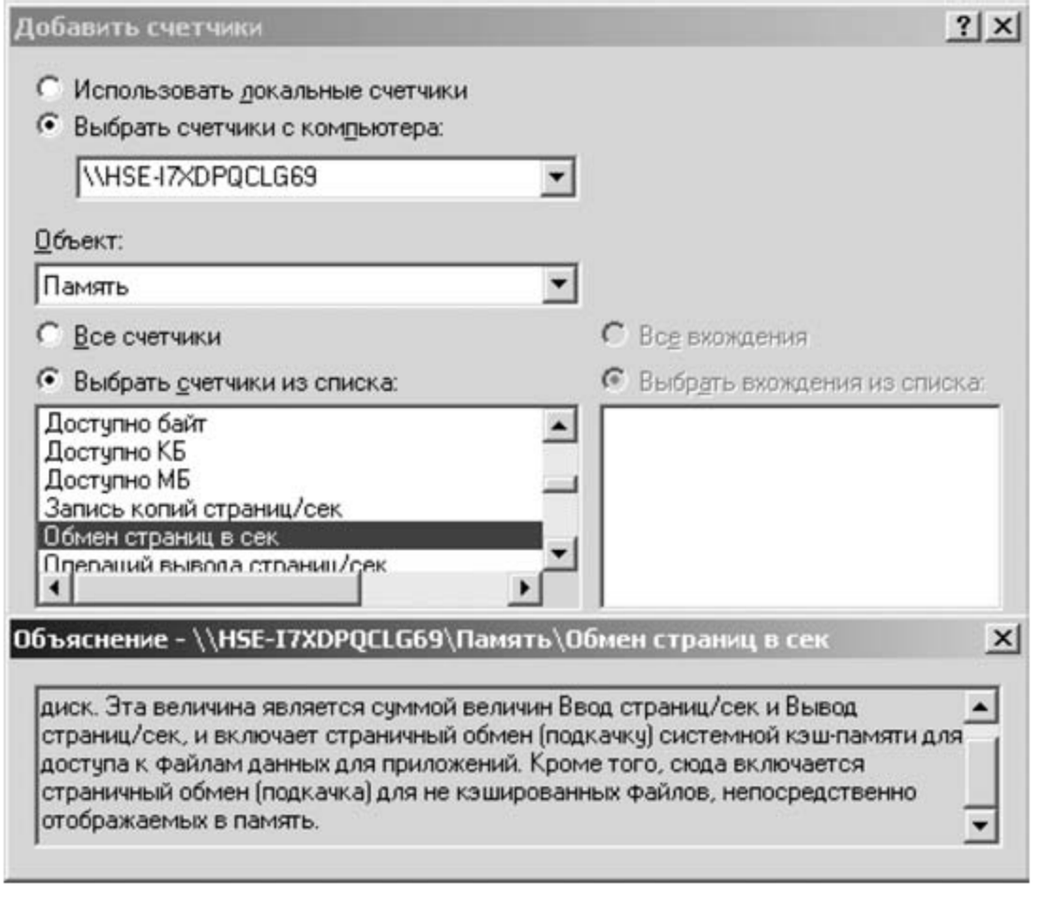


Рис. 3.31

Счетчик Доступно байт показывает текущий объем памяти в байтах, доступный для использования процессами. Счетчик Обмен страниц в сек показывает число страниц, полученных с диска при обращении к этим страницам или записанных на диск для освобождения свободной памяти в рабочем множестве.

Низкие значения счетчика Доступно байт (4 Мбайт и меньше) указывают на общий недостаток памяти на компьютере или на то, что какая-либо программа не освобождает память. Если значение Обмен страниц в сек достигает или превышает 20, следует внимательно изучить активность страничного обмена. Большое значение счетчика Обмен страниц в сек может не указывать на недостаток памяти, а являться результатом работы программы, которая использует файл, отображенный в памяти.

Чтобы определить, является ли причиной именно это, нужно понаблюдать за счетчиками Доступно байт, Обмен страниц в сек и Файл подкачки\% использования (рис. 3.32). Если происходит чтение некэшированного файла, отображенного в памяти, также будет наблюдаться низкая или средняя активность кэша.

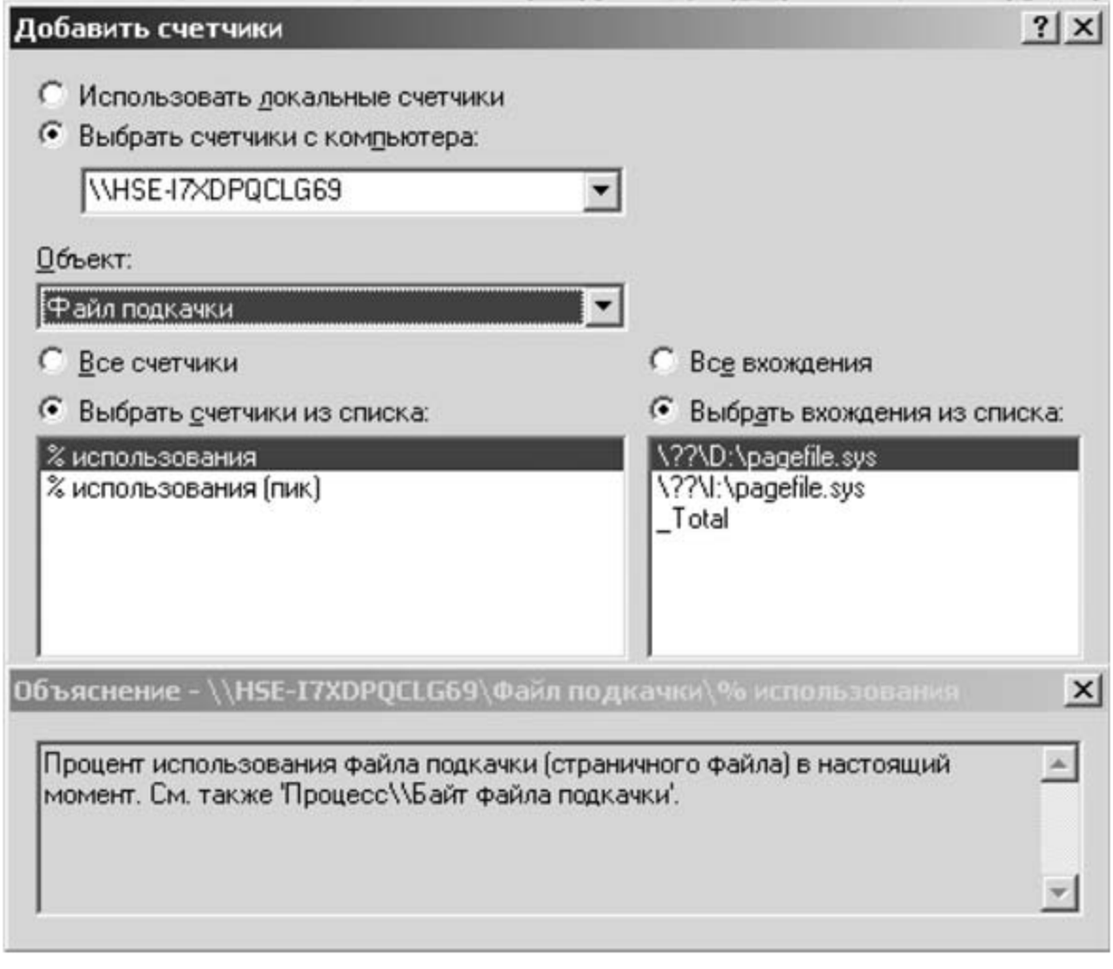


Рис. 3.32

Детальный анализ причин возникновения недостатка памяти требует наблюдения за счетчиками Память\Доступно байт и Память\Байт выделенной виртуальной памяти, чтобы отследить изменения объема памяти, и за счетчиками Процесс\Байт исключительного пользования, Процесс\Рабочее множество и Процесс\Счетчик дескрипторов процессов (рис. 3.33), которые, как предполагается, вызывают нехватку памяти. Также необходимо наблюдение за счетчиками Память\Байт в невыгружаемом страничном пуле, Память\Распределений в невыгружаемом страничном пуле и Процесс (имя_процесса)\Байт в невыгружаемом страничном пуле, если предполагается, что нехватка памяти обусловлена процессом ядра.

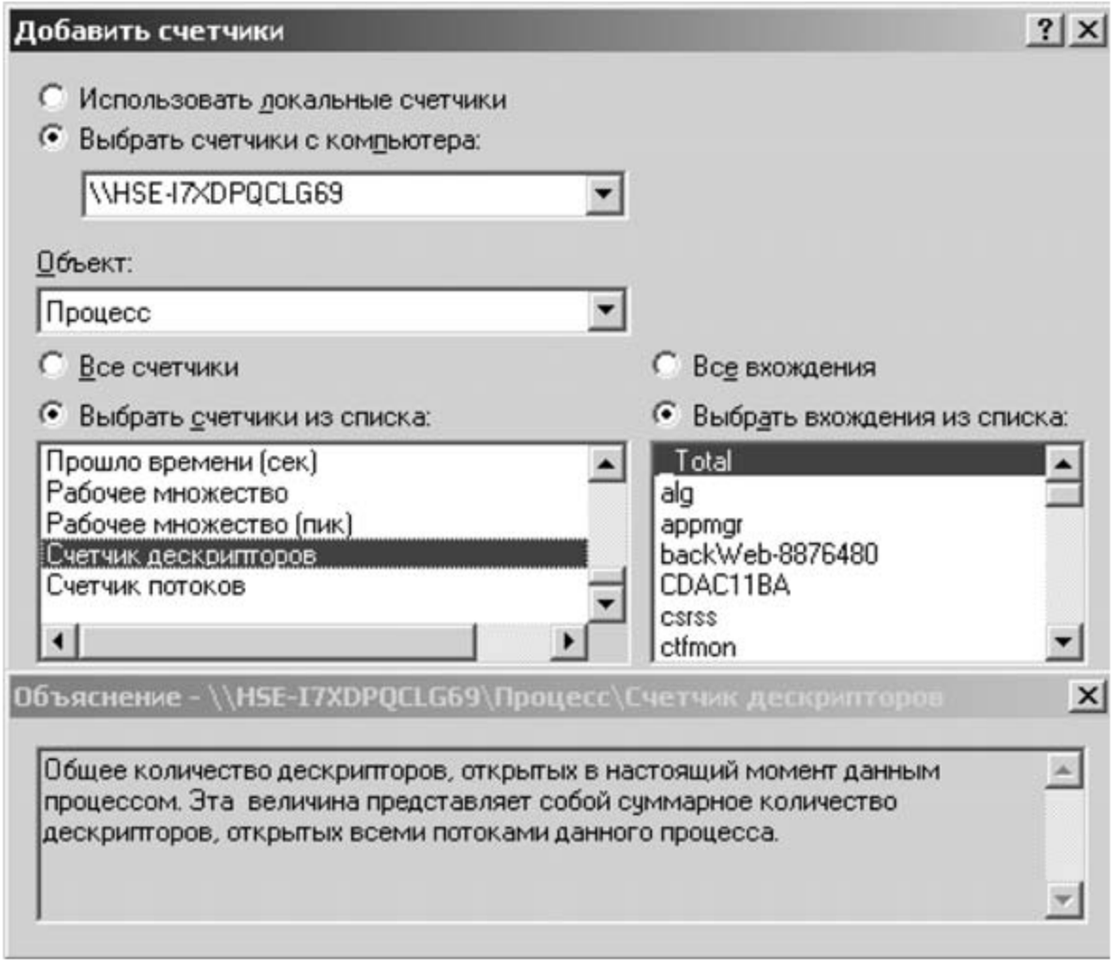


Рис. 3.33

Поскольку избыточная подкачка влечет сильную загрузку жесткого диска, кроме недостатка памяти, возможно также возникновение узкого места в дисковой системе. Поэтому, если при определении причины избыточной подкачки страниц недостаток памяти явно не прослеживается, наряду со счетчиками памяти следует наблюдать за счетчиками использования диска:

- Логический диск\% активности диска (рис. 3.34);
- Физический диск\Средняя длина очереди диска.

Например, данные счетчиков Чтение страниц/сек, % активности диска и Средняя длина очереди диска, показывающие сочетание низкой активности чтения страниц с высокими значениями активности диска и средней длины очереди диска, указывают на наличие узкого места в дисковой системе. Однако если увеличение длины очереди не сопровождается уменьшением частоты чтения страниц, это означает нехватку памяти.

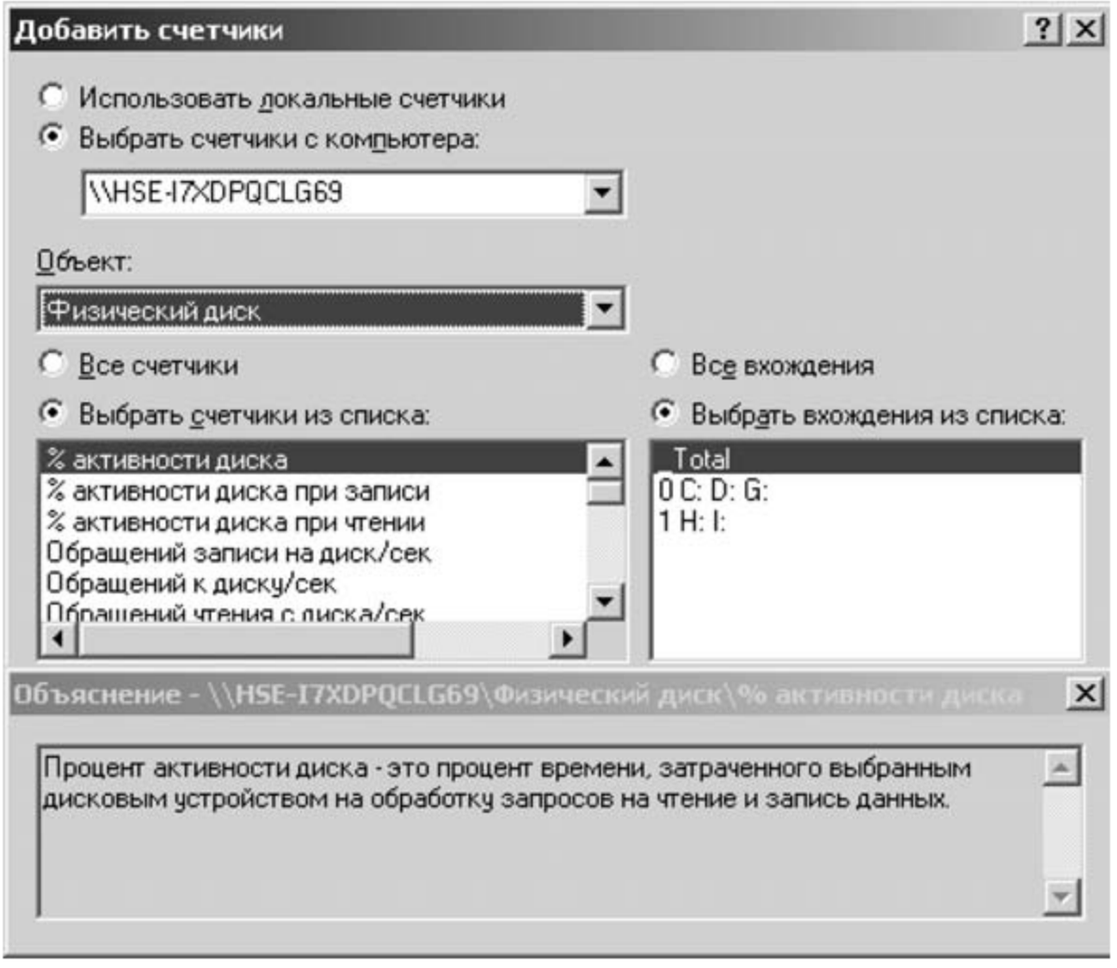


Рис. 3.34

Чтобы определить влияние избыточной подкачки на активность диска, нужно перемножить значения счетчиков Физический диск\Среднее время обращения к диску (сек) и Память\Обмен страниц/сек. Если произведение этих счетчиков превышает значение 0,1, подкачка занимает более 10% времени доступа к диску. Если такая ситуация наблюдается длительное время, следует нарастить объем памяти.

Целесообразно также проверить зависимость избыточной подкачки от запущенных программ. Для этого следует остановить (если

возможно) работу программы, когда рабочее множество имеет наибольшее значение, и посмотреть, как при этом изменится частота подкачки страниц. При обнаружении избыточной подкачки нужно проверить значения счетчика Память\Обмен страниц/сек. Он показывает число страниц, которые должны быть прочитаны с диска, так как они отсутствуют в физической памяти. Этот счетчик отличается от счетчика Ошибок страницы/сек, указывающего только на то, что доступ к данным не был получен немедленно, так как они не были найдены в заданном рабочем множестве страниц памяти.

Существует несколько приемов, позволяющих оптимизировать использование файла подкачки для повышения производительности. Файл подкачки по возможности следует размещать на отдельном жестком диске. При наличии нескольких жестких дисков файл подкачки нужно разделить, так как это повышает скорость работы с ним. Если при наличии двух жестких дисков разделить файл подкачки, то, поскольку доступ к данным на обоих жестких дисках осуществляется одновременно, это значительно повысит производительность. Однако, если имеются два жестких диска, из которых один быстрее другого, возможно, более эффективным решением будет размещение файла подкачки только на более быстром жестком диске. Определить наиболее производительную конфигурацию для данной системы можно экспериментальным путем.

Важен также размер файла подкачки. При запуске Windows XP на диске, где установлена эта ОС, автоматически создается файл подкачки (Pagefile.sys). Рекомендуется установить размер файла подкачки в 1,5—2 раза больше размера установленной оперативной памяти. В то же время размер этого файла также зависит от объема свободного пространства на жестком диске во время создания этого файла. Определить размер файла подкачки можно, узнав в проводнике размер файла Pagefile.sys.

Если на жестком диске еще есть свободное пространство, можно увеличить размер файла подкачки. Если пользователи обычно запускают несколько программ одновременно, при увеличении размера файла подкачки их запуск может ускориться. Хотя можно оставить предложенные системой размеры файла подкачки (исходный и максимальный), рекомендуется увеличить исходный размер, чтобы при запуске программ системе не приходилось увеличивать размер файла подкачки, фрагментируя его.

Когда размер файла подкачки достигает максимального значения, появляется сообщение о возможной остановке работы системы. Что-

бы выяснить, достигает ли размер файла подкачки максимального значения, нужно сравнить реальный размер файла с его максимальным размером, который задается в окне Свойства системы, открываемом с панели управления. Если эти значения близки, следует увеличить исходный размер файла подкачки или запускать одновременно меньшее количество приложений.

Другим способом определения оптимального значения файла подкачки является использование счетчиков файла подкачки:

- Файл подкачки\% использования;
- Файл подкачки\% использования (пик).

Если значение счетчика «% использования (пик)» достигает максимального размера файла подкачки или значение счетчика «% использования» близко к 100%, можно попробовать увеличить исходный размер файла подкачки. Если файлы подкачки распределены по нескольким дискам, в качестве экземпляров счетчиков объекта Файл подкачки будут отображаться полные имена файлов подкачки. Можно либо добавить счетчик для каждого файла подкачки, либо выбрать экземпляр _Total для наблюдения за общей активностью всех файлов подкачки.

Задачи

3.1. Система устраняет свободные участки памяти с помощью уплотнения. Предположим, что множество свободных участков и множество сегментов данных распределены случайно, а время для чтения или записи 32-разрядного слова в памяти равно 10 нс. Сколько времени займет уплотнение 128 Мбайт памяти в худшем случае? Построить график времени уплотнения в зависимости от объема занятой памяти.

3.2. Вычислить номер виртуальной страницы и смещение для виртуальных адресов 1230005, 3274893 и 6055445, если размер страницы равен 4 или 8 Кбайт.

3.3. Объем пространства на диске, который должен быть доступен для хранения страниц, связан с максимальным количеством процессов N, количеством байтов в виртуальном адресном пространстве V и числом байтов в оперативной памяти R. Выведете формулу требований на дисковое пространство в худшем случае. Насколько эта величина реалистична?

3.4. Вычислить номер виртуальной страницы и смещение для виртуальных адресов 1205600, 32789 и 13560445, если размер страницы равен 4 или 64 Кбайт.

3.5. Считая, что команда выполняется за 10 нс, а страничное прерывание требует дополнительно N нс, напишите выражение для фактического времени выполнения команды с учетом того, что прерывания происходят каждые K команд программы.

3.6. Компьютер имеет 32-разрядное адресное пространство и страницы размером 8 Кбайт. Таблица страниц целиком поддерживается аппаратно, на запись в ней отводится одно 32-разрядное слово. При запуске процесса таблица страниц копируется из памяти в аппаратуру, одно слово требует 10 нс. Если каждый процесс работает в течение 100 мс (включая время загрузки таблицы страниц), какая доля времени процессора жертвуется на загрузку таблицы страниц?

3.7. Компьютер с 32-разрядным адресом использует двухуровневую таблицу страниц. Виртуальные адреса расщепляются на 9-разрядное поле верхнего уровня таблицы, 11-разрядное поле второго уровня таблицы страниц и смещение. Чему равен размер страниц и сколько их в адресном пространстве?

3.8. Компьютер поддерживает 48-разрядные виртуальные адреса и 32-разрядные физические адреса. Размер страницы равен 8 Кбайт. Сколько требуется записей в таблице страниц и чему равен ее объем?

3.9. ОС использует алгоритм замещения страниц FIFO в системе с четырьмя страничными блоками и восемью страницами. Требуется определить, сколько страничных прерываний произойдет в системе для последовательности обращений 0172327103 при условии, что четыре страничных блока изначально пусты.

3.10. ОС использует алгоритм замещения страниц LRU в системе с четырьмя страничными блоками и восемью страницами. Требуется определить, сколько страничных прерываний произойдет в системе для последовательности обращений 0172327103 при условии, что четыре страничных блока изначально пусты.

3.11. В компьютере есть кэш, основная память и диск, который используется для организации виртуальной памяти. Если слово, к которому производится обращение, находится в кэше, для доступа к нему требуется 2 нс. Если это слово находится в основной памяти, но отсутствует в кэше, то оно сначала загружается в кэш за 10 нс, а затем к нему производится обращение. Если нужного слова нет в памяти, то чтобы найти его на диске и загрузить в основную память, требуется 12 мс; еще 10 нс нужны, чтобы скопировать его в кэш, и только затем к этому слову производится обращение. Результативность обращений к кэшу равна 0,9, а результативность обращений к основной памяти —

0,6. Требуется определить среднее время, которое требуется для доступа системы к нужному ей слову.

3.12. Вычислить номер виртуальной страницы и смещение для виртуальных адресов 1230005, 3274893 и 6055445, если размер страницы равен 4 или 8 Кбайт.

3.13. Объем пространства на диске, который должен быть доступен для хранения страниц, ограничен максимальным количеством процессов N, количеством байтов в виртуальном адресном пространстве V и числом байтов в оперативной памяти R. Выведете формулу требований на дисковое пространство в худшем случае. Насколько эта величина реалистична?

3.14. Вычислить номер виртуальной страницы и смещение для виртуальных адресов 1205600, 32789 и 13560445, если размер страницы равен 4 или 64 Кбайт.

3.15. Считая, что команда выполняется за 10 нс, а страничное прерывание требует дополнительно N нс, запишите выражение для фактического времени выполнения команды с учетом того, что прерывания происходят каждые K команд программы.

3.16. Компьютер с 32-разрядным адресом использует двухуровневую таблицу страниц. Виртуальные адреса расщепляются на 9-разрядное поле верхнего уровня таблицы, 11-разрядное поле второго уровня таблицы страниц и смещение. Чему равен размер страниц и сколько их в адресном пространстве?

3.17. Компьютер поддерживает 48-разрядные виртуальные адреса и 32-разрядные физические адреса. Размер страницы равен 8 Кбайт. Сколько требуется записей в таблице страниц и чему равен ее объем?

3.18. Компьютер, чьи процессы имеют 1024 страницы в своем адресном пространстве, хранит таблицы страниц в памяти. На чтение слова из таблицы страниц требуется 5 нс. Чтобы уменьшить затраты, в компьютере существует буфер быстрого преобразования адреса (TLB), содержащий 32 пары (виртуальная страница — физический страничный блок), который может выполнять поиск за 1 нс. При какой частоте обращений к памяти, успешно реализуемых в TLB, средние затраты на преобразование виртуального адреса будут ниже 2 нс?

ГЛАВА 4

СИСТЕМА ВВОДА-ВЫВОДА И ФАЙЛОВАЯ СИСТЕМА

4.1. Драйверы устройств

Задача системы ввода-вывода ОС Windows заключается в предоставлении основных средств (каркаса) для эффективного управления широким спектром устройств ввода-вывода. Основу этих средств составляет набор независимых от устройств процедур для определенных аспектов ввода-вывода и набор загруженных драйверов для общения с устройствами. Формирует этот каркас Менеджер ввода-вывода, который предоставляет другим модулям ОС независимый от устройств ввод-вывод, вызывая для выполнения физического ввода-вывода соответствующий драйвер.

Файловые системы формально являются драйверами устройств, работающих под управлением Менеджера ввода-вывода. В операционной системе Windows существует два драйвера для файловых систем — FAT и NTFS, которые независимы друг от друга и управляют различными разделами диска или различными дисками.

Чтобы гарантировать, что драйверы устройств хорошо работают с остальной частью ОС, корпорация Microsoft определила для драйверов модель Windows Driver Model, которой должны соответствовать драйверы устройств. Разработчикам драйверов предоставляется набор инструментов, который должен помочь в создании драйверов, удовлетворяющих требованиям этой модели.

Утилита Drivers из набора средств Microsoft Windows Resource Kit позволяет получить детальную информацию о загруженных драйверах в текстовом формате. Утилита запускается в командной строке и выдает следующую информацию:

- ModuleName — имя файла драйвера. Полный путь здесь не приводится, но большинство драйверов локализованы в папках System или System32;
- Code — длина исполняемой части драйвера;

■ Data — длина неблокируемой части драйвера. Эта информация является частью исполнимого образа драйвера;

■ Bss — длина части bss —присуща только 16-битным драйверам; как правило, таких драйверов не должно быть;

■ Paged — длина части драйвера, загружаемая в память;

■ Init — размер файла драйвера на диске. Поскольку большинство исполняемых программ хранятся в сжатом виде, это значение может совпадать или не совпадать с суммой значений колонок Code и Data;

■ LinkDate — дата и время компоновки файла.

После запуска программы в командной строке на экране отображается информация, показанная на рис. 4.1.

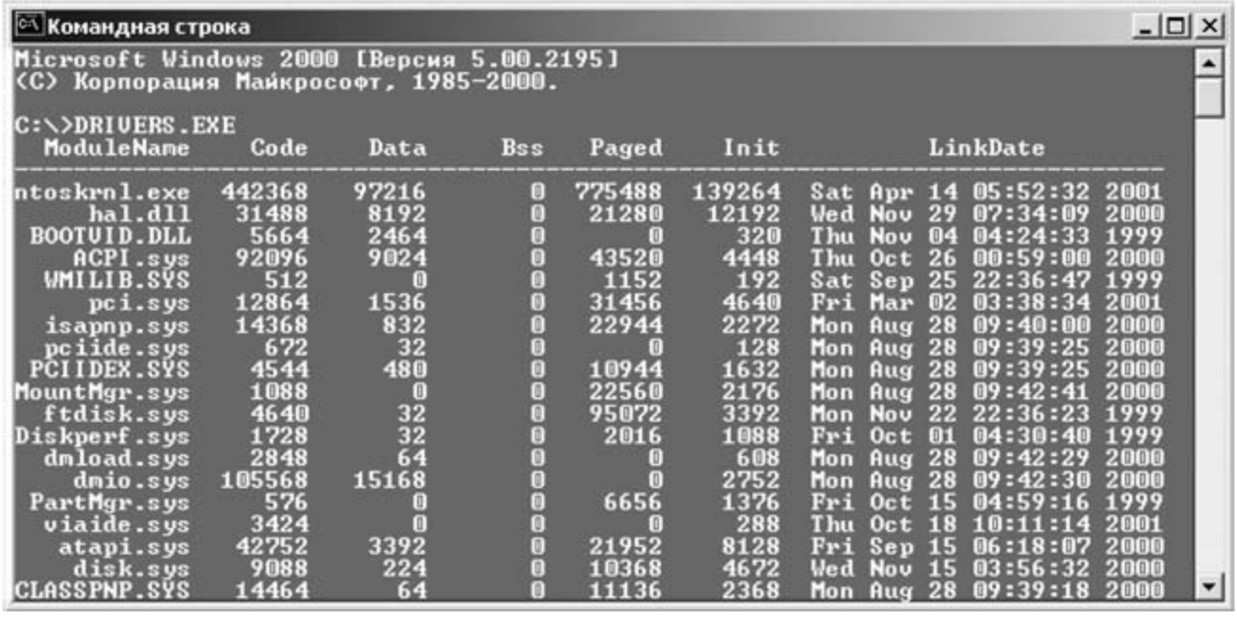


Рис. 4.1

Корпорацией Microsoft разработана утилита Bootvis [13], позволяющая выявлять проблемы, возникающие в процессе загрузки операционной системы. Эта утилита выполняет трассировку всех этапов загрузки системы, в том числе этапов загрузки системного ядра, драйверов устройств и запуска процессов. Утилита не входит в стандартную поставку Windows, но ее можно загрузить из Интернета (http://download.microsoft.com: 80/download/whistler/BTV/1.0/WXP/EN-US/BootVis-Tool.exe). После загрузки копии Bootvis ее нужно установить, щелкнув два раза в файле утилиты.

Утилита Bootvis помогает решить проблемы, связанные с драйверами и службами, которые находятся в памяти во время загрузки системы. Например, если в результате работы утилиты будет найден драйвер, который загружается очень долго, имеет смысл поискать для

соответствующего устройства новый драйвер. Перед обновлением драйвера или установкой новой службы можно провести эталонное тестирование, чтобы после модернизации можно было сравнить прежние и новые результаты и понять, повысилась производительность системы или нет.

В большинстве случаев установка последних версий драйверов и приложений помогает решить возникающие проблемы. Однако иногда обновление может привести к новым проблемам. В этих случаях нужно посмотреть, какое влияние оказала смена драйвера или установка службы. Для этого можно провести тестирование времени загрузки драйвера с помощью утилиты Bootvis:

1) запустить утилиту Bootvis и установить режим трассировки (рис. 4.2), выбрав команду Next Boot + Driver Delays (следующая загрузка плюс задержки драйверов) в меню Trace (трассировка). После этого появится диалоговое окно (рис. 4.3), в котором следует указать количество трассировок, которое нужно выполнить. Для точной оценки необходимо выполнить трассировку не менее трех раз. Количество повторений устанавливается с помощью кнопок со стрелками;

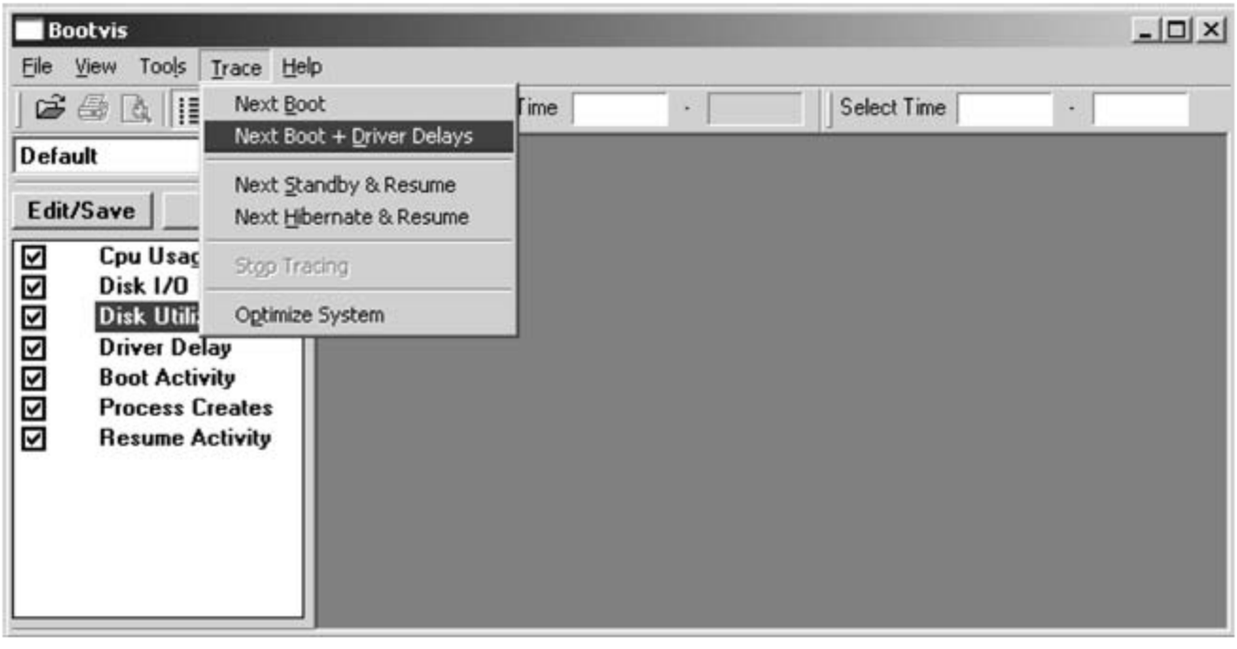


Рис. 4.2

2) щелкнуть по кнопке OK, после чего появится окно с обратным отсчетом времени (перезагрузка будет выполнена через 10 с). Можно не дожидаться автоматической перезагрузки и щелкнуть по кнопке Reboot Now;

Рис. 4.3

3) сразу после начала загрузки системы стартует процесс трассировки. В это время не следует нажимать никаких клавиш, кроме тех, которые необходимы для регистрации на компьютере. После регистрации утилита Bootvis выведет на экран сообщение о том, что пользователь ничего не должен делать на компьютере. Завершив трассировку, компьютер автоматически перезагрузится, и процесс повторится столько раз, сколько было указано ранее при настройке Bootvis;
4) когда все итерации будут выполнены, компьютер автоматически загрузит файл трассировки. Если трассировок было несколько, необходимо загрузить один из файлов трассировки. Для этого нужно снова запустить Bootvis, выбрать в меню File команду Open и найти файлы трассировки (они имеют расширение BIN). В качестве примера на рис. 4.4 показано открытие файла TRACE_BOOT-DRAVERS_1_2;
5) открыв нужный файл, Bootvis отобразит результаты трассировки (рис. 4.5). Наиболее важная информация, которую предоставляет утилита Bootvis, содержится в разделе Driver Delay. Загрузка драйверов занимает большую часть времени при старте системы. Выявив наиболее медленные драйверы устройств, можно ускорить процесс загрузки всей системы. Для этого нужно прокрутить окно с полученной информацией, чтобы увидеть раздел Driver Delay. Все драйверы устройств, загружаемые во время старта системы, делятся на различные категории и представляются зелеными прямоугольниками. Если поместить указатель мыши на определенный прямоугольник, появится подсказка с названием драйвера и примерным временем его загрузки

(см. рис. 4.5). Используя горизонтальную прокрутку, можно увидеть, какие драйверы и в какой последовательности загружались при старте системы;

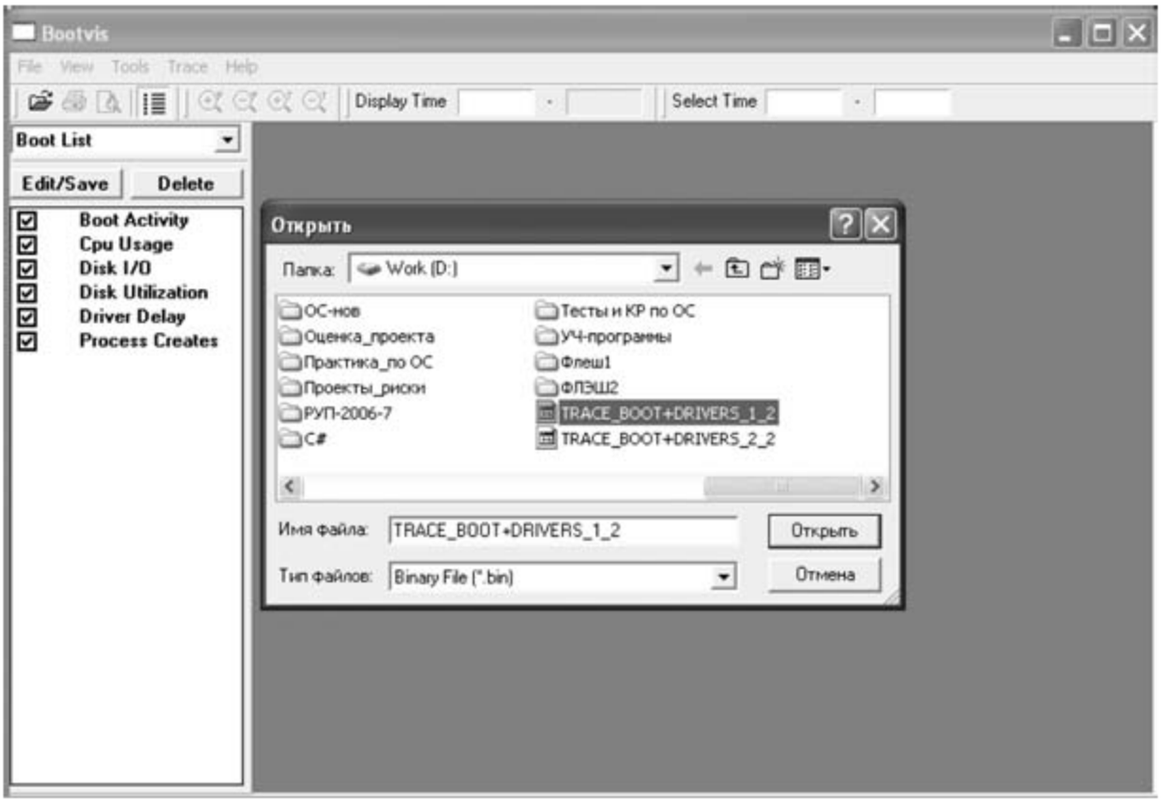


Рис. 4.4

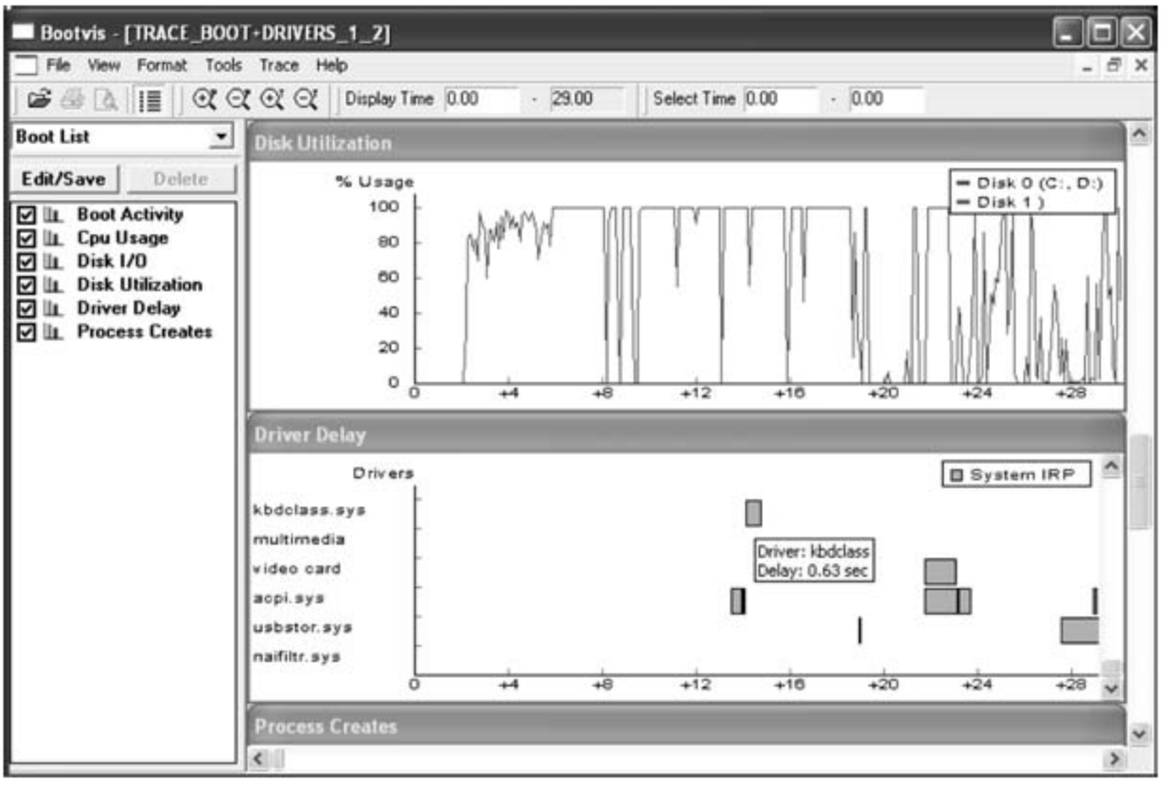


Рис. 4.5

6) выписав все интервалы задержки для каждого из файлов трассировки, можно вычислить среднее арифметическое значение загрузки для каждого драйвера. После обновления определенного драйвера можно повторить рассмотренную процедуру для вычисления новой оценки, которую можно сравнить с прежней. Это позволит принять правильное решение о целесообразности замены того или иного драйвера.

Задания для самостоятельной работы

1. Используя рассмотренные средства, просмотреть список драйверов компьютера, определить их количество и занимаемый ими объем памяти.
2. С помощью утилиты Bootvis провести трехкратную трассировку загрузки ОС. Выполнить анализ полученной информации. Определить драйверы устройств, которые занимают 50% времени загрузки ОС.

4.2. Диспетчер устройств

Диспетчер устройств используют для обновления драйверов (или программного обеспечения) оборудования, изменения настроек оборудования, а также для устранения неполадок. Драйверы устройств для аппаратных продуктов с эмблемой Для Microsoft Windows XP или Для Microsoft Windows Server 2003 снабжаются цифровой подписью корпорации Microsoft, которая подтверждает, что данный продукт проверен на совместимость с Windows и не изменился после проведения проверки. В окне диспетчера устройств представлено графическое отображение оборудования, установленного на компьютер. Для открытия окна Диспетчера устройств нужно щелкнуть правой клавишей мыши по значку Мой компьютер и выбрать в контекстном меню строку Свойства. В открывшемся окне Свойства системы перейти на вкладку Оборудование и нажать кнопку Диспетчер устройств (рис. 4.6).

В окне Диспетчера устройств (рис. 4.7) можно, раскрывая соответствующие узлы, видеть устройства, которые либо подключены и работают, либо отключены. Диспетчер устройств обычно используется для

проверки состояния оборудования, подключения-отключения оборудования и обновления драйверов устройств, установленных на компьютере. Кроме того, возможности диагностики диспетчера устройств могут использоваться опытными пользователями, обладающими глубокими знаниями о компьютерном оборудовании, для разрешения конфликтов устройств и изменения параметров ресурсов, однако при этом следует соблюдать большую осторожность.

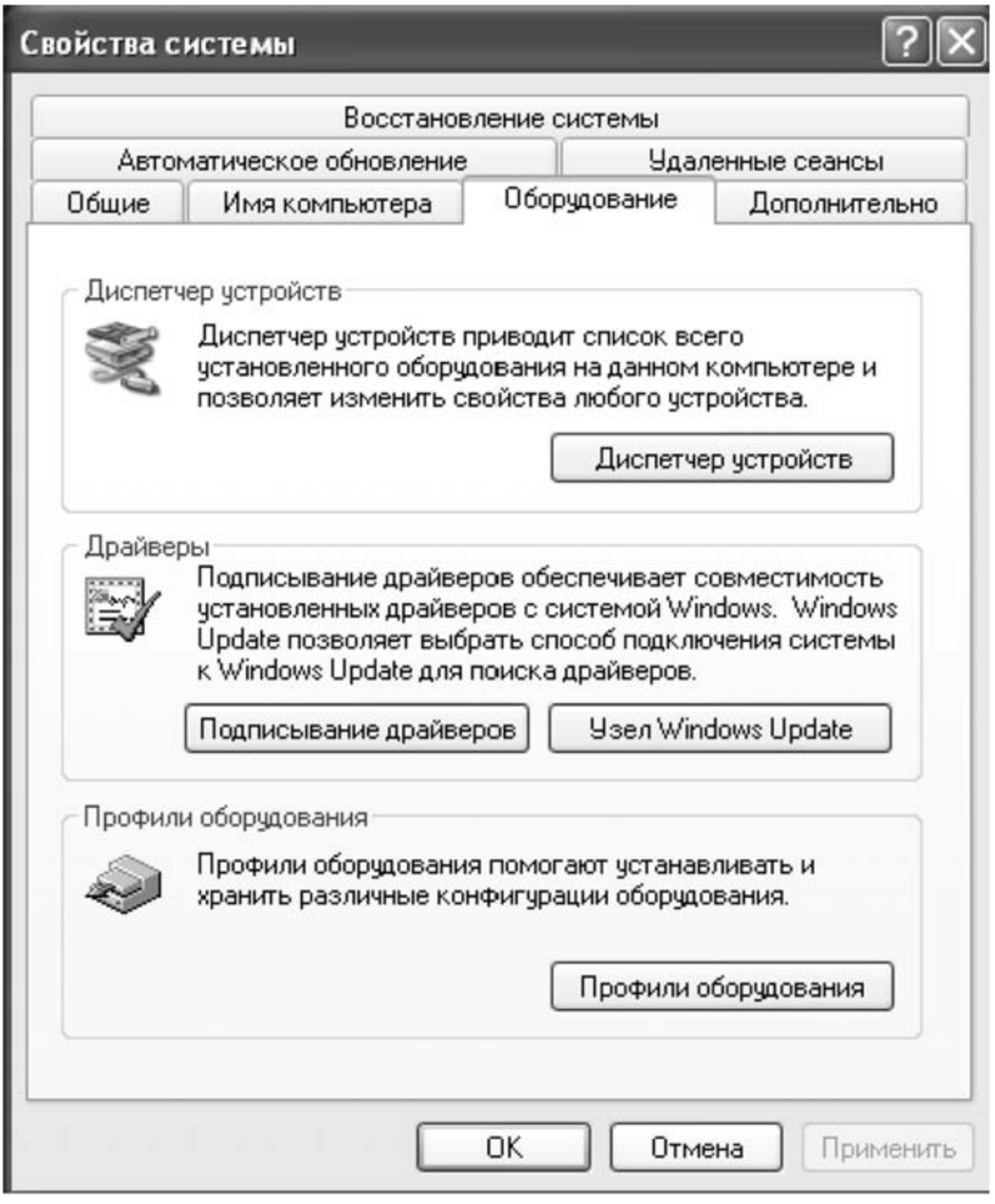


Рис. 4.6

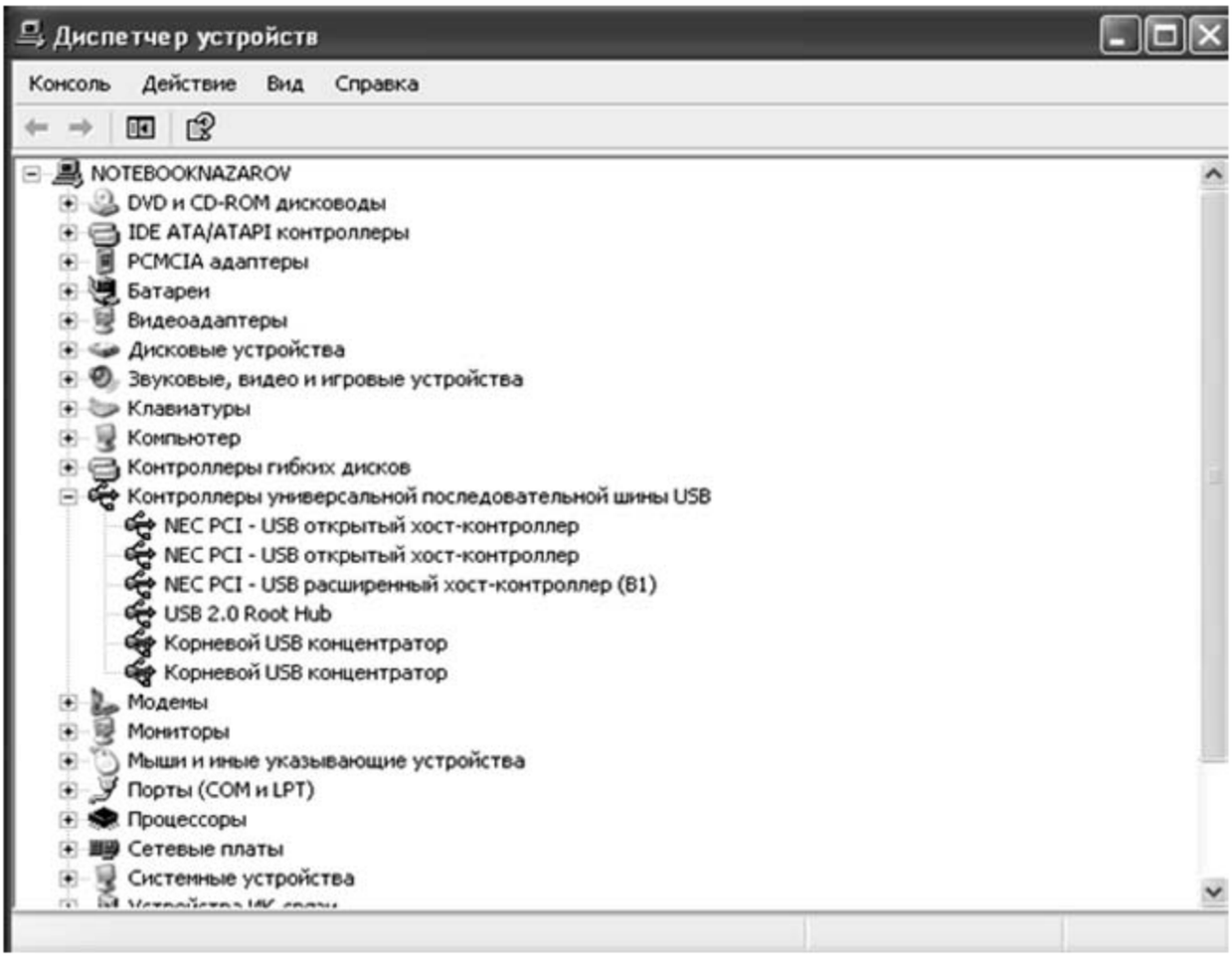


Рис. 4.7

При установке устройства Plug and Play Windows автоматически настраивает его, обеспечивая его правильную работу с другими установленными на компьютере устройствами. В ходе процесса настройки Windows назначает устанавливаемому устройству уникальный набор системных ресурсов. Эти ресурсы могут включать один или несколько из следующих параметров:

- номера строк запросов на прерывание (IRQ);
- каналы прямого доступа к памяти (DMA);
- адреса портов ввода/вывода (I/O);
- диапазоны адресов памяти.

Каждый ресурс, назначаемый устройству, должен быть уникальным. Это необходимо для правильной работы устройства. Для устройств Plug and Play Windows автоматически проверяет правильность настройки ресурсов. Для просмотра системных ресурсов, выделенных устройством (например, группе Порты COM и LPT), нужно раскрыть группу и выбрать в контекстном меню команду Свойства, например сначала для Порт принтера, а затем в открывшемся окне перейти на вкладку Ресурсы (рис. 4.8). Далее перейти на вкладку Па-

раметры порта (рис. 4.9). Обратите внимание на включенную кнопку Не использовать прерывание в любом случае (почему?). Далее можно просмотреть вкладки Общие и драйвер и выполнить аналогичные действия для Последовательного порта (рис. 4.10) и контроллера гибких дисков (рис. 4.11).

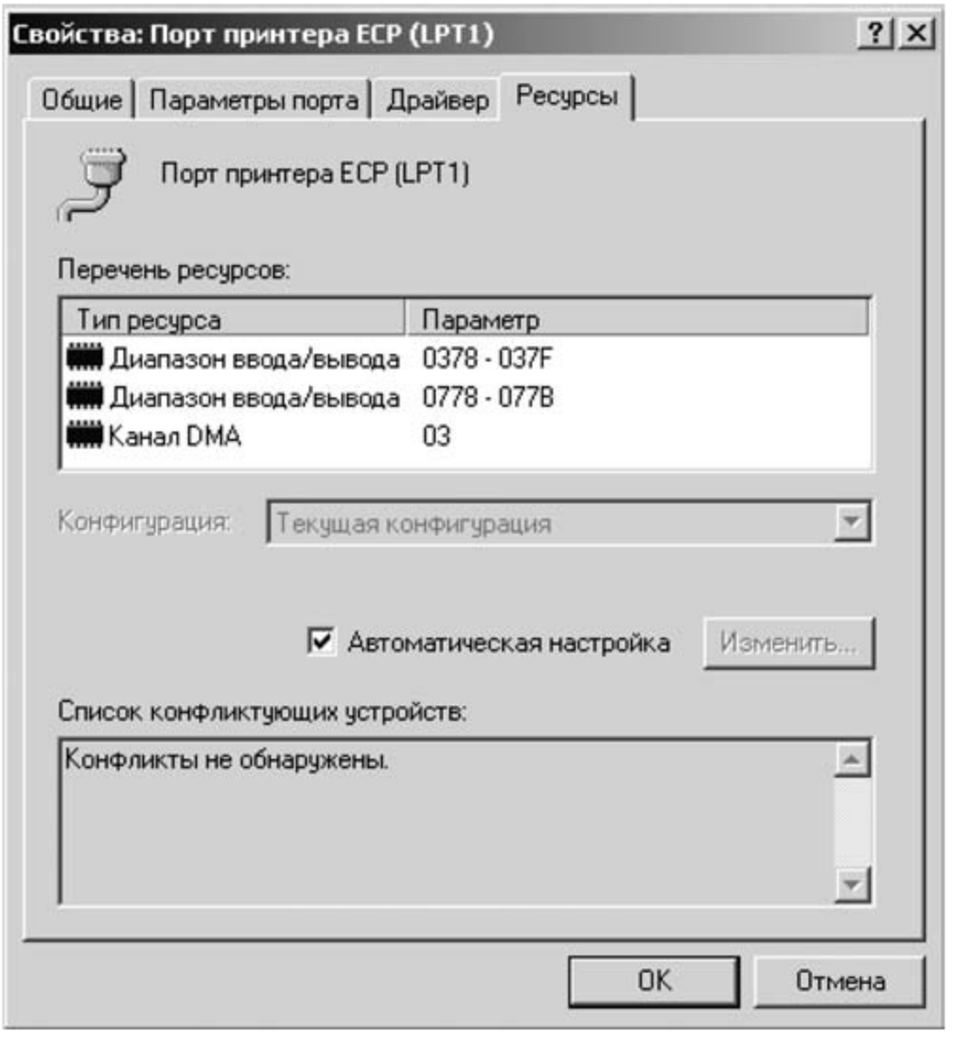


Рис. 4.8

Иногда двум устройствам требуются одинаковые ресурсы, что приводит к конфликту устройств. В этом случае необходимо вручную изменить настройку ресурсов таким образом, чтобы все параметры были уникальными. Некоторые ресурсы, например прерывания устройств PCI, могут в зависимости от драйверов и компьютера использоваться совместно. В операционной системе Windows имеется служебная программа Сведения о системе (рис. 4.12), с помощью которой можно получить исчерпывающую информацию о системных ресурсах. В том числе имеется возможность получить информацию о конфликтующих устройствах и устройствах, использующих системные ресурсы совместно (рис. 4.13).

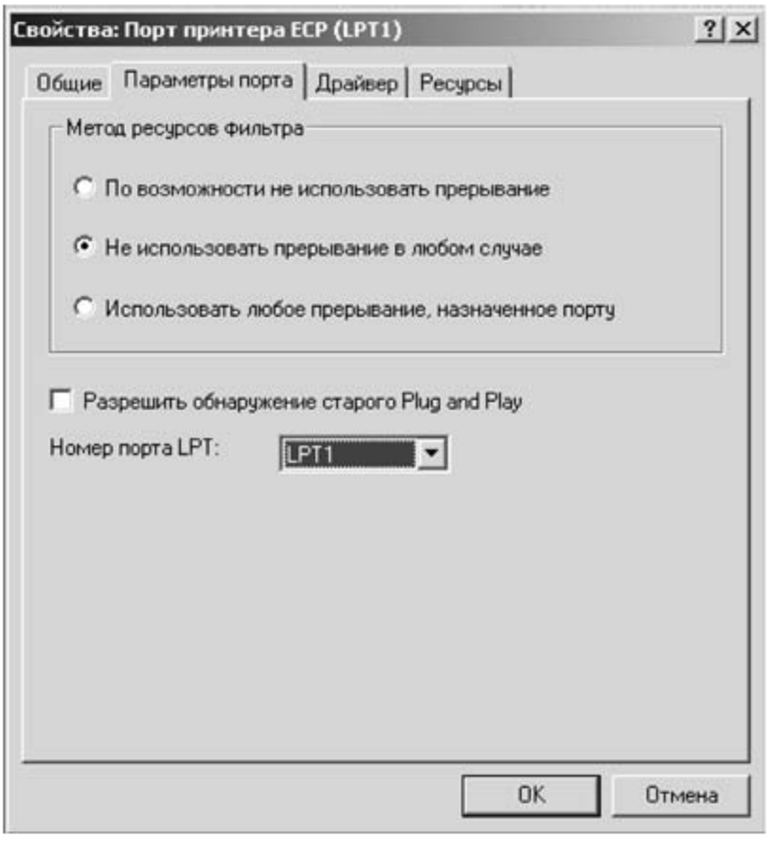


Рис. 4.9

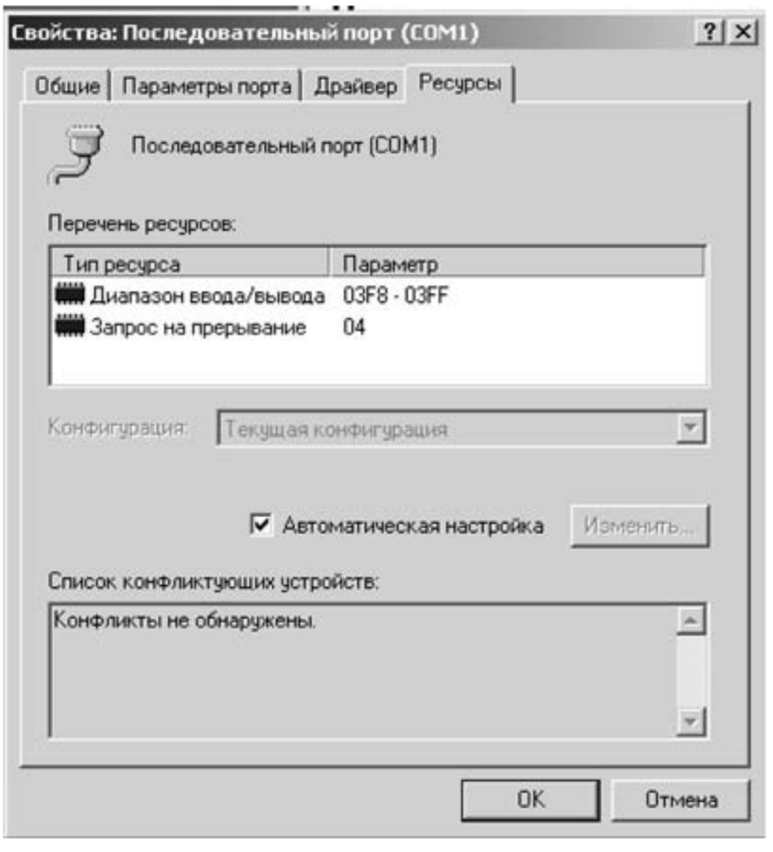


Рис. 4.10

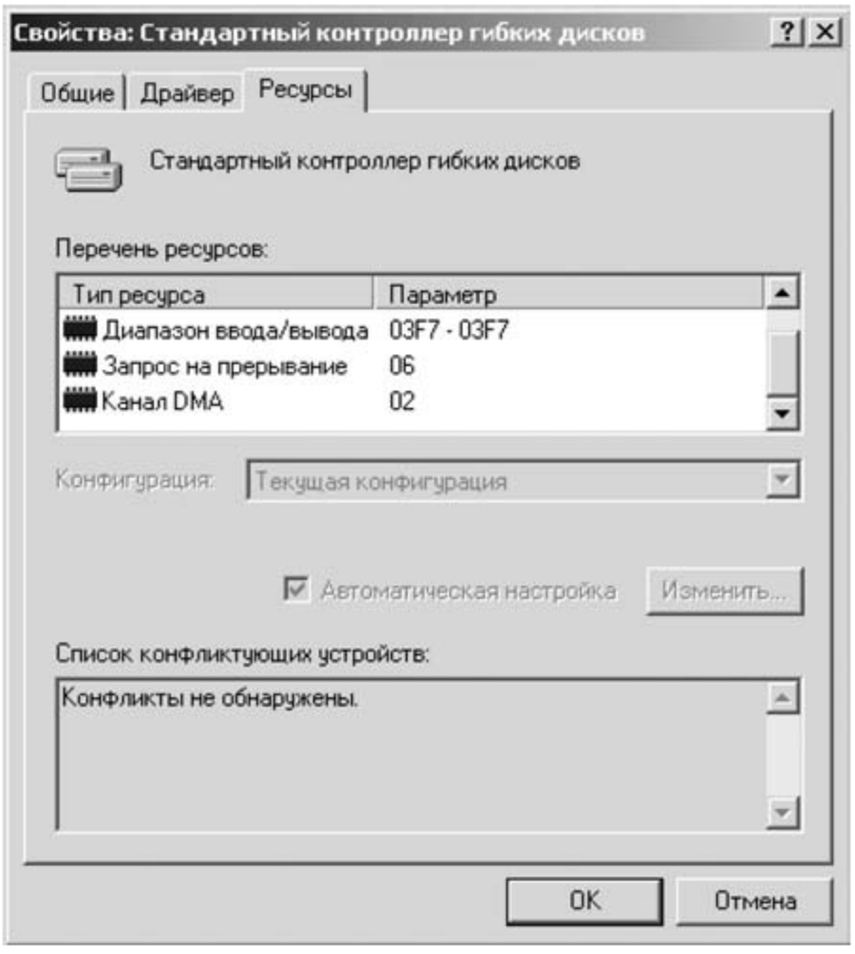


Рис. 4.11

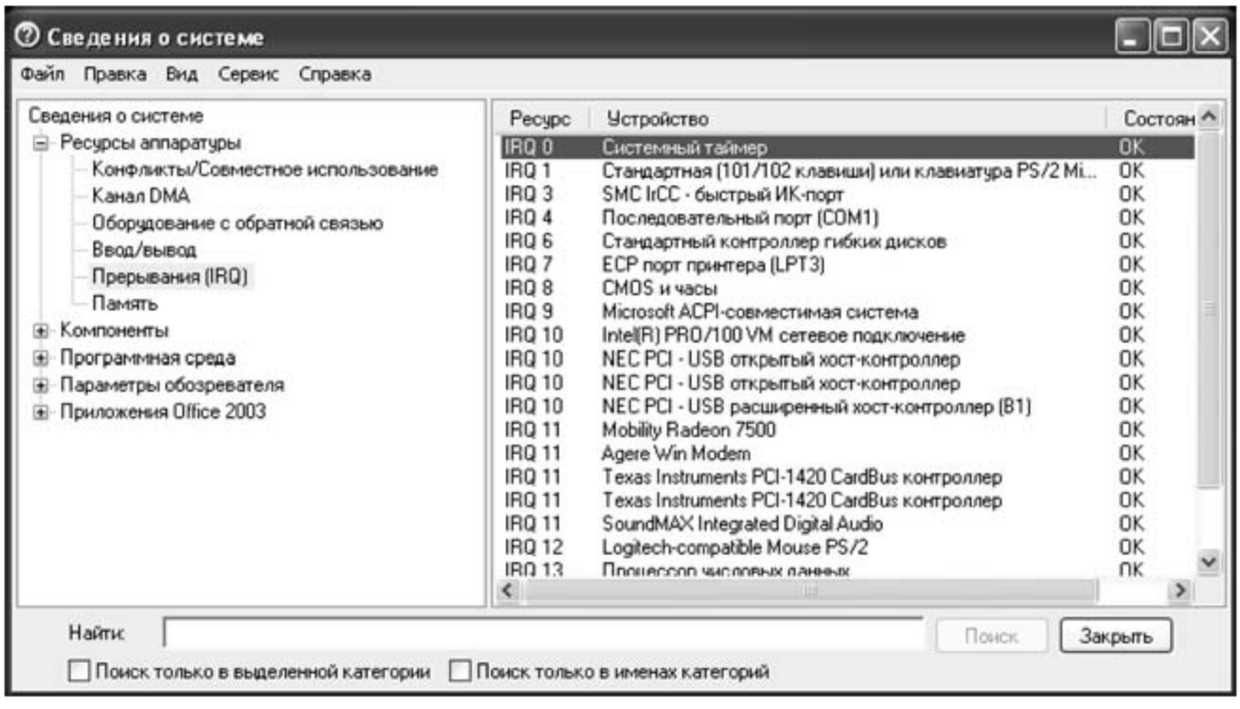


Рис. 4.12

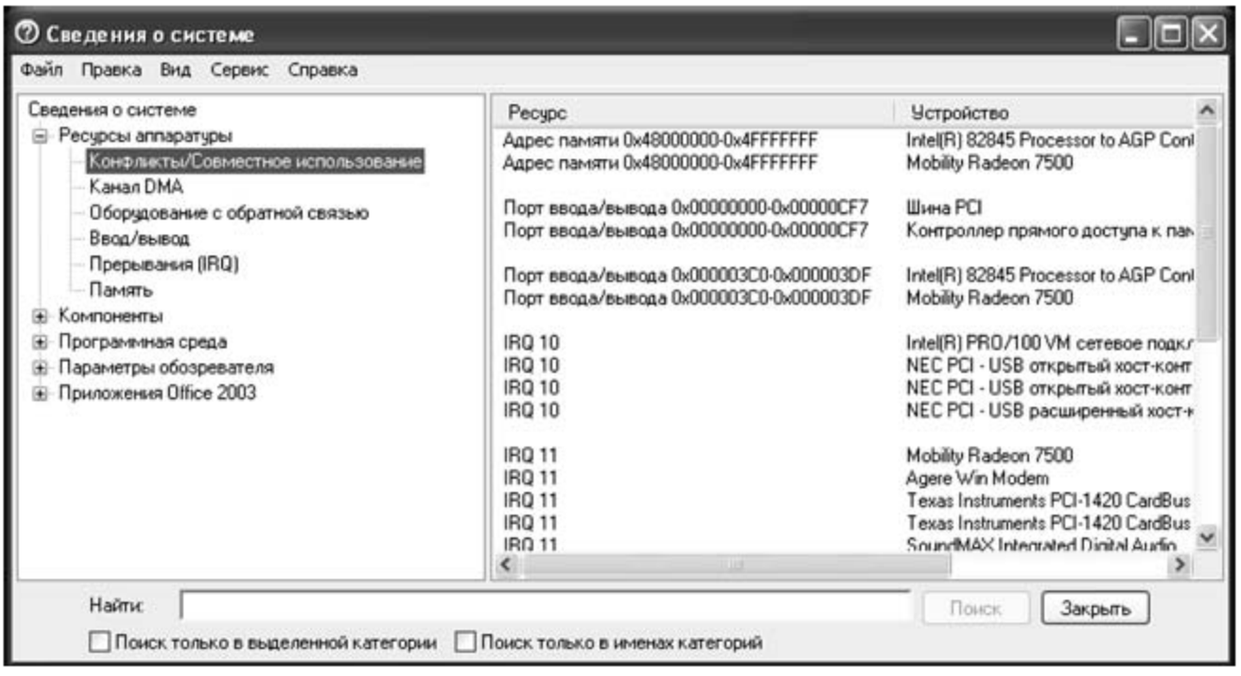


Рис. 4.13

При установке устройств, кроме устройств Plug and Play, автоматическая настройка ресурсов не производится. Некоторые типы устройств требуется настраивать вручную. Необходимые инструкции содержатся в руководстве, поставляемом вместе с устройством. Изменять параметры ресурсов вручную обычно не рекомендуется, поскольку при этом значения фиксируются, что снижает возможности Windows по выделению ресурсов для других устройств. Если зафиксировано слишком много значений параметров для отдельных ресурсов, Windows не сможет автоматически устанавливать новые устройства Plug and Play.

Для настройки устройств вручную используется Диспетчер устройств. Неправильное изменение параметров ресурсов может привести к отключению устройства или явиться причиной неправильной работы компьютера. Их следует изменять только при полной уверенности в том, что новые параметры не будут конфликтовать с другим оборудованием, или в том случае, если изготовитель данного устройства предлагает для него конкретные параметры. Обнаружить устройства с неполадками можно, раскрыв папку Компоненты и выбрав строку Устройства с неполадками (рис. 4.14).

Ряд устройств компьютера задействуют в процессах обмена информацией контроллер прямого доступа к памяти. Посмотреть, какие устройства используют этот контроллер, можно, выбрав в папке Ресурсы аппаратуры строку Канал DMA (рис. 4.15).

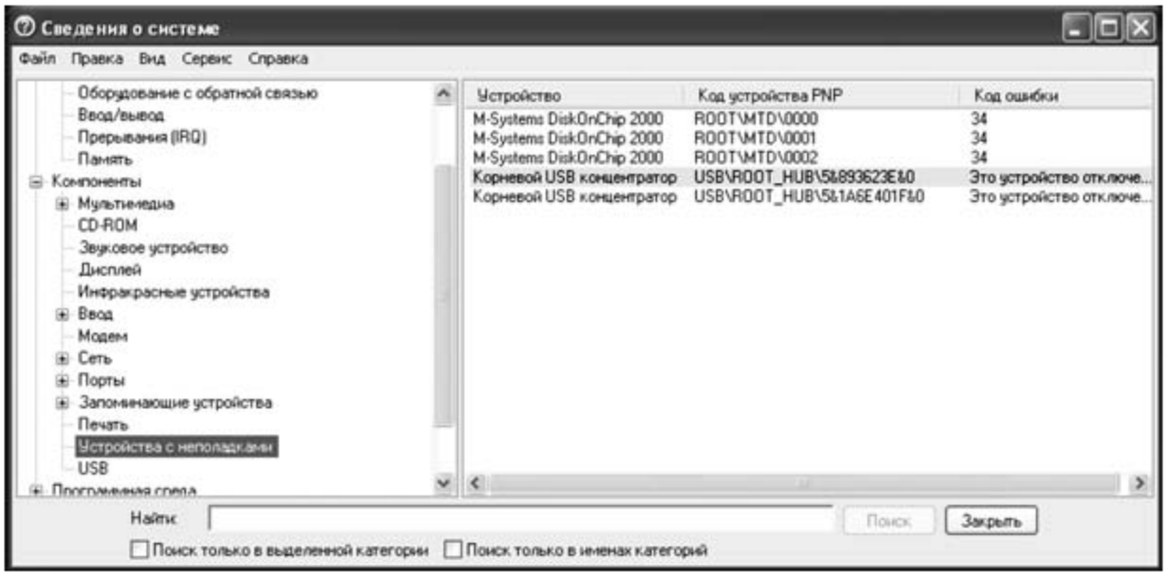


Рис. 4.14

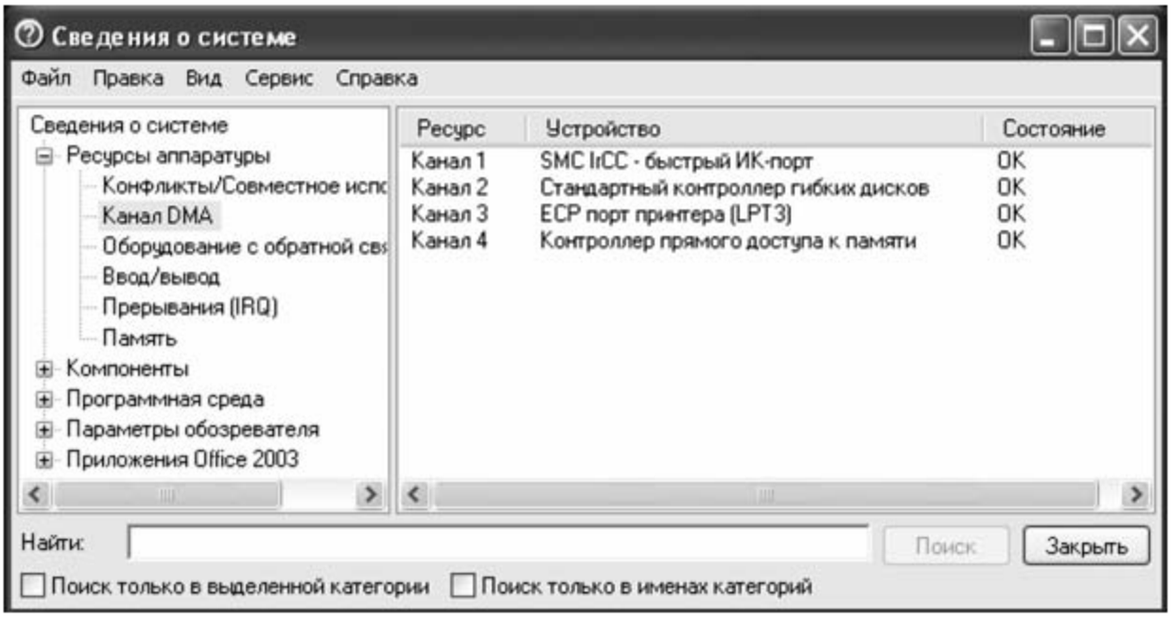


Рис. 4.15

С помощью Диспетчера устройств можно отключать подсоединенные к компьютеру устройства и удалять их из конфигурации компьютера. Для удаления устройства Plug and Play обычно достаточно его отключить или удалить из конфигурации, для удаления некоторых устройств необходимо сначала выключить компьютер. Чтобы правильно выполнить последовательность действий по удалению устройства, следует обратиться к инструкциям производителя по установке и удалению устройства. Удаление устройств, кроме устройств Plug and Play, обычно состоит из двух шагов:

1) отмены установки устройства с помощью диспетчера устройств;
2) удаления устройства из конфигурации компьютера.

Диспетчер устройств используется для уведомления системы о том, что требуется удалить устройство, не поддерживаемое Plug and Play. После уведомления системы об удалении устройства необходимо физически отключить или удалить его из компьютера. Например, если устройство подключено к внешнему порту компьютера, следует выключить компьютер, отключить устройство от порта, а затем отсоединить шнур питания от устройства.

Не обязательно удалять устройство, которое требуется отключить, не отсоединяя от компьютера (например, модем). Не отменяя установку самонастраиваемого устройства, его можно просто отключить. При отключении такого устройства оно физически остается подключенным к компьютеру, но Windows обновляет системный реестр таким образом, что драйверы отключенного устройства не загружаются при запуске компьютера. При включении устройства драйверы снова становятся доступными, это удобно при необходимости переключения между двумя устройствами, например сетевым адаптером и модемом, или при устранении неполадок в оборудовании.

Задания для самостоятельной работы

1. С помощью Диспетчера устройств определить, какие системные ресурсы используются портами COM и LPT.
2. Просматривая параметры порта LPT, обратить внимание на включенную кнопку Не использовать прерывание в любом случае. Просмотреть все вкладки. Выполнить аналогичные действия для Последовательного порта.
3. Объяснить, почему для порта принтера используется канал DMA и не используется прерывание, а для последовательного порта используется прерывание и есть возможность установки скорости работы порта.
4. Выполнить аналогичные действия для Стандартного контроллера гибких дисков и других устройств и объяснить полученные данные.
5. Объяснить, почему несколько устройств используют один и тот же номер IRQ. Как ОС их различает? Как изменяется уровень приоритета по шинам IRQ? Какие устройства используют DMA? В какие области памяти производится ввод-вывод? Почему?

4.3. Диски и файловая система

4.3.1. Дефрагментация жестких дисков и загрузочных файлов

Для получения доступа к просмотру состояния и управлению дисками нужно щелкнуть правой клавишей мыши на значке Мой компьютер, выбрать и щелкнуть на строке Управление. В открывшемся окне щелкнуть на строке Управление дисками (рис. 4.16). В правой части окна будут отображены все дисковые устройства компьютера и основные параметры их состояния.

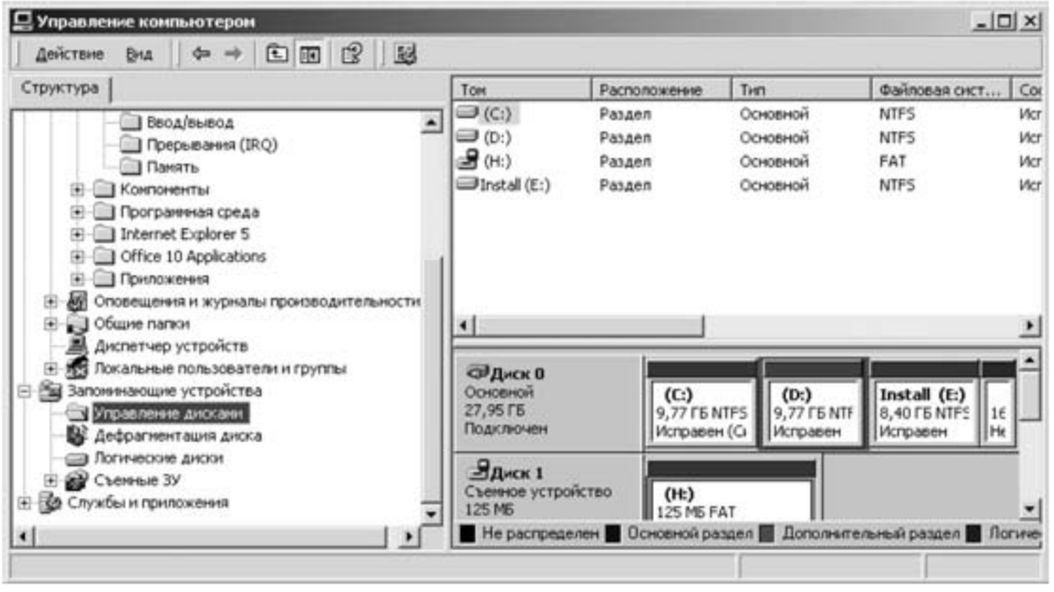


Рис. 4.16

При работе с жестким диском всегда происходит его фрагментация. С течением времени после установки программ диск заполняется. А после их удаления файлы фрагментируются, и ОС приходится искать свободные фрагменты на диске для размещения файлов. Часто файл оказывается разрезанным на сотни и тысячи маленьких фрагментов, разбросанных по всему жесткому диску. Это может привести к заметному снижению быстродействия компьютера. Негативный эффект фрагментации устраняется с помощью встроенной в Windows программы дефрагментации, запустить которую можно, указав предварительно имя диска в левой панели оснастки Управление компьютером (рис. 4.17).

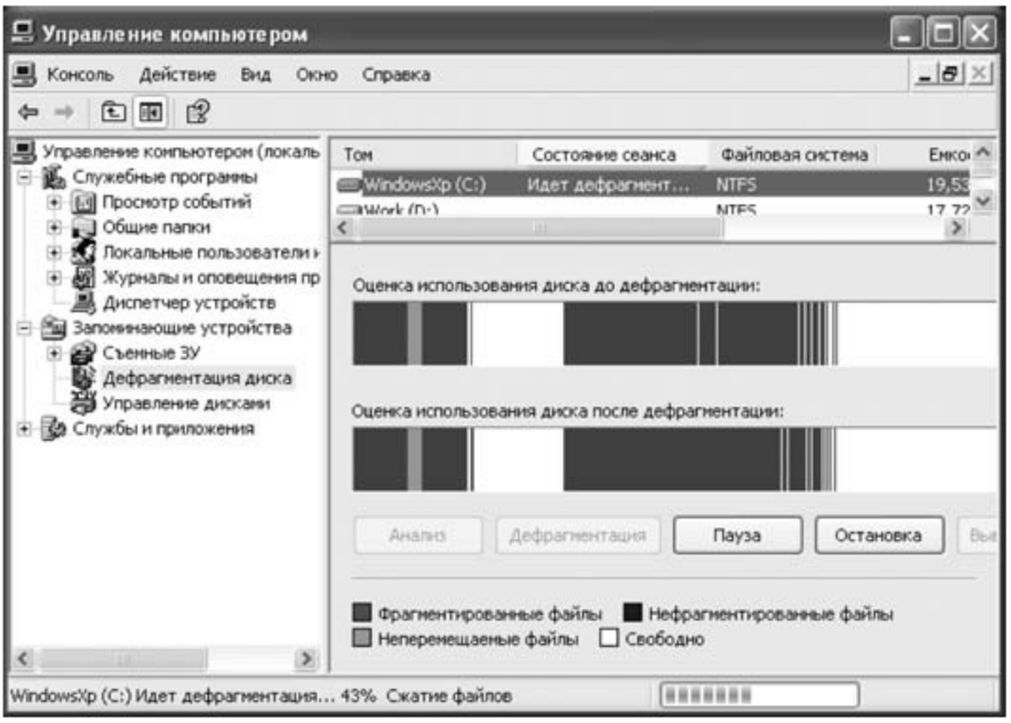


Рис. 4.17

Результаты дефрагментации можно просмотреть, нажав кнопку Вывести отчет, которая становится доступной после завершения дефрагментации (рис. 4.18).

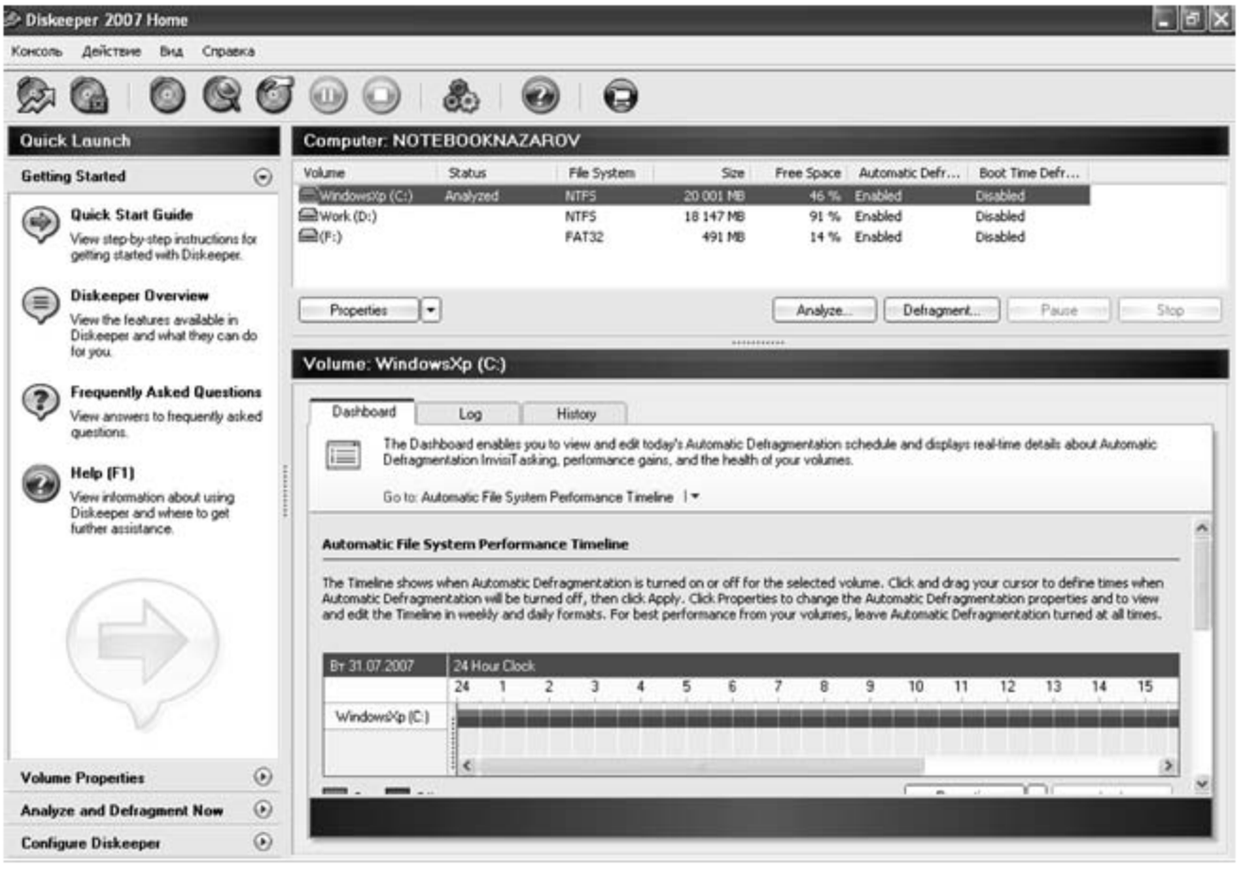


Рис. 4.18

4.3.2. Дефрагментация загрузочных файлов

Скорость чтения файлов зависит от их местоположения на диске. Кроме того, если файл фрагментирован, то для доступа к нему требуется больше времени, чем если бы части файла были расположены друг за другом. С помощью стандартных утилит Windows и программ сторонних производителей можно дефрагментировать загрузочные файлы Windows и расположить их на диске так, чтобы ускорить обращение к ним.

В операционной системе Windows XP поддерживается новый режим, в котором определяются файлы, участвующие в процессе загрузки, и затем эти файлы располагаются на жестком диске так, чтобы обеспечить оптимальную скорость загрузки. Кроме того, в Windows XP имеется программа дефрагментации загрузочных файлов, однако пользователь напрямую запустить ее не может. По умолчанию она работает только в фоновом режиме. После бездействия компьютера в течение определенного промежутка времени (5—30 мин) система считывает загрузочные данные и начинает дефрагментацию незаметно для пользователя.

Нельзя ввести команду, которая начала бы дефрагментацию загрузочных файлов. Однако можно заставить компьютер начать выполнение заданий, запускаемых при простое, даже если в это время система не загружена [13]. В результате начнется и дефрагментация загрузочных файлов. При этом будут запускаться и другие процессы, поэтому компьютеру потребуется выполнить большой объем работ. Пока не закончится выполнение всех заданий, предназначенных для запуска при простое, компьютер нельзя использовать для других ресурсоемких задач.

Для выполнения заданий, предназначенных для запуска при простое, нужно выполнить следующие действия:

- нажать кнопку Пуск, затем выбрать команду Выполнить;
- в поле ввода открывшегося окна набрать строку Rundll32.exe advapi32.dll.ProcessIdleTasks и щелкнуть на кнопке OK.

После этого компьютер начнет выполнять задания, запускаемые при простое, в частности, приведенная команда позволит системе выполнить дефрагментацию загрузочных файлов.

Встроенная в Windows XP программа дефрагментации загрузочных файлов работает довольно хорошо, но не обрабатывает файлы реестра, записи таблицы размещения файлов и некоторые другие системные файлы. В то же время файлы реестра, как и любые другие системные файлы с данными, при каждодневном использовании могут фрагментироваться. Поэтому их тоже надо иногда дефрагментировать. Имеют-

ся программы сторонних производителей, обладающие дополнительными возможностями.

Одной из программ, позволяющей получить полный доступ ко всем файлам в процессе дефрагментации, является программа Diskeeper. Условно-бесплатную копию этой программы можно получить на сайте фирмы-разработчика (www.executive.com/downloads/menu.asp).

Загрузив, установив и запустив эту программу, нужно выполнить следующие действия для выполнения дефрагментации во время загрузки:

1) выполнить обычную дефрагментацию с помощью утилиты Diskeeper, чтобы подготовить непрерывное свободное пространство на диске (рис. 4.19);

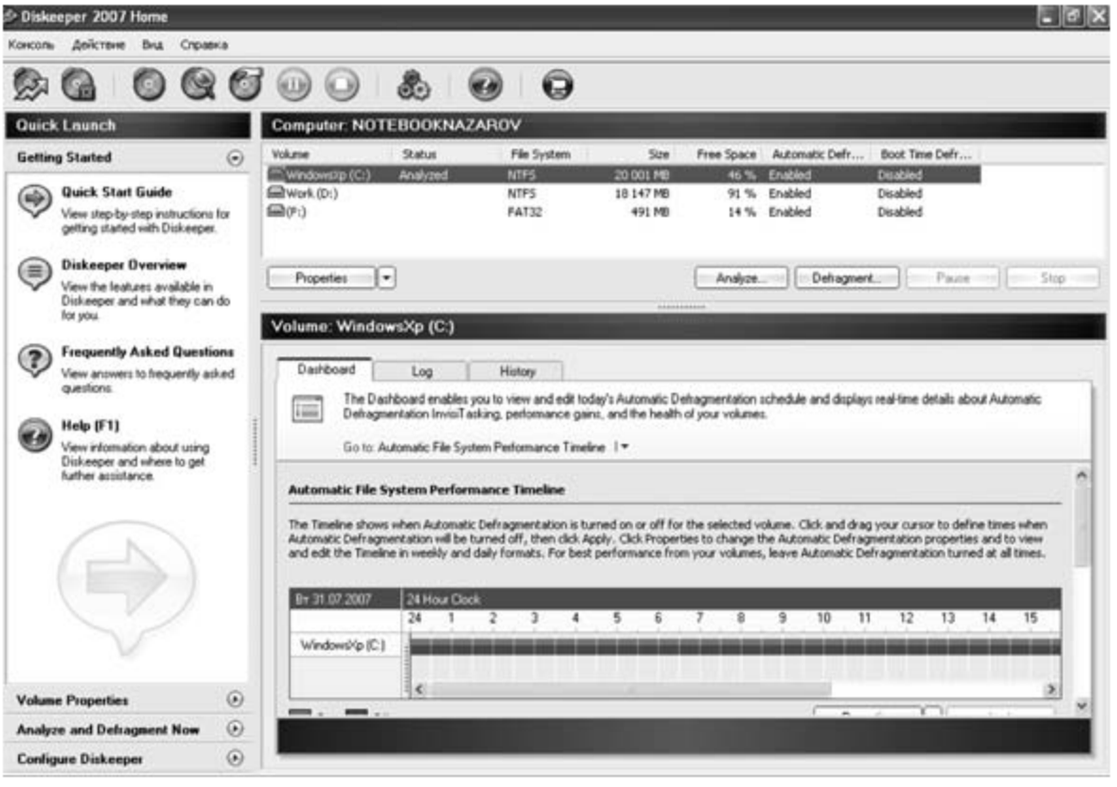


Рис. 4.19

2) после завершения дефрагментации, результат которой показан на рис. 4.20, раскрыть список Properties, выбрать строку Boot-Time Defragmentation и в новом окне выбрать загрузочный диск и задать нужные параметры дефрагментации. Как показано на рис. 4.21, следует установить следующие флажки: Enable boot-time defragmentation to run on the selected volumes (Разрешить запуск дефрагментации на выбранном томе), Defragment the paging file (Дефрагментировать файл подкачки), Defragment the Master File Table (MFT) (Дефрагментировать главную таблицу размеще-

ния файлов), Pause to view screen after defragmentation (Пауза для просмотра результатов дефрагментации) и Produce the summary log file (Создать итоговый файл журнала);

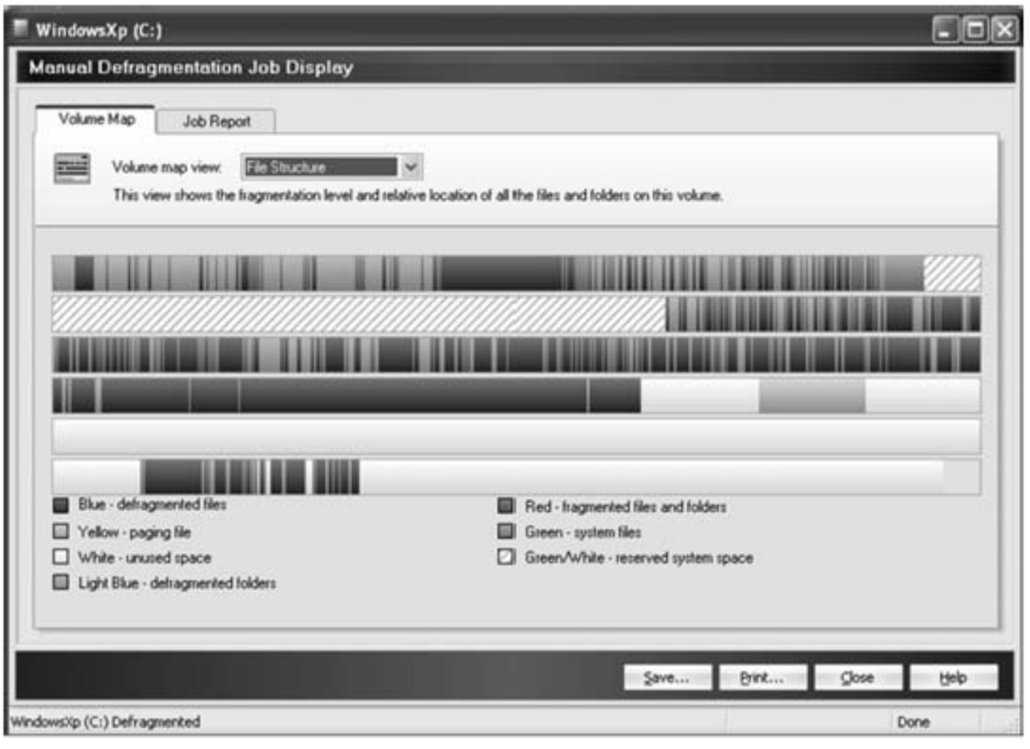


Рис. 4.20

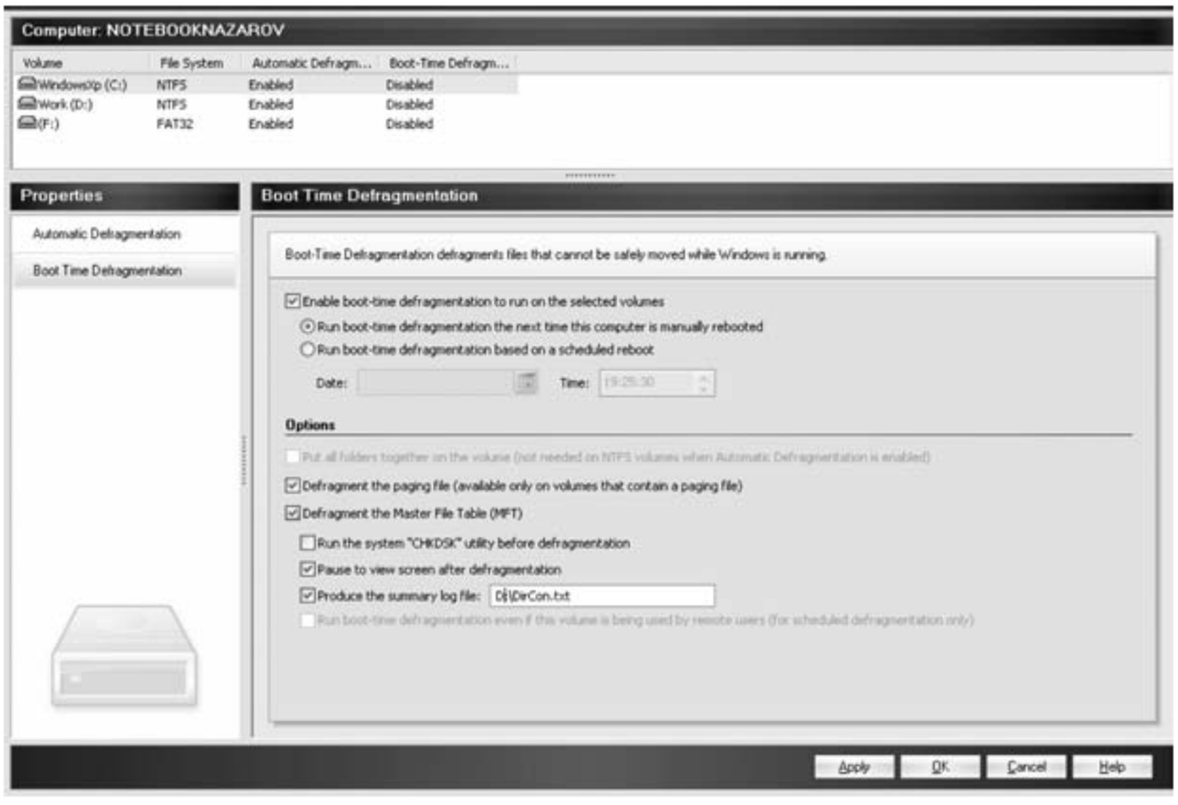


Рис. 4.21

3) установить переключатель момента времени дефрагментации в процессе загрузки в положение Run boot-time defragmentation the next time this computer is manually rebooted (Запустить дефрагментацию во время загрузки при следующей перезагрузке компьютера вручную).

После выполнения этих действий можно перезагрузить компьютер. В течение времени перезагрузки (несколько минут) утилита Diskeeper выдает на экран информацию о ходе выполнения дефрагментации. Итоговая информация фиксируется в файле, который был задан в поле флажка Produce the summary log file.

Еще одна известная утилита дефрагментации O&O Defrag разработана компанией O&O Software. Утилита позволяет выполнять дефрагментацию во время загрузки, но алгоритм ее работы отличается от алгоритмов программы Diskeeper. Что лучше, нужно определить на практике, многое зависит от конкретной системы и компьютера. Условно-бесплатную копию программы O&O Defrag можно загрузить с сайта компании O&O Software (www.oo-software.com/en/download/).

Дефрагментация с помощью утилиты O&O Defrag включает следующие действия:

1) запустить программу O&O Defrag и выбрать в меню Lobs and Report команду Add job (рис. 4.22);

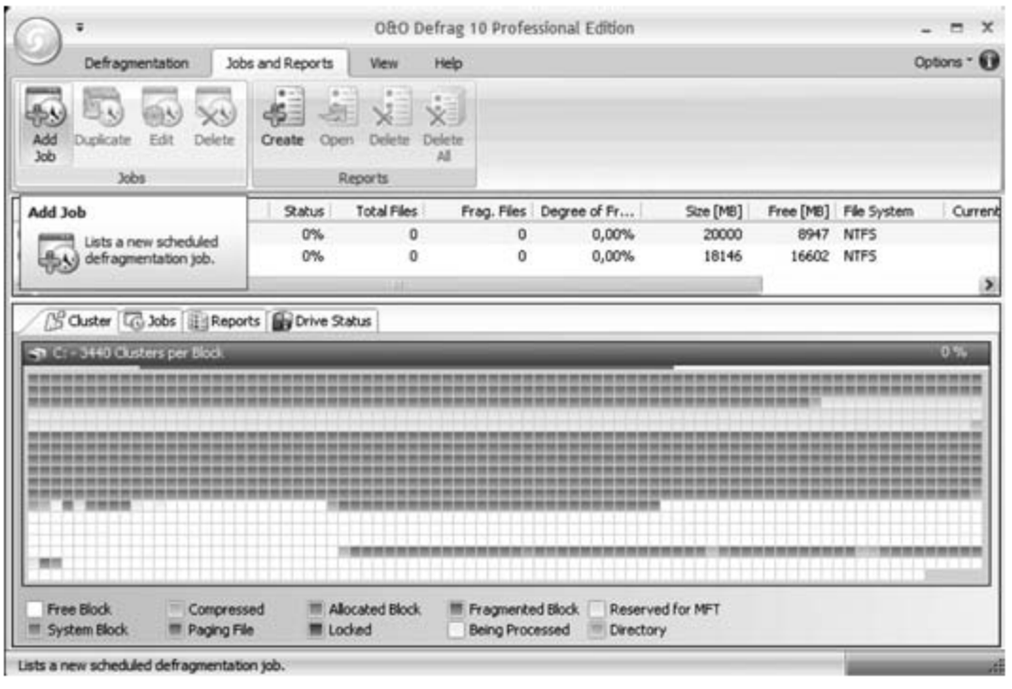


Рис. 4.22

2) в окне Edit Job перейти на вкладку Drives;
3) щелкнуть мышью на диске, предназначенном для дефрагментации, а также на диске (удерживая нажатой клавишу Shift), на котором распложены загрузочные файлы (как правило, диск C:);
4) выбрать метод дефрагментации (рекомендуется COMPLETE/Access);
5) выбрать действие, когда будет выполняться дефрагментация из раскрывающегося внизу окна списка (рис. 4.23);

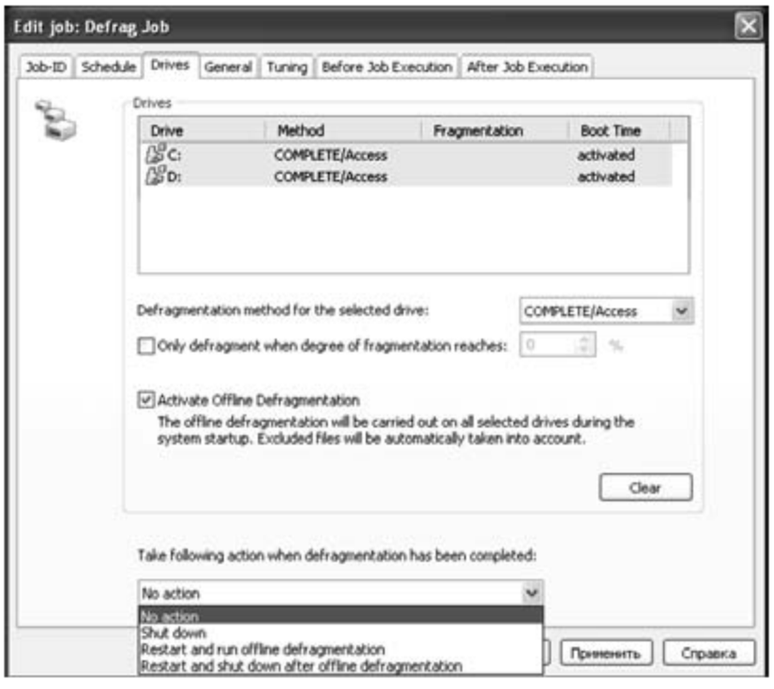


Рис. 4.23

6) перейти на вкладку Schedule и задать планируемое время дефрагментации (рис. 4.24);
7) щелкнуть на кнопке OK (созданное задание появляется последним в списке заданий). На этом подготовка к дефрагментации закончена. Она будет выполнена в запланированное время.

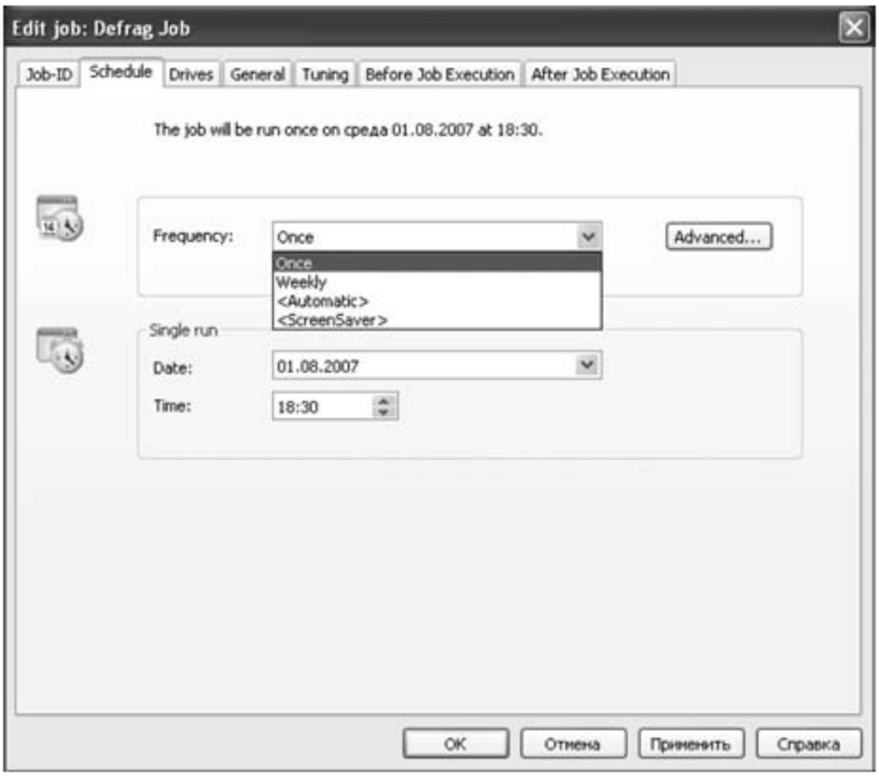


Рис. 4.24

Задания для самостоятельной работы

1. С помощью штатной программы Windows провести дефрагментацию жесткого диска компьютера.
2. Загрузить и установить утилиты Diskeeper и O&O Defrag.
3. Провести дефрагментацию во время загрузки системных и других файлов, например главной таблицы размещения файлов, поочередно обеими утилитами.
4. Сравнить полученные результаты. Можно ли сделать заключение о качестве работы той или иной утилиты?

4.3.3. Дисковые квоты

При совместном использовании дисковой памяти несколькими пользователями, работающими на одном компьютере, необходим контроль расходования дискового пространства. В Windows эта проблема решается квотированием дискового пространства по каждому тому (независимо от количества физических дисков) и для каждого пользователя.

После установки квот дискового пространства пользователь сможет хранить на томе ограниченный объем данных, в то время как на этом томе может оставаться свободное пространство. Если пользователь

превышает выданную ему квоту, в журнал событий вносится соответствующая запись. Затем в зависимости от конфигурации системы пользователь либо сможет записать информацию на том (более мягкий режим), либо ему будет отказано в записи.

Устанавливать и просматривать квоты на диске можно только в разделе NTFS 5.0 и при наличии необходимых полномочий (задаваемых с помощью локальных или доменных групповых политик) у пользователя, устанавливающего квоты.

Чтобы установить квоты нужно выполнить следующие действия:

1) щелкнуть правой кнопкой мыши на конфигурируемом томе и выбрать в контекстном меню команду Свойства. В появившемся окне перейти на вкладку Квота (рис. 4.25);
2) установить флажок Включить управление квотами. В этом случае будет установлен мягкий режим контроля используемого дискового пространства. Для задания жесткого режима контроля нужно установить флажок Не выделять место на диске при превышении квоты. На этой же вкладке устанавливается размер выделяемой квоты и порог, превышение которого вызовет запись предупреждений в журнале событий.

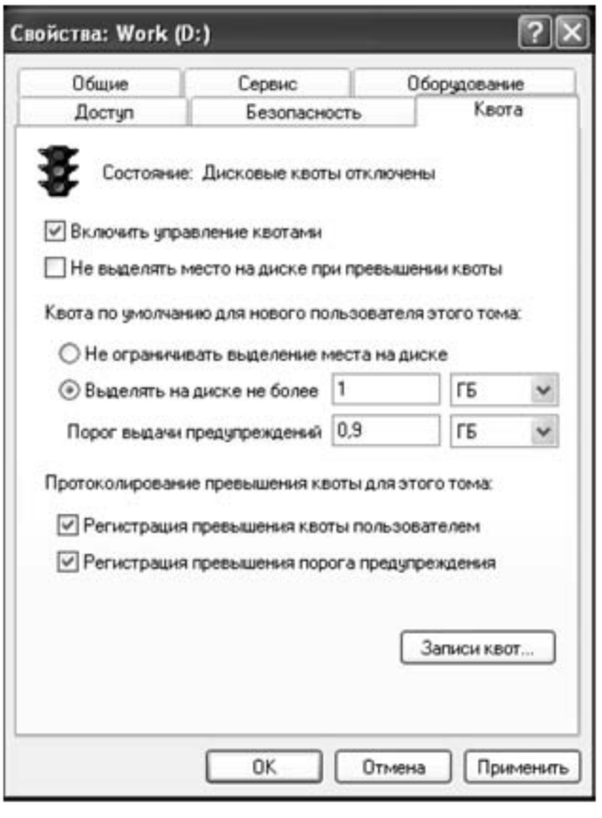


Рис. 4.25

Чтобы узнать, какие пользователи превысили выделенную им квоту (в мягком режиме), нужно нажать кнопку Записи квот (рис. 4.26), где

будет отражен список пользователей с параметрами квот и объемом используемого ими пространства диска.

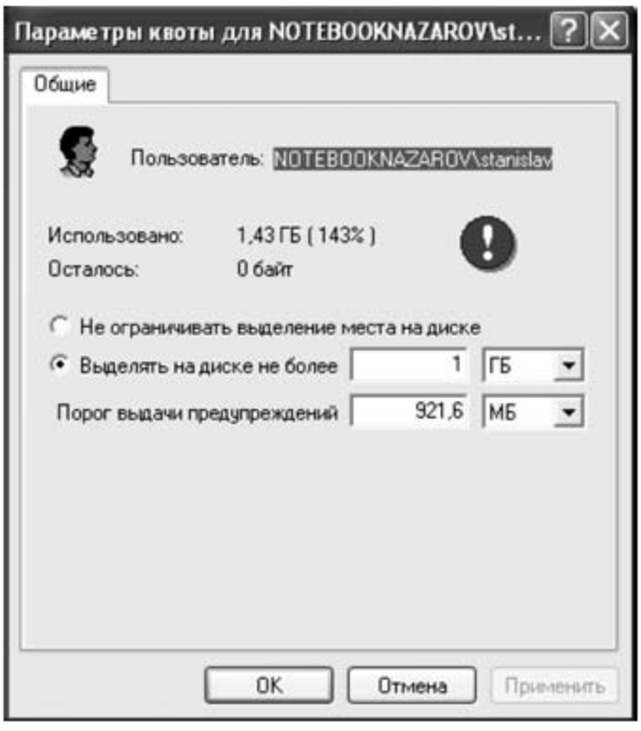


Рис. 4.26

В окне Записей квот можно изменить параметры квоты, задаваемой для конкретного пользователя. Для этого нужно выделить конфигурируемую строку и дважды щелкнуть на ней. Появится окно диалога Параметры квоты (рис. 4.27), в котором можно изменить ранее установленные параметры.

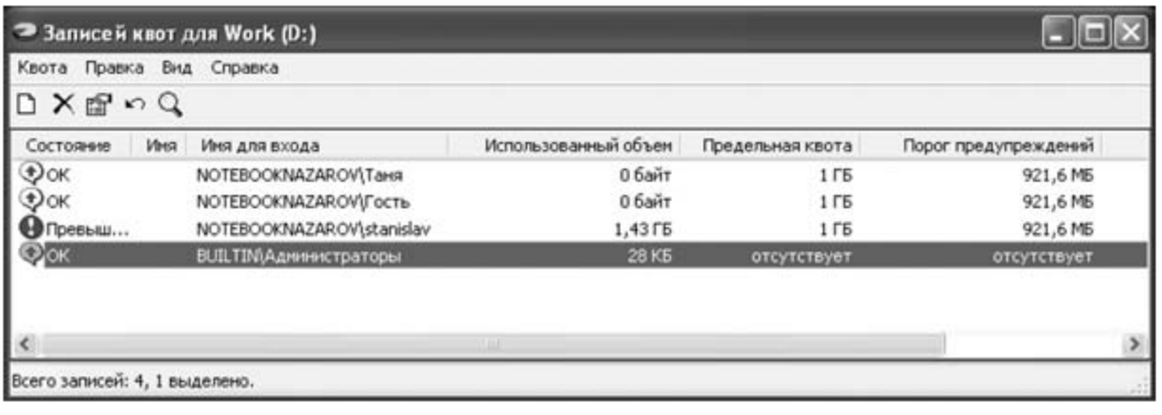


Рис. 4.27

4.3.4. Исследование алгоритмов дискового планирования

Время записи или чтения блока данных на диск определяется тремя факторами:

1) временем поиска (временем перемещения головки на нужный цилиндр;
2) задержкой вращения (временем для поворота нужного сектора под головку);
3) временем передачи данных.

Для большинства дисков первая составляющая существенно превосходит две остальные, поэтому значительного увеличения производительности системы можно добиться, снижая время поиска. Многие дисковые драйверы содержат таблицу, индексированную по номерам цилиндров, в которой в единый связный список собираются все поступившие и ждущие обработки обращения к цилиндрам диска. С помощью такой структуры данных можно создать более совершенный алгоритм планирования, чем простое обслуживание в порядке поступления запросов.

Программа HDD, приведенная на сайте www.hse.ru, позволяет моделировать различные алгоритмы обслуживания запросов к магнитному диску (рис. 4.28):

■ FCFS (First Come — First Served, первым пришел — первым обслужен);

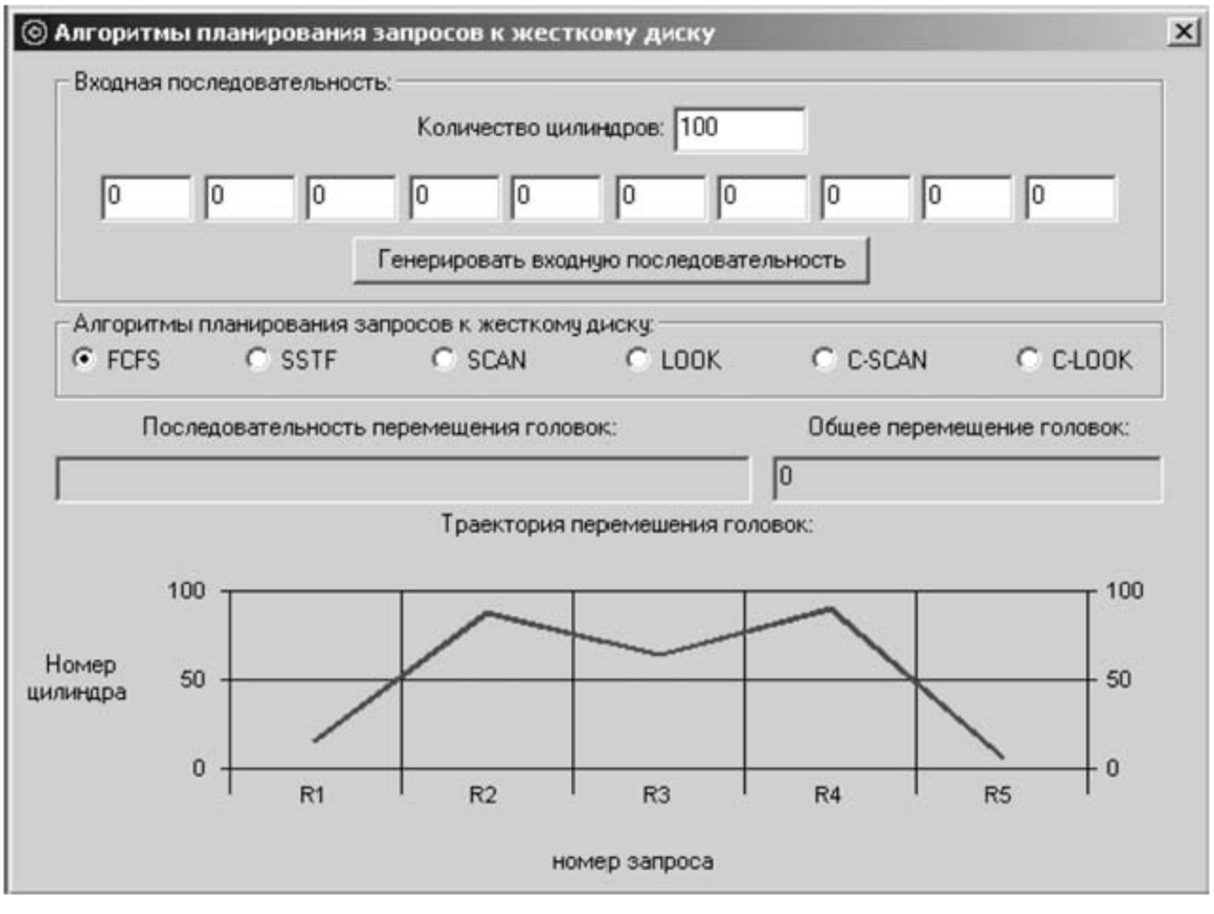


Рис. 4.28

■ SSTF (Shortest Service Time First — выбор наименьшего времени обслуживания, т.е. запроса, требующего наименьшего перемещения головок из текущей позиции);

■ SCAN (сканирование) — перемещение головки только в одном направлении с удовлетворением запросов, которые соответствуют выбранному направлению до крайних цилиндров, если их даже нет в запросах. После достижения последней дорожки направление изменяется на противоположное;

■ LOOK — аналог SCAN, но движение до крайних цилиндров, имеющихся в запросах;

■ C-SCAN (циклическое сканирование) — сканирование только в одном направлении. Когда обнаруживается последняя дорожка в заданном направлении, головка возвращается в противоположный конец диска и сканирование начинается снова;

■ C-LOOK — аналог LOOK, но циклическое сканирование.

В качестве исходных данных программа использует количество цилиндров диска, номера цилиндров запросов генерируются случайным образом. Результатом моделирования является общее число цилиндров, по которым перемещались головки.

На сайте www.hse.ru приведена, возможно, более интересная программа DiskPlan, которая также позволяет провести сравнительный анализ алгоритмов дискового планирования (рис. 4.29).

Задания для самостоятельной работы

1. В оснастке Управление компьютером просмотреть папку Управление дисками. Выбрать диск C: и просмотреть его свойства, раскрыв поочередно все вкладки окна Свойства. Рассмотреть все возможности, предоставляемые вкладками окна Свойства. Проверить, установлены ли квоты на использование диска. Установить квоты для части пользователей. Проверить наличие записи о нарушении квоты в журнале событий.
2. Запустить программу HDD или DiskPlan и провести исследование эффективности алгоритмов дискового планирования для значений количества цилиндров диска от 500 до 4000 (с шагом = 500). Сделать выводы о лучшем алгоритме для дисков с малым и большим количеством цилиндров.

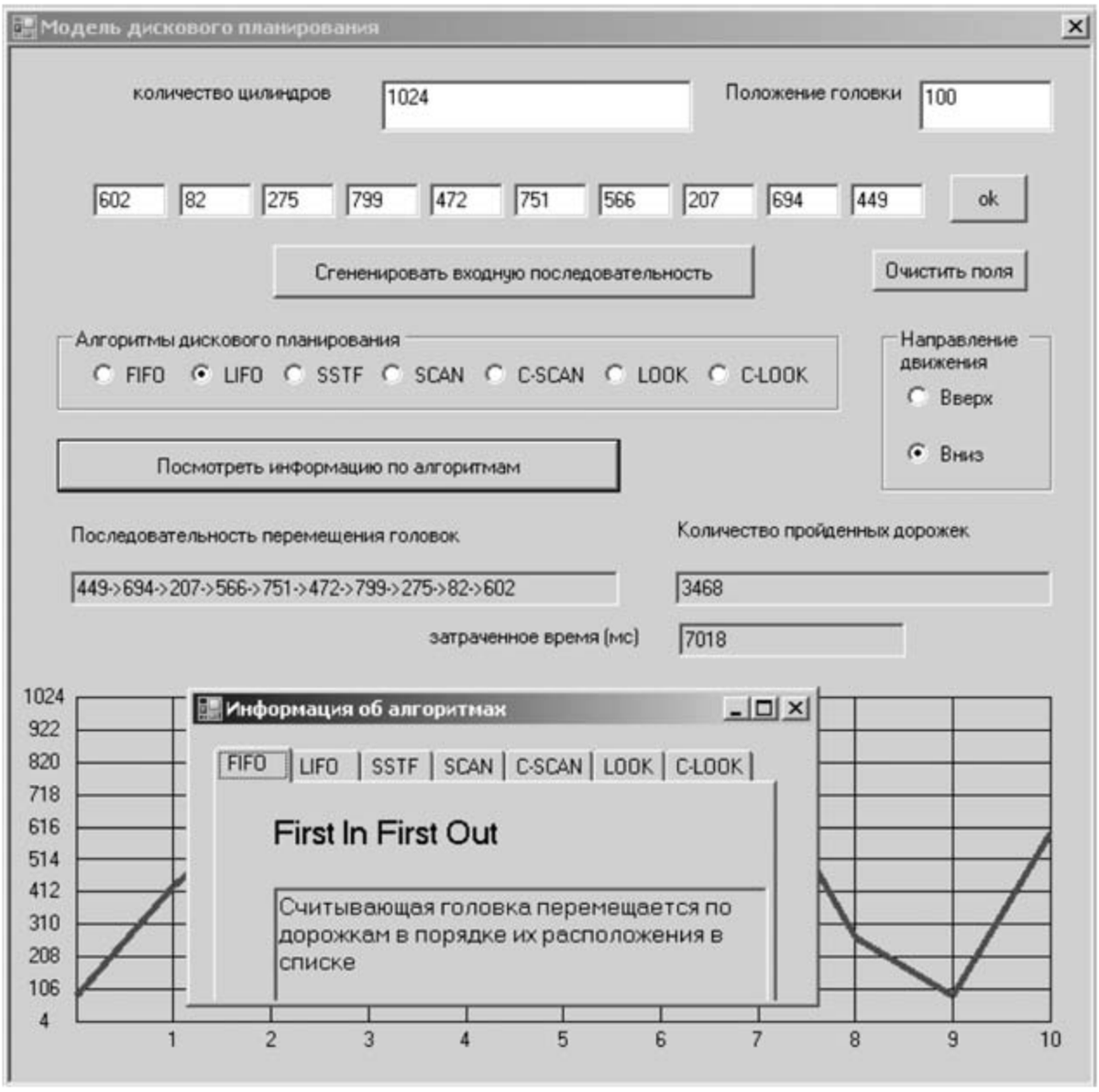


Рис. 4.29

4.4. Возможности файловой системы NTFS 5.0 по безопасности и надежности хранения данных на дисковых накопителях

4.4.1. Назначение разрешений для файлов

Устанавливая пользователям определенные разрешения для файлов и каталогов (папок), администраторы системы могут защищать конфиденциальную информацию от несанкционированного доступа [7]. Каждый пользователь имеет определенный набор разрешений на доступ к конкретному объекту файловой системы. Администратор

может назначить себя владельцем любого объекта файловой системы (обратная передача владения невозможна).

Разрешения пользователя на доступ к объектам файловой системы работают по принципу аддитивности. Это значит, что действующие разрешения в отношении конкретного файла или каталога образуются из всех прямых и косвенных разрешений, назначенных пользователю для данного объекта с помощью логической функции ИЛИ.

Для назначения пользователю или группе разрешения на доступ к некоторому файлу нужно выполнить следующие действия:

1) щелкнуть по файлу правой кнопкой мыши, выбрать в контекстном меню команду Свойства и в открывшемся окне перейти на вкладку Безопасность (рис. 4.30);

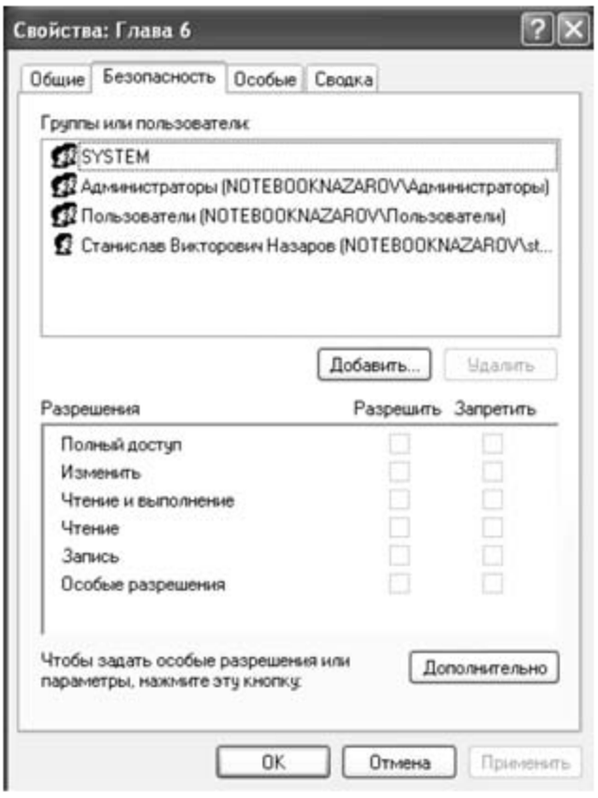


Рис. 4.30

2) в верхней части окна (Группы и пользователи) отображен список пользователей и групп, которым уже предоставлены разрешения для данного файла. Для добавления или удаления пользователя нужно нажать кнопку Добавить или Удалить;
3) в новом окне (рис. 4.31) ввести имя нужного объекта (в нашем случае пользователя), проверить это имя, нажав кнопку Проверить имена, нажать кнопку Добавить, а затем OK, чтобы вернуться на вкладку Безопасность (см. рис. 4.30);

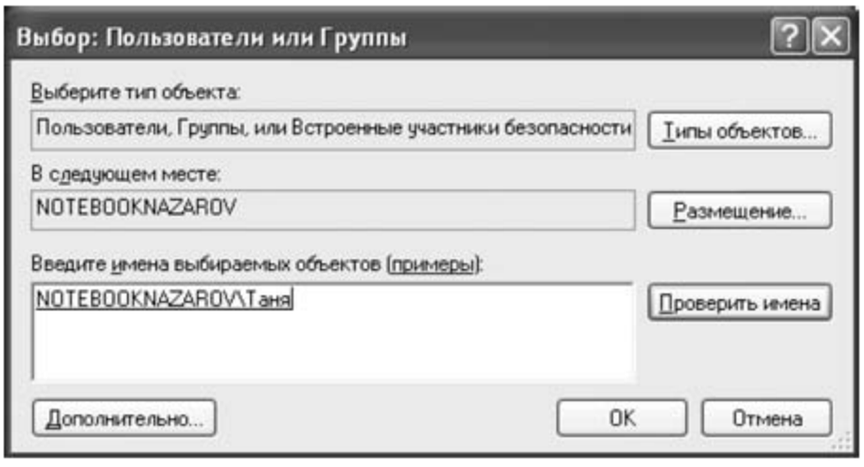


Рис. 4.31

4) теперь можно назначить или запретить стандартные разрешения для файлов. Если нужно назначить разрешения более детально, по видам возможных действий, то следует нажать кнопку Дополнительно, после чего в новом окне (рис. 4.32) нажать кнопку Изменить и в появившемся окне выбрать нужные разрешения (рис. 4.33). Дважды нажать кнопку OK (возвращаясь по окнам), а затем Применить и OK.

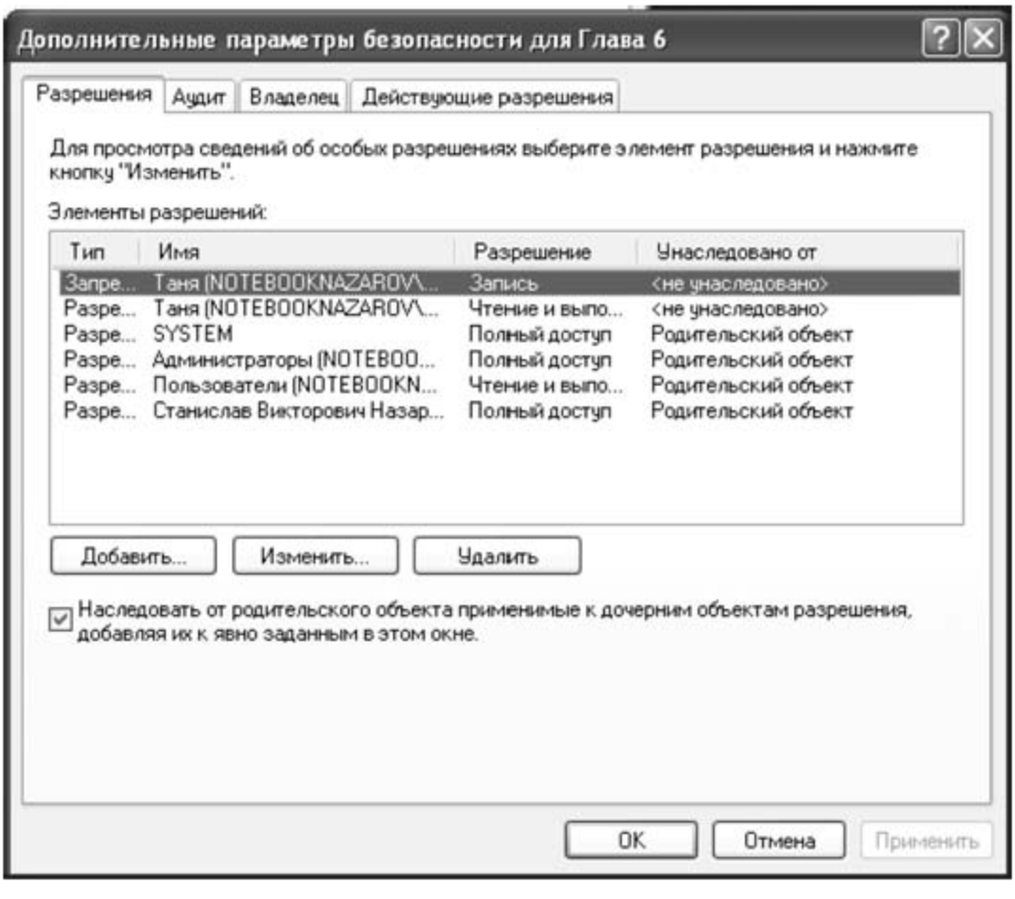


Рис. 4.32

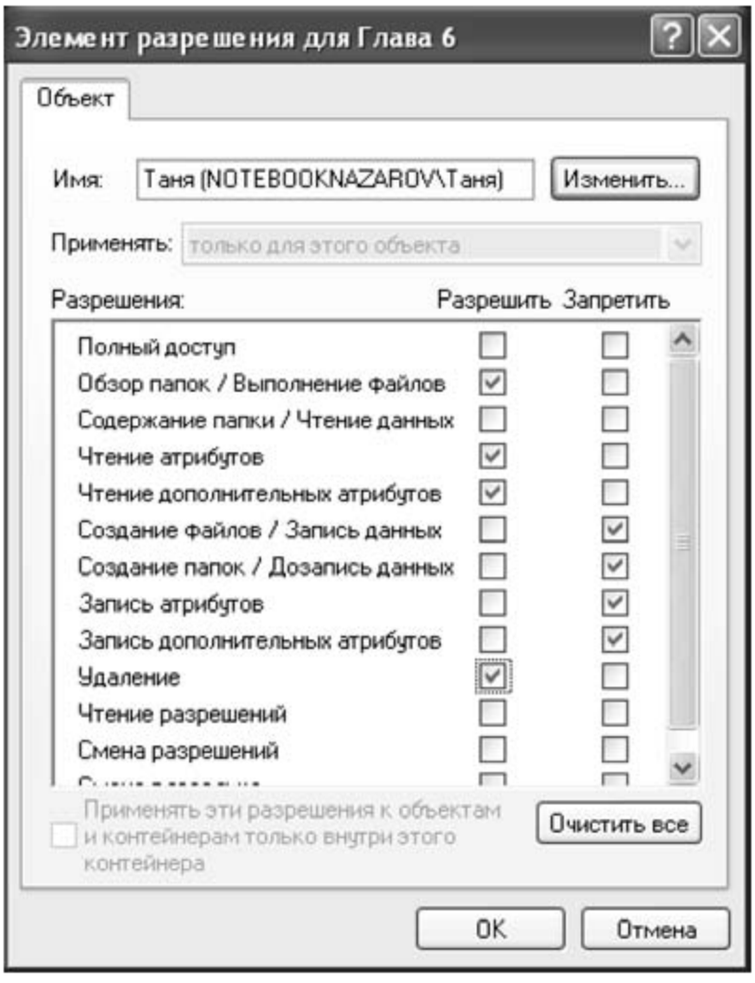


Рис. 4.33

4.4.2. Назначение разрешений для папок

Последовательность действий в этом случае практически не отличается от назначения разрешений для файлов. Для назначения разрешения на доступ к каталогу (папке) пользователю или группе нужно:

1) выбрать каталог и нажать правую кнопку мыши. В контекстном меню выбрать команду Свойства, после чего в появившемся окне (рис. 4.34) перейти на вкладку Безопасность;
2) как и в случае разрешений для файлов, предоставление разрешений пользователям и группам выполняется с помощью кнопок Добавить и Удалить. Список стандартных разрешений в группе Разрешения для каталога несколько отличается от набора разрешений для файла. Кроме того, нужно помнить, что разрешения для каталогов распространяются на находящиеся в них файлы;

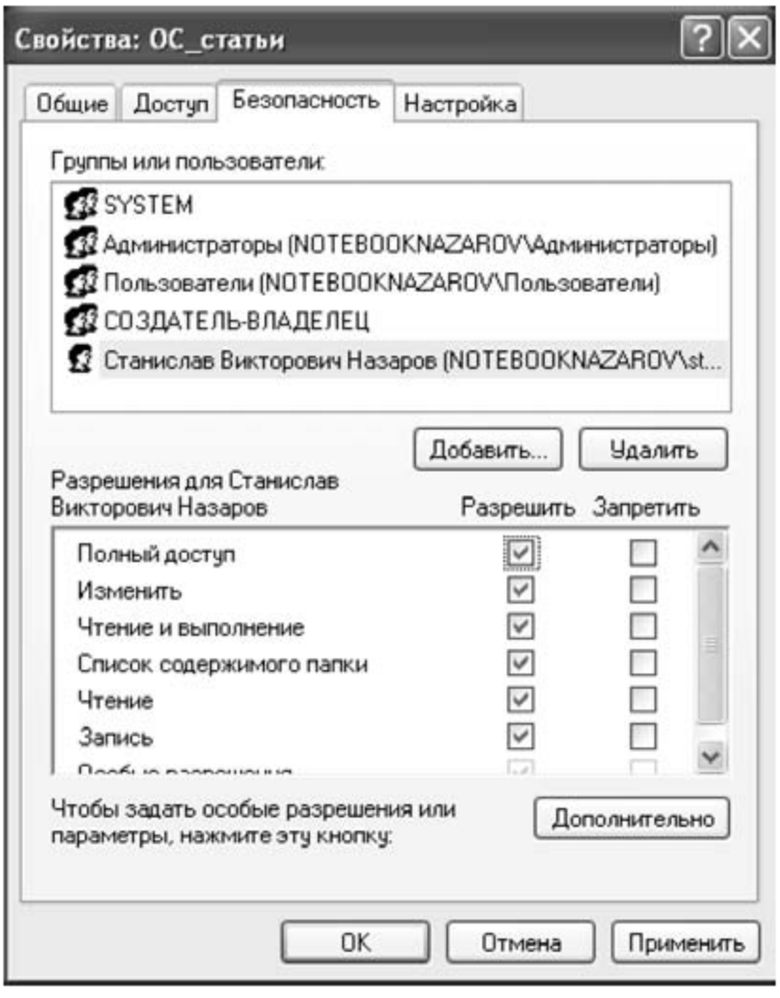


Рис. 4.34

3) в группе Разрешения можно назначить или запретить стандартные разрешения для каталогов, аналогичные разрешениям для файлов. Чтобы задать особые разрешения, нужно нажать кнопку Дополнительно;
4) в новом окне (рис. 4.35) можно установить два варианта наследования разрешений. По умолчанию каталоги наследуют разрешения пользователя на родительский каталог. Если наследование разрешений необходимо запретить, следует снять флажок Наследовать от родительского объекта применимые к дочерним объектам разрешения, добавляя их к явно заданным в этом окне;
5) после снятия значка появляется окно (рис. 4.36), в котором можно выбрать желаемый вид наследования;
6) чтобы выполнить более тонкую настройку доступа к папке, нужно нажать кнопку Изменить (см. рис. 4.35) и в появившемся окне (рис. 4.37) установить требуемые разрешения.

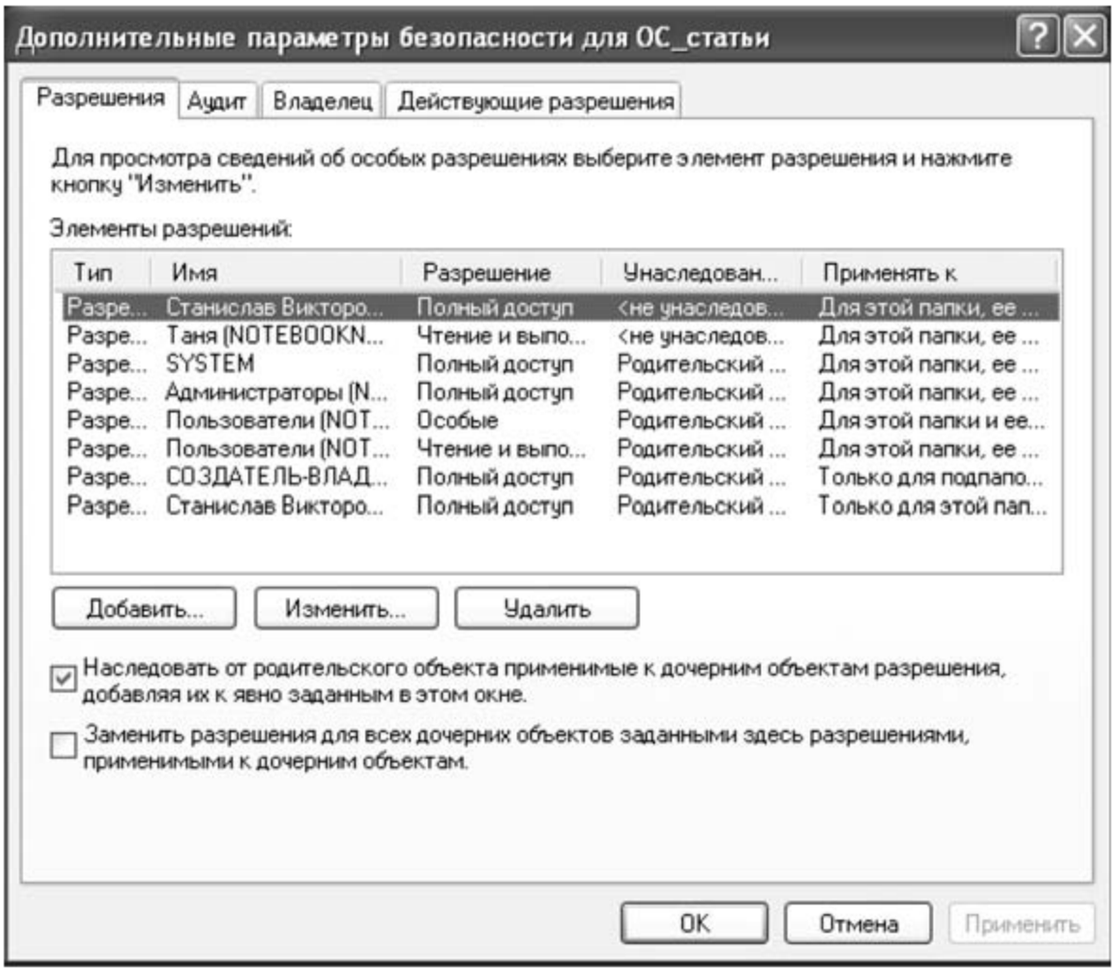


Рис. 4.35

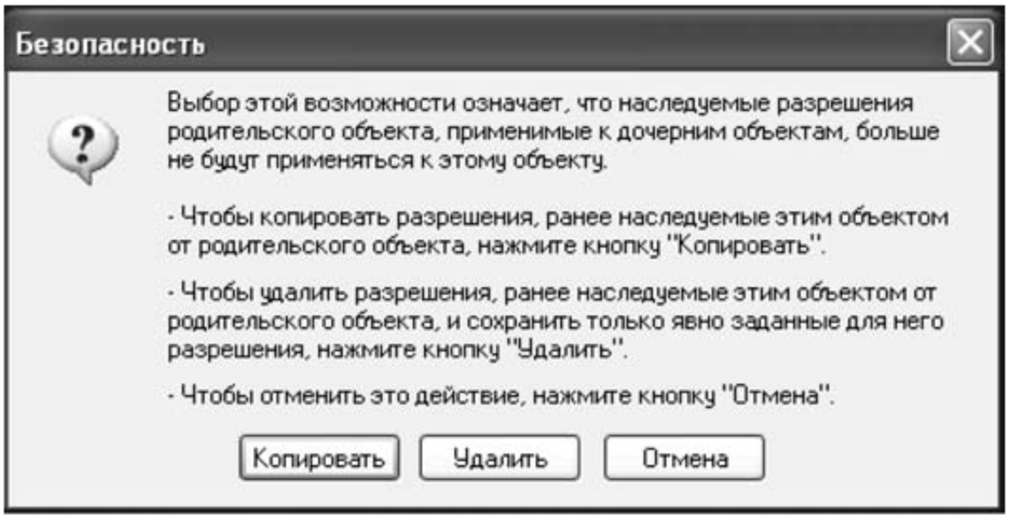


Рис. 4.36

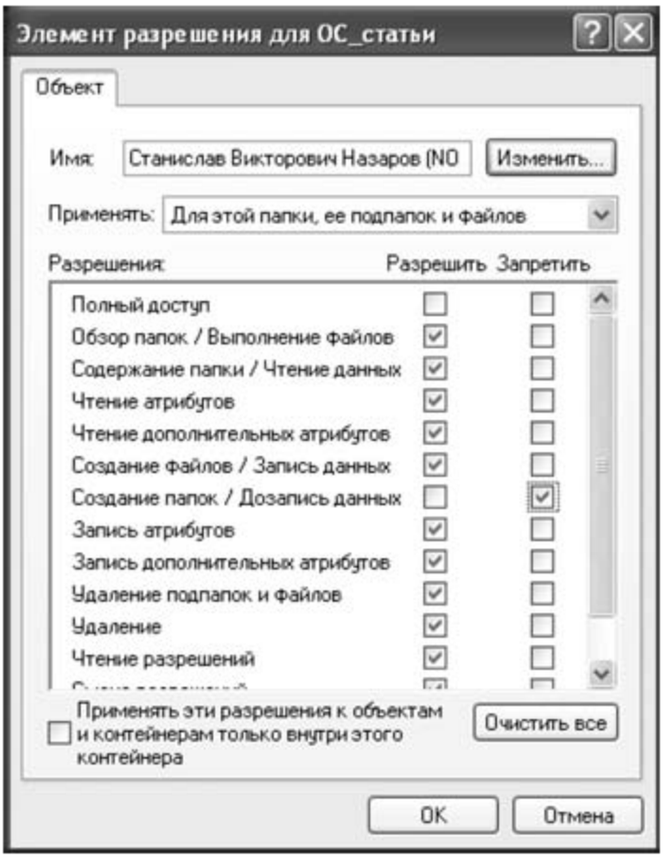


Рис. 4.37

4.4.3. Передача права владения

Пользователь может назначить себя владельцем какого-либо объекта файловой системы, если у него есть необходимые права. Для передачи владения или просмотра текущего владельца файла (папки), нужно открыть окно Свойства, перейти на вкладку Безопасность (см. рис. 4.34) и нажать кнопку Дополнительно. В появившемся окне перейти на вкладку Владелец. Текущий владелец виден в поле Текущий владелец этого элемента. В списке Изменить владельца на перечислены пользователи, имеющие право получения владения данным объектом. Выбрать нужного пользователя и нажать кнопку Применить и затем OK.

4.4.4. Точки соединения NTFS

Точки соединения (аналог монтирования в UNIX) позволяют отображать целевую папку (диск) в пустую папку, находящуюся в пространстве имен файловой системы NTFS 5.0 локального компьютера. Целевой папкой может служить любой допустимый путь Windows 2000. Точки соединений (поддерживаются только в NTFS 5.0) прозрачны

для приложений, это означает, что приложение или пользователь, осуществляющий доступ к локальной папке NTFS, автоматически перенаправляется к другой папке.

Точки соединения NTFS отличаются от точек соединения распределенной файловой системы DFS. Последние отображают общий ресурс сети [7], управляемый DFS. Таким ресурсом может быть любой допустимый общий ресурс сети. Однако оба средства служат для создания общего пространства имен хранения информации. Сравнение точек соединения DFS и NTFS и их свойств рассмотрено в работе [17].

Для работы с точками соединения на уровне томов можно использовать стандартные средства системы — утилиту Mountvol.exe и оснастку Управление дисками. Для монтирования папок нужна утилита Linkd (из Windows 2000 Resource Kit).

С помощью утилиты Mountvol можно выполнить следующие действия:

- отобразить корневую папку локального тома в некоторую целевую папку NTFS, т.е. подключить или монтировать том;
- вывести на экран информацию о целевой папке точки соединения NTFS, использованной при подключении тома;
- просмотреть список доступных для использования томов файловой системы;
- уничтожить точки подключения томов.

Параметры утилиты Mountvol можно получить, введя в командной строке ее имя (рис. 4.38).

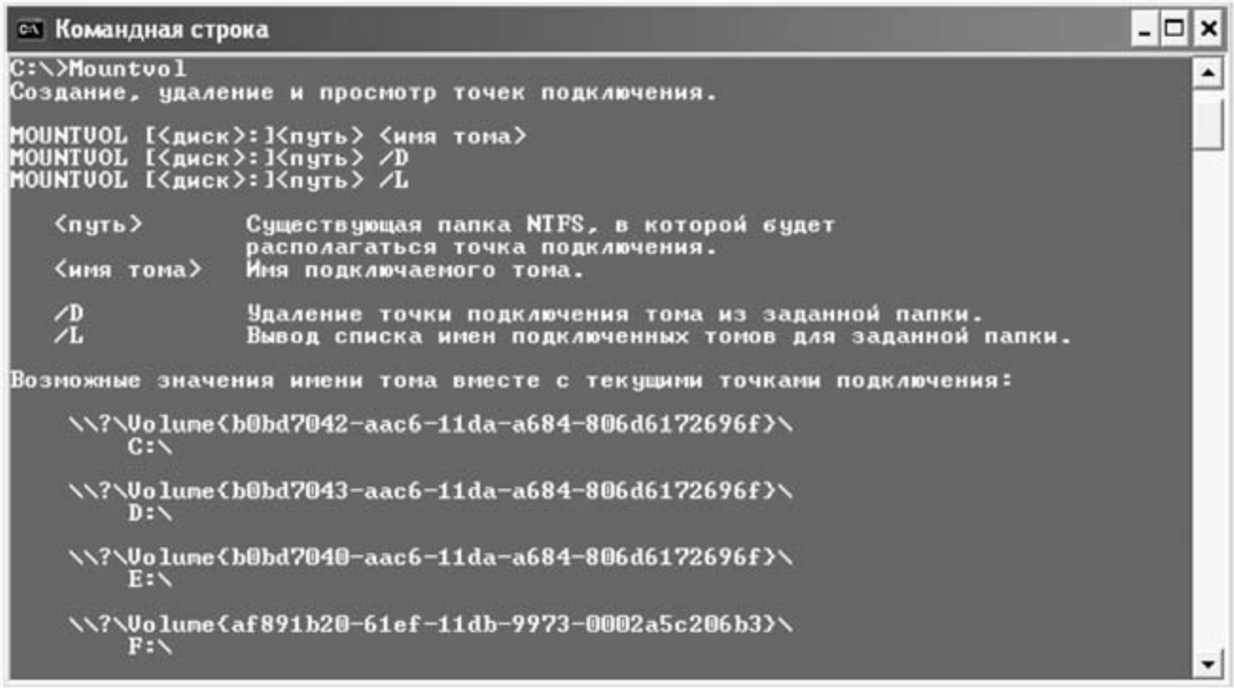


Рис. 4.38

Пусть на компьютере установлено два тома (C: и D:), флэш-память (F:) и устройство CD-ROM (E:). Том C: отформатирован под NTFS 5.0, поэтому на нем можно расположить несколько точек соединения. Для монтирования некоторого тома нужно выполнить следующие действия:

1) в окне командной строки запустить утилиту Mountvol и просмотреть список имен устройств компьютера (см. рис. 4.38);
2) создать пустые папки на диске C: (например, CD — для подключения устройства CD-ROM и MoreDisrSpace — для подключения MoreDisrSpace диска D:), как это показано на рис. 4.39. С помощью утилиты Mountvol подключить тома CD-ROM и диск D:;

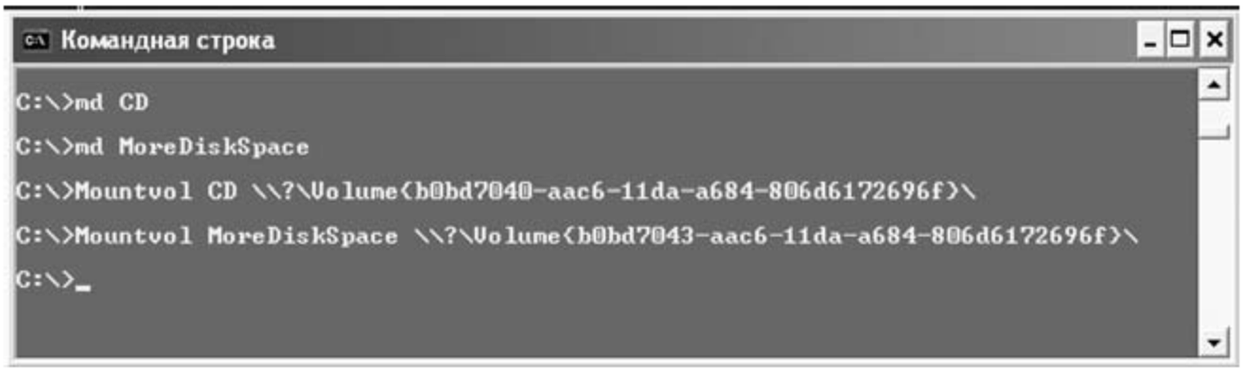


Рис. 4.39

3) открыв папку Мой компьютер, можно увидеть папки CD и MoreDisrSpace в корневом каталоге диска C:, которыми можно пользоваться точно так же, как ранее дисками E: и D: (рис. 4.40);

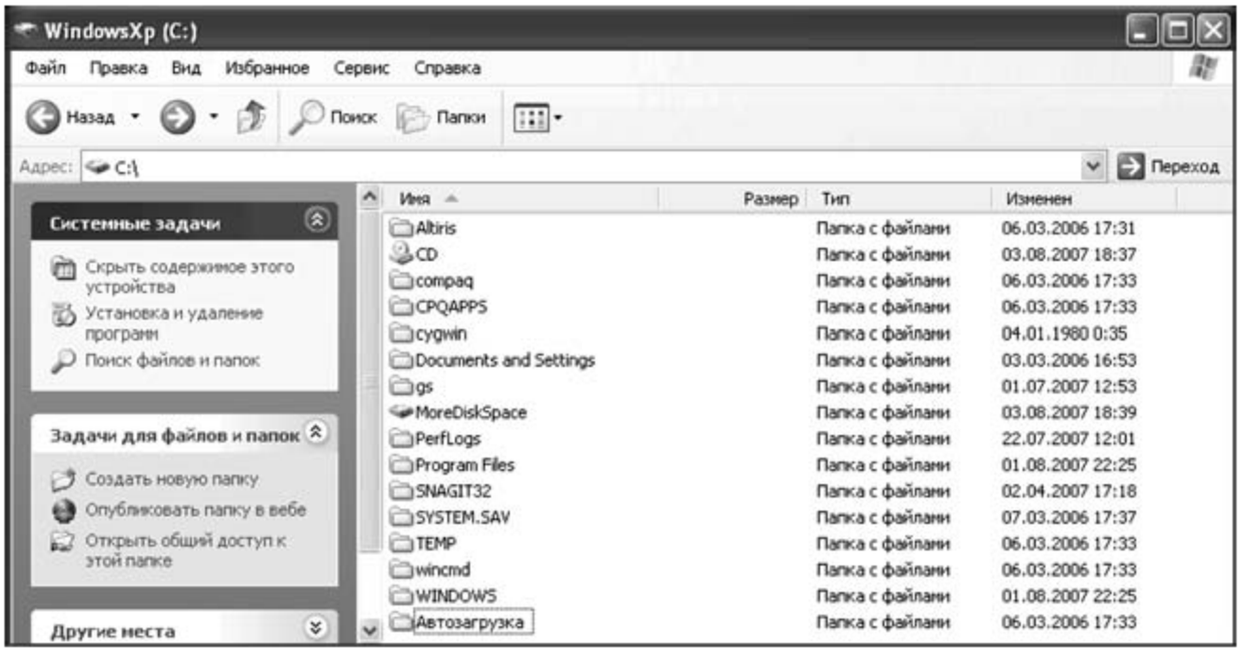


Рис. 4.40

4) для удаления созданных точек соединения в командой строке нужно ввести команды, как показано на рис. 4.41.

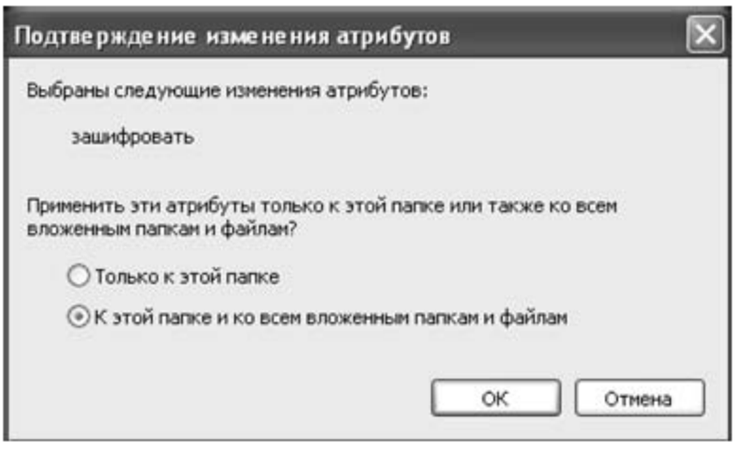


Рис. 4.41

Оснастка Управление дисками позволяет создать точки соединения для дисков компьютера. В качестве примера рассмотрим подключение съемного диска с файловой системой FAT. Для этого нужно запустить оснастку Управление дисками, выбрать нужный диск (в нашем случае диск H:) и нажать правую кнопку мыши.

В контекстном меню выбрать команду Изменение буквы и пути диска. В открывшемся окне нажать кнопку Добавить (рис. 4.42). Откроется окно диалога (рис. 4.43) Добавление новой буквы диска или пути диска. Нажать кнопку Обзор и в новом окне выбрать диск (Точку соединения), например C: (рис. 4.44). Нажать кнопку Создать папку и ввести ее имя, например Диск H, вместо слов Новая папка (рис. 4.45). Нажать OK. Все окна диалога закроются, кроме оснастки Управление компьютером. Если теперь открыть окно Мой компьютер, то в корневом каталоге можно увидеть папку Диск H (рис. 4.46).

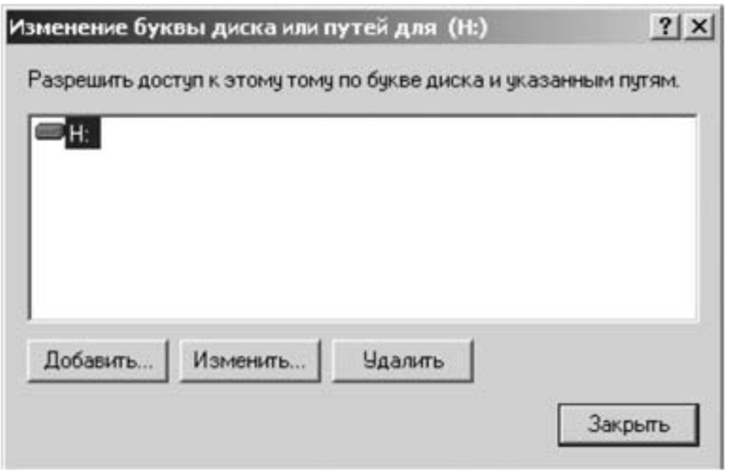


Рис. 4.42

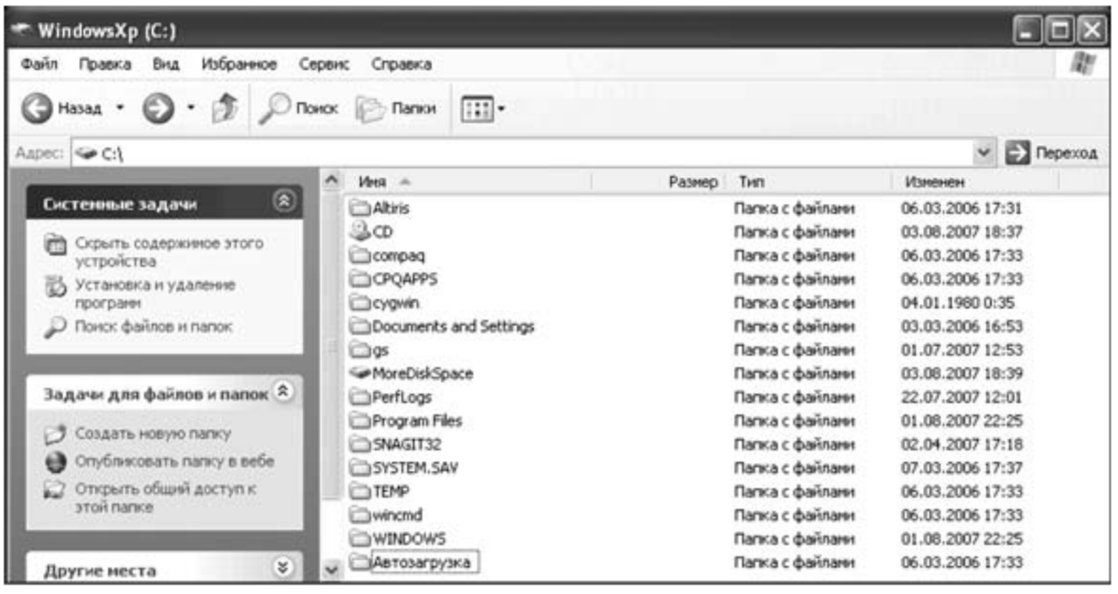


Рис. 4.43

Рис. 4.44

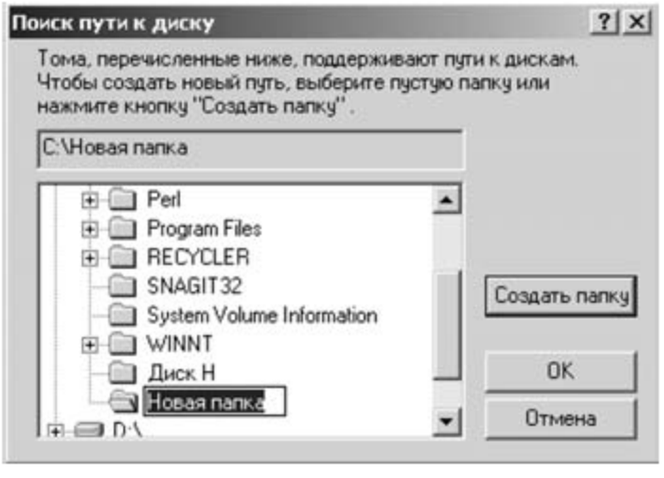


Рис. 4.45

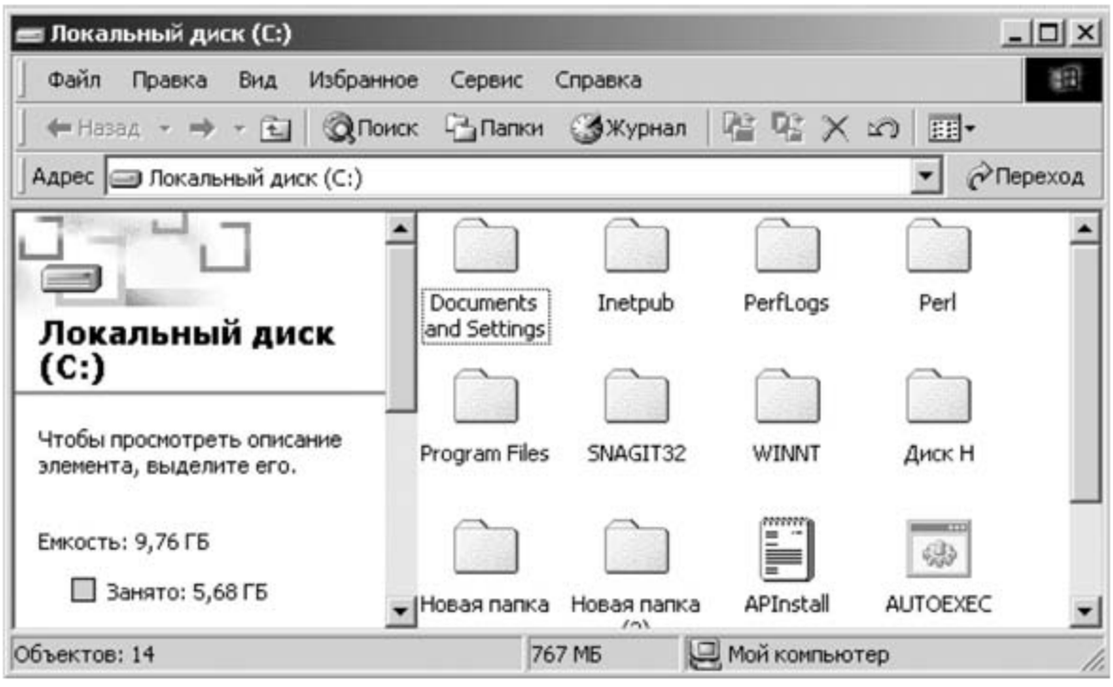


Рис. 4.46

Для удаления точки соединения запустить оснастку Управление дисками, указать нужный том файловой системы и нажать правую кнопку мыши. Выбрать команду Изменить буквы диска и пути диска. В открывшемся окне выбрать нужный путь и нажать кнопку Удалить.

4.5. Шифрующая файловая система EFS

Поскольку шифрование и дешифрование выполняются автоматически, пользователь может работать с файлом так же, как и до установки его криптозащиты. Все остальные пользователи, которые попытаются получить доступ к зашифрованному файлу, получат сообщение об ошибке доступа, поскольку они не владеют необходимым личным ключом, позволяющим им расшифровать файл.

Шифрование информации задается в окне свойств файла или папки. В окне свойств файла на вкладке Общие нужно нажать кнопку Другие. Появится окно диалога Дополнительные атрибуты (рис. 4.47). В группе Атрибуты сжатия и шифрования установить флажок Шифровать содержимое для защиты данных и нажать кнопку OK.

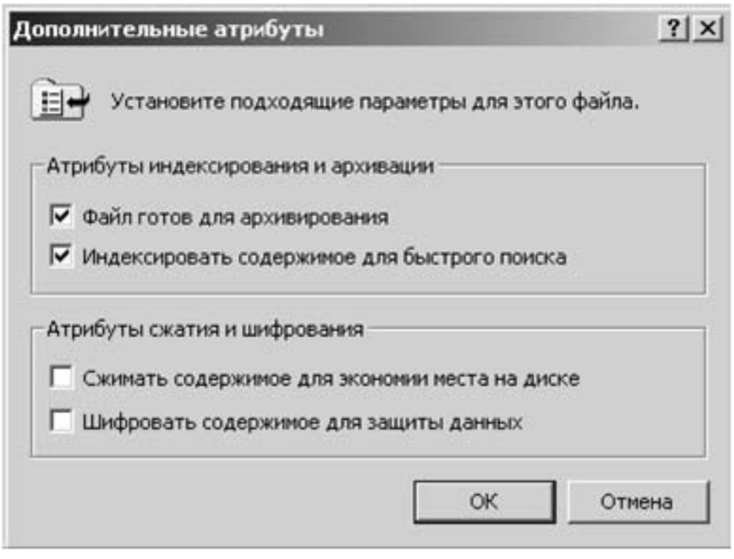


Рис. 4.47

Далее следует нажать кнопку OK в окне свойств зашифровываемого файла или папки. Появится окно (рис. 4.48), в котором надо указать режим шифрования.

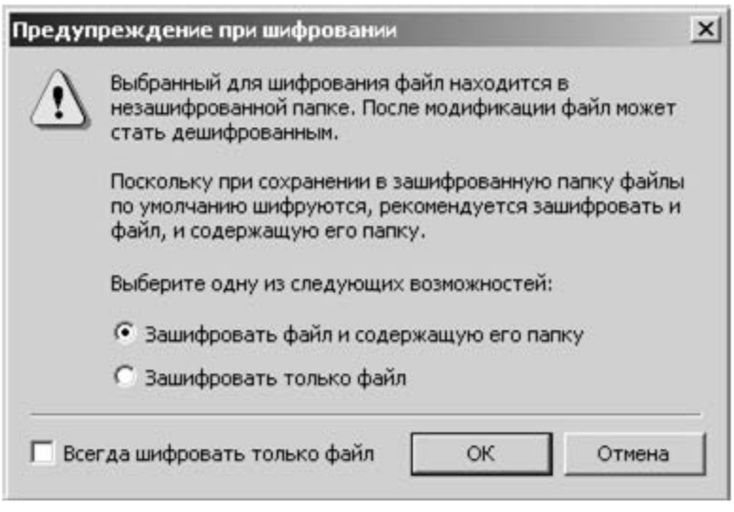


Рис. 4.48

При шифровании папки можно указать следующие режимы применения нового атрибута: Только к этой папке или К этой папке и ко всем вложенным папкам и файлам (рис. 4.49). Для дешифрования файла или папки на вкладке Общие окна свойств соответствующего объекта нажать кнопку Другие и в открывшемся окне сбросить флажок Шифровать содержимое для защиты данных.

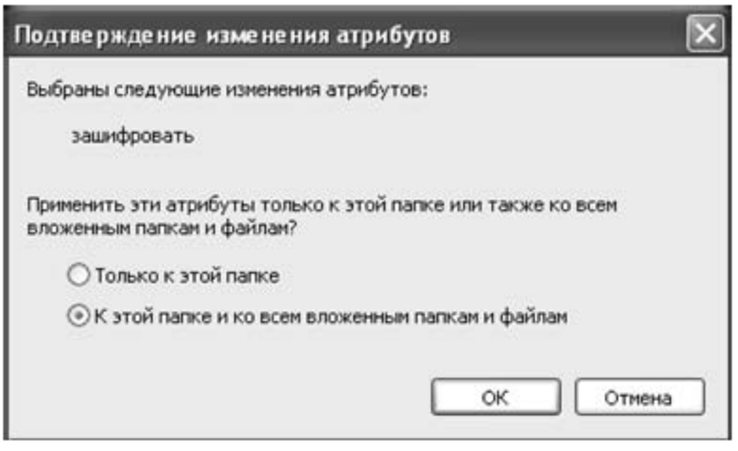


Рис. 4.49

В процессе шифрования файлов и папок система Encrypting File System (EFS) формирует специальные атрибуты (Data Decryption Field — Поле дешифрования данных), содержащие список зашифрованных ключей (FEK — File Encryption Key), что позволяет организовать доступ к файлу со стороны нескольких пользователей (рис. 4.50). Для шифрования файла FEK используется открытая часть пары ключей каждого пользователя. Информация, требуемая для дешифрования, привязывается к самому файлу. Секретная часть ключа пользователя используется при дешифровании FEK. Она хранится в безопасном месте, например на смарт-карте или в устройстве высокой степени защищенности.

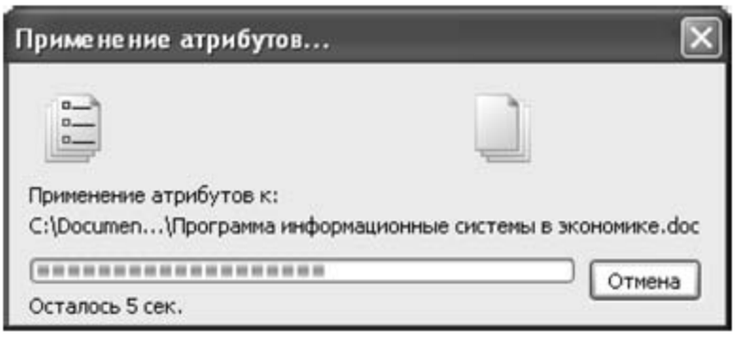


Рис. 4.50

Файл FEK применяется для создания ключей восстановления, которые хранятся в другом специальном атрибуте — Поле восстановления данных (Data Recovery Field, DRF). Сама процедура восстановления выполняется довольно редко (при уходе пользователя из организации или забывании секретной части ключа).

Система EFS имеет встроенные средства восстановления зашифрованных данных в условиях, когда неизвестен личный ключ пользова-

теля. Пользователи, которые могут восстанавливать зашифрованные данные в условиях утраты личного ключа, называются агентами восстановления данных. Они обладают сертификатом (X.509 v.3) на восстановление данных и личным ключом, с помощью которого выполняется операция восстановления зашифрованных данных.

Задания для самостоятельной работы

1. Создать нескольких пользователей на вашем компьютере (условно пользователи А, Б и т.д.).
2. Выполнить следующие действия пользователем А: создать папку Folder1 на диске D: и в ней с помощью программы Блокнот создать три файла: File1.txt, File2.txt, File3.txt.
3. Выполнить аналогичные действия пользователем Б (предварительно перезагрузив компьютер): создать папку Folder2 на диске D: и в ней с помощью программы Блокнот создать три файла: File1.txt, File2.txt, File3.txt.
4. Установить пользователем А (предварительно перезагрузив компьютер) следующие разрешения для пользователя Б:
   — полный доступ к папке Folder1, кроме чтения дополнительных атрибутов;
   — чтение и выполнение для файла File1.txt;
   — разрешить чтение и выполнение, но запретить запись для файла File2.txt;
   — разрешить запись атрибутов, чтение разрешений, запретить запись данных и выполнение файлов для файла File3.txt;
   — передать права владения файлом File1.txt пользователю Б.
5. Установить пользователем Б (предварительно перезагрузив компьютер) следующие разрешения для пользователя А:
   — чтение и запись для Folder2;
   — полный доступ для файла File1.txt;
   — разрешить только выполнение для файла File3.txt;
   — разрешить запись атрибутов, чтение разрешений, запретить запись данных и выполнение файлов для файла File3.txt;
   — передать права владения файлом File31.txt пользователю А.
6. Проверить возможности доступа к созданным папкам и файлам для каждого пользователя.

7. С помощью утилиты Mountvol отобразить корневую папку локального тома флэш-памяти и CD-ROM в точку C:.
8. Используя оснастку Управление дисками, отобразить диск D: и CD-ROM в точку C:.
9. Загрузить компьютер пользователем А, зашифровать File2.txt в папке Folder1. Сменить пользователя А на пользователя Б, попробовать прочитать File2.txt. Объяснить результат.

4.6. Диагностика и мониторинг устройств компьютера

4.6.1. Утилита SiSoftware Sandra

Известная утилита SiSoftware Sandra является по-настоящему универсальной программой, позволяющей работать с широким спектром современных компьютерных систем. Сюда относятся компьютеры, начиная от платформ Pocket PC ARM (КПК и смартфоны) и заканчивая платформами Win64 IA64 (системы на базе Itanium/Itanium2) и AMD64 (системы на базе процессоров AMD Athlon 64/Athlon 64 FX/Opteron) и, конечно же, наиболее распространенной сегодня платформой Win32 x86. Для работы с 64-битными архитектурами следует использовать соответственно 64-битную версию утилиты.

Утилита SiSoftware Sandra является своего рода образцом информационно-диагностического программного обеспечения. Сущность и предназначение этой программы отражены в ее названии: Sandra — это сокращение от System ANalyser, Diagnostic and Reporting Assistant. Она выпускается в нескольких версиях, которые различаются условиями лицензии, ценой и, как следствие, функциональностью. В данном случае рассматривается бесплатная версия (Standard), предназначенная для личного использования и не требующая регистрации. В настоящее время существует несколько версий русификации этой программы (сайт Starsoft — http://www.starsoft.org/).

Утилита требует предварительной установки, но все сполна окупается возможностями, предоставляемыми этой программой. Утилита SiSoftware Sandra имеет традиционный оконный интерфейс (рис. 4.51). Все инструменты мониторинга и диагностики программы поделены на пять категорий согласно их целевому назначению:

1) мастера (Wizard Modules);
2) информационные модули (Information Modules);

3) бенчмаркинговые модули (Benchmarking Modules);
4) тестовые модули (Testing Modules);
5) просмотровые модули (Listing Modules).

Рис. 4.51

В общей сложности в версии Standard доступны 58 модулей. В категории Мастера доступно шесть мастеров:

1) мастер добавления модулей (Add New Module) — позволяет добавлять новые модули в состав утилиты;
2) мастер обобщенного индекса производительности (Combined Performance Index Wizard) — проводит тестирование основных подсистем компьютера: процессорной (арифметическая производительность и мультимедийная производительность), подсистемы памяти, дисковой подсистемы и сетевого интерфейса, на основе чего выставляется обобщенный индекс мастера. Наибольший интерес вызывает графическое представление результатов в виде пятиугольной матрицы покрытия, которая позволяет наглядно оценить производительность тестируемой системы в сравнении с другими конфигурациями (рис. 4.52). При этом пользователь может сам составить эталонную конфигурацию компьютерной системы, производительность которой он хотел бы сравнить со своим ПК;
3) мастер стресс-тестирования (Burn-in Wizard) — позволяет проверить компьютерную систему на «выносливость» путем многократного циклического запуска тестов (которые можно найти

в категории «Бенчмаркинговые модули»). Немаловажно, что можно обеспечить защиту системы от последствий таких жестких нагрузок, задав условие прекращения теста при перегреве или ошибках, при этом критические температуры и предельные параметры работы систем охлаждения (скорость вращения вентиляторов охлаждения) также могут быть определены пользователем. Кроме того, имеются возможности выбора тестов, которые будут запускаться, и количества запусков, причем можно даже задавать приоритет данного приложения;

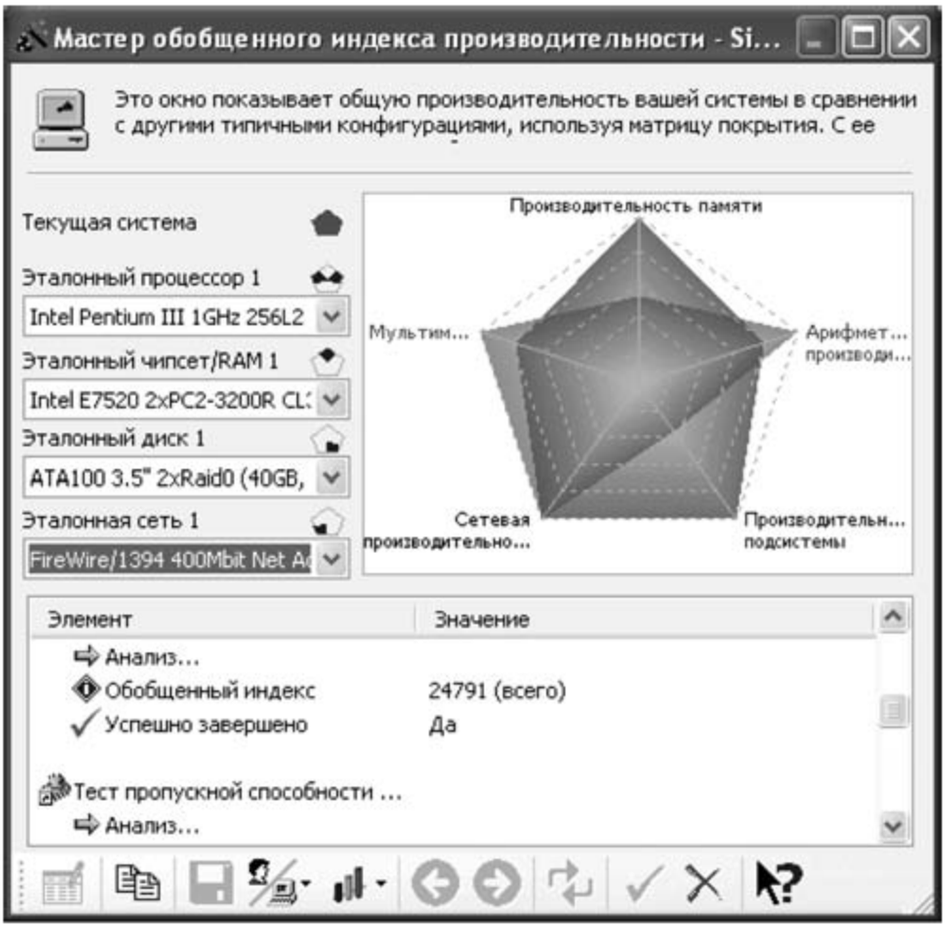


Рис. 4.52

4) мастер обновлений (WebUpdate Wizard) — позволяет производить онлайн-обновления версии утилиты;
5) мастер увеличения производительности (Performance Tune-up Wizard) — запускает все активные информационные модули и на основе полученной информации дает советы по оптимизации и модернизации системы, способствующие повышению производительности компьютерной системы;
6) мастер составления отчетов (Create a Report Wizard) — помогает сохранять полученную информацию в наиболее удобном для

пользователя формате: текстовом (txt), иерархическом (rpt), SMS/DMI (mif) или Web (html и xml). Кроме того, предоставляет возможность выбрать место доставки полученного отчета.

В категории «Информационные модули» можно найти инструменты, позволяющие получить исчерпывающую информацию практически обо всех аппаратных и программных компонентах компьютерной системы.

Категория «Бенчмаркинговые модули» включает ряд хорошо известных синтетических тестов, позволяющих оценить производительность наиболее важных компьютерных подсистем (за исключением видеоподсистемы). Эта категория содержит следующие тестовые утилиты:

- арифметический тест процессора (CPU Arithmetic Benchmark) — позволяет оценить производительность выполнения арифметических вычислений и операций с плавающей запятой в сравнении с другими эталонными компьютерными системами;
- мультимедийный тест процессора (CPU Multi-Media Benchmark) — дает возможность оценить производительность системы в работе с мультимедийными данными при использовании поддерживаемых процессором наборов SIMD-инструкций в сравнении с другими эталонными компьютерными системами;
- тест съемных флэш-накопителей (Removable Storage/Flash Benchmark) — дает возможность оценить производительность системы (скорость чтения, записи и удаления, на основе чего вычисляется обобщенный индекс) при работе со съемными накопителями в сравнении с другими эталонными компьютерными системами;
- тест файловой системы (File System Benchmark) — позволяет определить производительность дисковой (файловой) подсистемы компьютера в сравнении с другими эталонными компьютерными системами;
- тест CD-ROM/DVD (CD-ROM/DVD Benchmark) — позволяет оценить производительность оптических приводов (CD-ROM/DVD) в сравнении с другими эталонными компьютерными системами;
- тест пропускной способности памяти (Memory Bandwidth Benchmark) — позволяет определить пропускную способность подсистемы памяти (связка процессор—чипсет—память) при выполнении целочисленных операций и операций с плавающей запятой в сравнении с другими эталонными компьютерными системами;
- тест кэш и памяти (Cache & Memory Benchmark) — дает возможность определить пропускную способность подсистемы памяти (связ-

ка процессор—кэш—чипсет—память) в сравнении с другими эталонными компьютерными системами;

■ тест пропускной способности сети (Network/LAN Bandwidth Benchmark) — позволяет определить пропускную способность сетевого соединения с выбранным узлом сети.

Кроме того, среди тестовых модулей можно найти две тестовые утилиты, оценивающие скорость работы Интернета: 1) Тест соединения с Интернетом (Internet Connection Benchmark) — позволяет оценить скорость соединения с Интернет-провайдером; 2) Тест скорости с Интернет (Internet Peerage Benchmark) — скорость соединения с различными Интернет-сайтами.

Категория «Просмотровые модули» осуществляет доступ к инструментам просмотра наиболее важных системных файлов, определяющих конфигурацию системной среды.

В завершение для более полного представления о возможностях описываемой утилиты SiSoftware Sandra необходимо упомянуть еще об одной интересной функции, предоставляемой данным ПО, — это быстрый вызов сервисов и утилит Windows, предназначенных для настройки, диагностики и оптимизации работы системы.

4.6.2. Утилита CPU-Z

Утилита CPU-Z — небольшая (505 Кбайт) не требующая установки программа с удобным дружественным интерфейсом, предоставляющая пользователю доступ к информации, сгруппированной по пяти категориям, переход между которыми осуществляется посредством кнопок-закладок. Разработчиком утилиты является фирма Frank Delattre (http://www.cpuid.com/).

Первая вкладка — CPU — содержит подробнейшую информацию о центральном процессоре компьютерной системы (рис. 4.53). В этом окне отображается информация о процессорном ядре, а также сведения о текущем напряжении питания, частоте системной шины, FSB, установленном множителе процессора и текущей тактовой частоте процессорного ядра. Здесь же можно найти данные о размере кэша первого (L1) и второго (L2) уровней.

Вторая вкладка — Cache — содержит более подробную информацию о структуре и рабочих параметрах кэш-памяти.

Вкладка Mainboard включает информацию о системной плате:

■ сведения о производителе системной платы;

■ название чипсета;

- название микросхемы южного моста;
- название используемого чипа контроллера ввода-вывода (Super I/O);
- информацию о версии кода BIOS;
- информацию об основных параметрах контроллера графического порта (AGP).

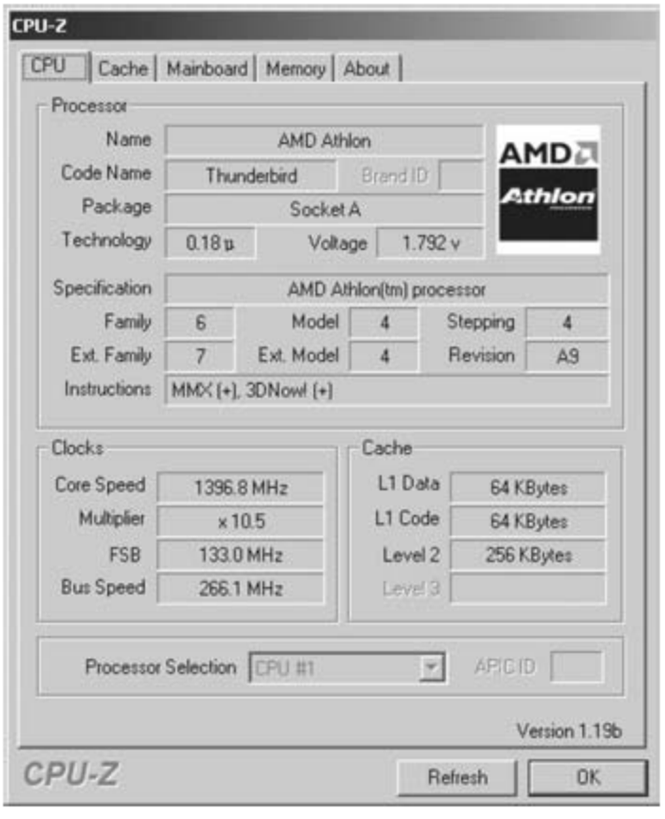


Рис. 4.53

На вкладке Memory можно получить сведения об оперативной памяти: о размере, режиме работы, установленных таймингах, а также об используемых модулях оперативной памяти.

Последняя вкладка — About — помимо традиционной информации об авторе программы открывает доступ к ряду инструментов, предоставляющих в распоряжение пользователя следующие возможности:

- HTML Summary — позволяет сохранить всю полученную информацию в html-документе;
- CPUID Output — дает возможность просмотреть CPUID-информацию, сохранив ее в текстовом формате;
- PCI Device List — выдает подробную низкоуровневую информацию об установленных в системе PCI-устройствах, также сохраняемую в текстовом файле;
- Memory SPD — позволяет получить информацию об установках SPD (Serial Presence Detect) модулей памяти;

■ инструмент Cache Latency представляет собой тест, с помощью которого можно определить латентность памяти.

Задачи

4.1. Пользовательский процесс формирует строку из 70 символов для вывода на принтер, затрачивая на это 5 мс. Объем буфера равен одной строке. Страница текста содержит 50 строк. Принтер способен печатать 10 страниц в мин. Будет ли приостанавливаться пользовательский процесс? Если да, то насколько? Улучшит ли ситуацию двойная буферизация?

4.2. Страница текста состоит из 50 строк по 80 символов. Принтер способен печатать 6 страниц в мин, причем время вывода символа на принтер настолько мало, что им можно пренебречь. Имеет ли смысл управлять выводом на этот принтер при помощи прерываний, если для печати каждого символа требуется прерывание, занимающее 50 мс?

4.3. Информация от модема поступает со скоростью 50 Кбит в 1 с, размещаясь в двух переключаемых системных буферах, каждый из которых имеет емкость в 1 Кбайт. Перемещение данных из буфера в пользовательский процесс занимает 7 мс. Пользовательский процесс затрачивает 50 мс на обработку одного блока данных. Возможны ли при этих условиях потери данных, поступающих от модема?

4.4. Информация от модема поступает в два переключаемых системных буфера, каждый из которых имеет емкость 1 Кбайт. Перемещение данных из буфера в пользовательский процесс занимает 10 мс. Пользовательский процесс затрачивает 50 мс на обработку одного блока данных. Какова максимально возможная скорость работы модема в этих условиях?

4.5. При работе модема (скорость 56 Кбит/с) драйвер посылает один символ, после чего блокируется. После передачи символа в линию происходит прерывание, драйвер разблокируется и посылает следующий символ и т.д. Какую часть времени центрального процессора занимает управление модемом, если обработка прерывания, вывод одного символа и блокировка занимают 100 мкс? Предположите, что у каждого передаваемого символа имеется один стартовый и один стоповый бит, а всего символ занимает 10 бит.

4.6. Жесткий диск имеет емкость 20 Гбайт и размещение файлов в виде связного списка кластеров. Размер кластера 8 Кбайт. Определите максимальное количество файлов, которое можно разместить на диске и долю адресной информации в процентах от емкости диска.

4.7. Жесткий диск имеет емкость 8 Гбайт и непрерывное размещение файлов. Размер кластера 2 Кбайт. Определите максимальное количество файлов, которое можно разместить на диске и долю адресной информации в процентах от емкости диска.

4.8. Жесткий диск имеет емкость 40 Гбайт и размещение файлов в виде перечня номеров кластеров. Размер кластера 4 Кбайт. Определите максимальное количество файлов, которое можно разместить на диске и долю адресной информации в процентах от емкости диска.

4.9. Жесткий диск имеет емкость 2 Гбайт и файловую систему FAT16. Выберите размер кластера и оцените, сколько кластеров будет содержать область данных и FAT-таблица. Определите максимальное количество файлов, которое можно разместить на диске и долю неиспользованной емкости диска за счет внутренней фрагментации.

4.10. В файловой системе UNIX блоки диска имеют емкость 2 Кбайт, а дисковые адреса 4 байт. Чему равен максимальный размер файла, если i-узел содержит 10 прямых адресов и по одному одинарному, двойному и тройному косвенному элементу?

4.11. Сравните возможную степень неиспользуемого дискового пространства за счет внутренней фрагментации в файловых системах FAT16 и FAT32 при емкости винчестера 2 Гбайт и размере кластера 4 Кбайт.

4.12. Для устранения недостатка файловой системы с непрерывным размещением файлов нужно уплотнять диск при каждом удалении файла. Поскольку все файлы являются непрерывными, для копирования файла требуется время на поиск цилиндра и задержку вращения диска при считывании файла, после чего происходит перенос данных на полной скорости. При записи на диск требуются такие же операции. При времени поиска цилиндра, равном 5 мс, задержке вращения в 4 мс, скорости передачи данных 8 Мбайт/с и среднем размере файла 8 Кбайт сколько понадобится времени, для того чтобы прочитать файл в оперативную память, а затем записать его обратно на новое место на диске? При тех же параметрах сколько потребуется времени для уплотнения половины 16-гигабайтного диска?

4.13. Сравните возможную степень неиспользуемого дискового пространства за счет внутренней фрагментации в файловых системах FAT12 и FAT16 при емкости винчестера 512 Мбайт. Какой размер будет иметь кластер? Сколько места займет FAT-таблица?

4.14. Сравните для жесткого диска емкостью 20 Гбайт с размещением файлов в виде связанного списка кластеров и в виде связанного

списка индексов при размере кластера 8 Кбайт максимальное количество файлов, которое можно разместить на диске, и долю адресной информации в процентах от емкости диска.

4.15. Диск имеет емкость 4 Гбайт и разбит на два раздела, каждый по 2 Гбайт. В первом разделе используется FAT16, а во втором FAT32. При равном размере кластера (каком именно?) какая из FAT-таблиц будет иметь больший объем и какой из разделов будет иметь большую фрагментацию?

4.16. У гибкого диска 40 цилиндров. Операция поиска занимает 6 мс на цилиндр. Если не пытаться разместить блоки файла близко друг к другу, два логически последовательных блока окажутся в среднем на расстоянии 13 цилиндров друг от друга. Однако если ОС пытается объединять логически соседние блоки в кластеры, то среднее межблоковое расстояние может быть уменьшено до двух цилиндров. Сколько понадобится времени в обоих случаях для считывания 100-блочного файла, если задержка вращения составляет 100 мс, а время переноса одного блока равно 25 мс?

4.17. У контроллера DMA четыре канала. Контроллер способен запрашивать 32-разрядное слово через каждые 100 нс. Ответ на запрос занимает столько же времени. Насколько быстрой должна быть шина, чтобы не стать узким местом системы?

4.18. Драйвер диска получает запросы на чтение-запись к цилиндрам 10, 22, 20, 2, 40, 6, 38. Перемещение блока головок с одного цилиндра на соседний занимает 6 мс. Сколько времени потребуется на перемещение головок при использовании алгоритма:

а) обслуживания в порядке поступления запросов;

б) обслуживания в первую очередь ближайшего цилиндра?

Во всех случаях начальное положение блока головок на цилиндре 20.

4.19. Предполагая, что для копирования одного байта требуется 10 нс. Сколько времени понадобится, чтобы полностью перерисовать отображаемый на адресное пространство памяти экран, работающий в символьном режиме с разрешением 25 строк по 80 символов? Какой результат получится в графическом режиме с разрешением 1024 x 768 пикселов в формате RGB?

4.20. В компьютере есть кэш, основная память и диск, который используется для организации виртуальной памяти. Если слово, к которому производится обращение находится в кэше, для доступа к нему требуется 2 нс. Если это слово находится в основной памяти, но отсутствует в кэше, то оно сначала загружается в кэш за 10 нс, а затем к нему

производится обращение. Если нужного слова нет в памяти, то чтобы найти его на диске и загрузить в основную память, требуется 12 мс; еще 10 нс нужны, чтобы скопировать его в кэш, и только затем к этому слову производится обращение. Результативность обращений к кэшу равна 0,9, а результативность обращений к основной памяти — 0,6.

Требуется определить среднее время, которое требуется для доступа системы к нужному ей слову.

ГЛАВА 5

СРЕДСТВА ЗАЩИТЫ И ВОССТАНОВЛЕНИЯ ОПЕРАЦИОННЫХ СИСТЕМ

5.1. Цифровая подпись драйверов

Не существует абсолютно надежных и отказоустойчивых ОС. В процессе функционирования компьютера может возникать ряд проблем, приводящих к сбоям, повреждениям и отказам ОС. В частности, возникновение проблем может быть вызвано следующими ситуациями:

- сбои в работе жесткого диска или ошибки в работе контроллера жесткого диска;
- повреждение главной загрузочной записи (Master Boot Record, MBR) или загрузочного сектора на системном разделе;
- отсутствие или повреждение одного из файлов, необходимого для загрузки ОС;
- сбои в подаче электропитания;
- некорректно работающие приложения;
- неподходящие (или плохо написанные) драйверы;
- ошибки пользователей (случайные и преднамеренные);
- вирусные атаки;
- противоправные действия третьих лиц.

Регулярное выполнение профилактических процедур и выработанных многолетней практикой рекомендаций позволит восстановить поврежденную систему в кратчайшие сроки и с минимальными потерями. В составе ОС Windows имеется большой набор средств обеспечений отказоустойчивости и предотвращения сбоев. Основными из них являются:

1) средства защиты системных файлов цифровой подписью;
2) безопасный режим загрузки;
3) средства резервного копирования и восстановления;
4) консоль восстановления.

Все системные файлы и драйверы ОС Windows XP/2000/2003 защищены цифровой подписью. Цифровая подпись Microsoft гарантирует, что файл, подписанный ею, тестировался на совместимость с Windows 2000 и не был модифицирован или переписан во время установки дополнительного программного обеспечения.

В зависимости от установленных опций настройки Windows 2000 может игнорировать драйверы, не имеющие цифровой подписи, или выводить предупреждение при обнаружении таких драйверов (опция по умолчанию), или не допускать их установки. Для установки требуемой опции защиты системных файлов Windows 2000 необходимо выполнить следующие действия:

1) на панели управления щелкнуть значок Система (System) и перейти на вкладку Оборудование (Hardware) (рис. 5.1);

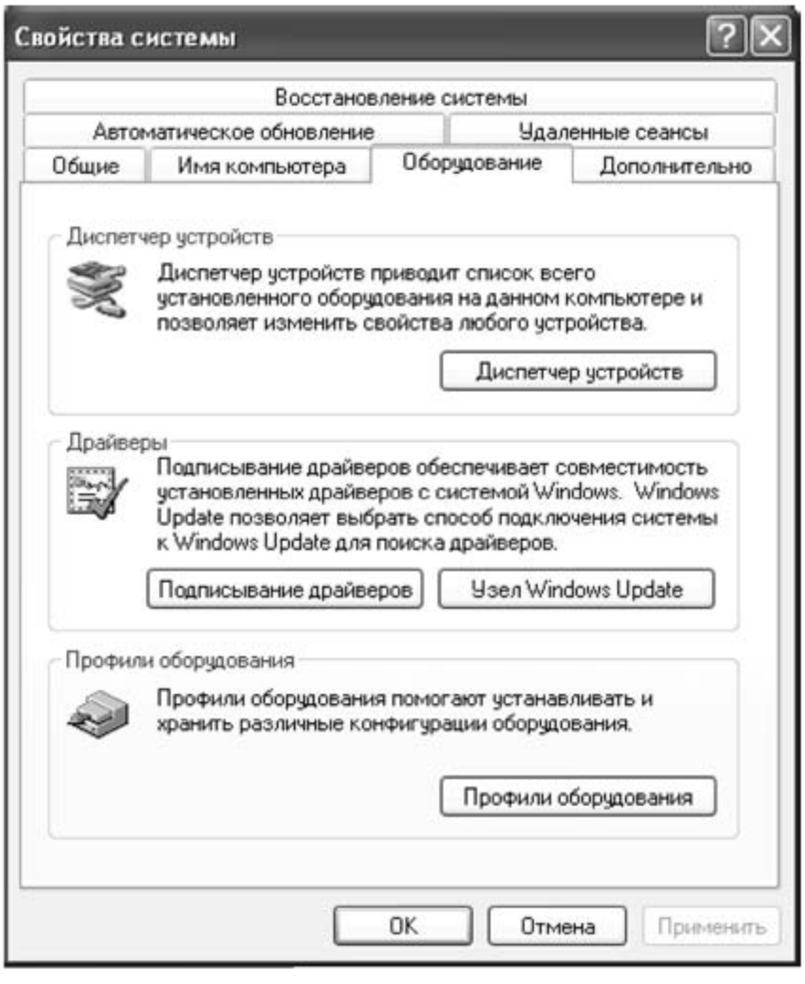


Рис. 5.1

2) нажать кнопку Подписывания драйверов (Driver Signing). На экране появится диалоговое окно Параметры подписывания драйвера (Driver Signing Options) (рис. 5.2), в котором имеется группа Проверка подписи файла (File Signature Verification);

3) выбрать требуемую опцию из следующих возможных:
   — Пропускать (Ignore);
   — Предупреждать (Warn);
   — Блокировать (Block).

Если установлен переключатель Пропускать, то система позволяет устанавливать любые драйверы и системные файлы, игнорируя наличие или отсутствие у них цифровой подписи. Если установлен переключатель Предупреждать, то система будет выводить предупреждающие сообщения при попытке установить драйвер или системный файл, не имеющий цифровой подписи. Если установлен переключатель Блокировать, то драйверы, не имеющие цифровой подписи, устанавливаться не будут.

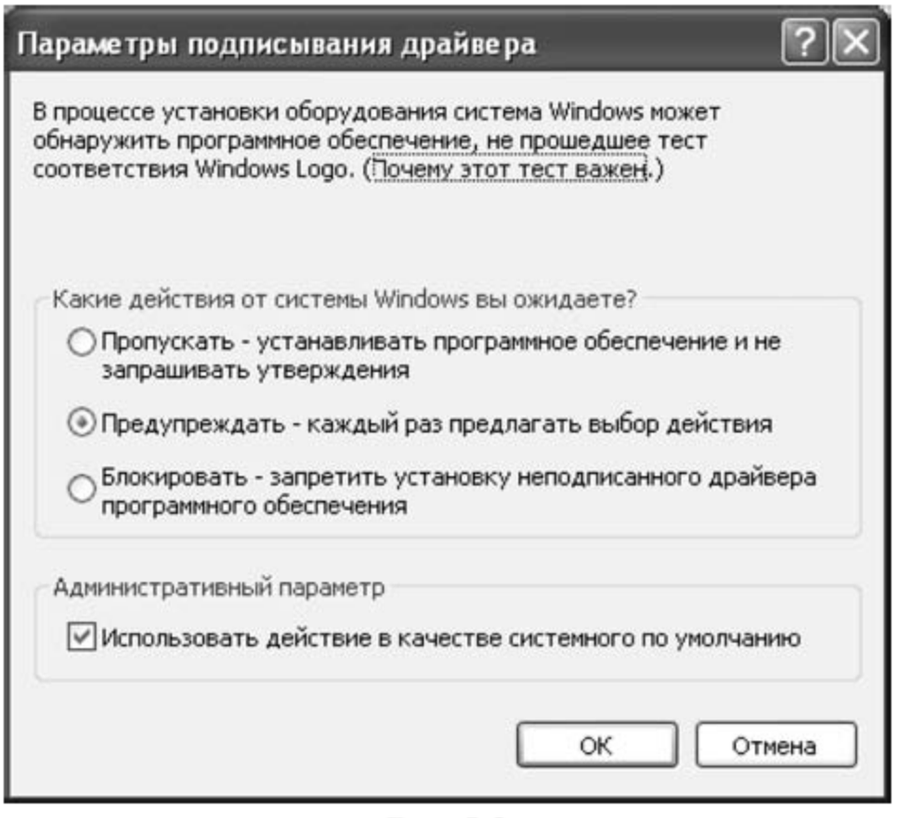


Рис. 5.2

5.2. Защита системных файлов

Кроме цифровых подписей ОС, Windows 2000 имеет ряд функциональных возможностей по защите драйверов и системных файлов. Все версии Windows (до Windows 2000) имели недостаток — при установке дополнительного программного обеспечения практически любые совместно используемые системные файлы, в том числе *.dll и *. exe, могли быть изменены. Последствия замены этих файлов некоррект-

ными или несовместимыми версиями могли быть непредсказуемыми: от снижения производительности ОС до некорректного поведения остальных приложений, периодического появления ошибок STOP и даже проблеем с загрузкой.

Попытка исправить такую ситуацию сделана в Windows 2000/NT. Специальная функция защиты системных файлов Windows (Windows File Protection, WFP) включает в свой состав два компонента:

1) сервис защиты системных файлов WFP (Windows File Protection service);
2) утилиту командной строки System Checker (Sfc.exe).

Сервис WFP работает по принципу определения цифровых подписей защищенных системных файлов, не позволяя произвольно модифицировать и замещать эти файлы. Он функционирует в фоновом режиме и защищает все системные файлы, установленные программой Windows Setup при инсталляции ОС. Выявив попытку изменения или перемещения защищенного файла, сервис WFP проверяет измененный файл на наличие у него цифровой подписи.

В случаях когда версия, предназначенная для замены, не является корректной, этот файл замещается резервной копией из папки %SystemRoot%\system32\dllcache и регистрируется попытка замещения файла в журнале системных событий.

По умолчанию функция WFP всегда активизирована и позволяет выполнять замену системных файлов только в случае установки следующих видов программного обеспечения:

- сервисных пакетов Windows 2000 (с использованием программы Update.exe);
- дистрибутивных пакетов типа Hotfix (с использованием Hotfix.exe);
- обновления версии ОС (с помощью Wjnnt32.exe);
- программного обеспечения Windows Update.

5.3. Проверка системных файлов

Средство проверки системных файлов (System File Checker, Sfc.exe) представляет собой утилиту командной строки, которая сканирует все установленные системные файлы и выполняет проверку их версий при перезагрузке компьютера. Если она обнаружит, что один из защищаемых системных файлов был замещен, она найдет корректную версию этого файла в каталоге %SystemRoot%\system32\dllcache и запишет ее поверх измененного файла. Синтаксис этой утилиты показан на рис. 5.3.

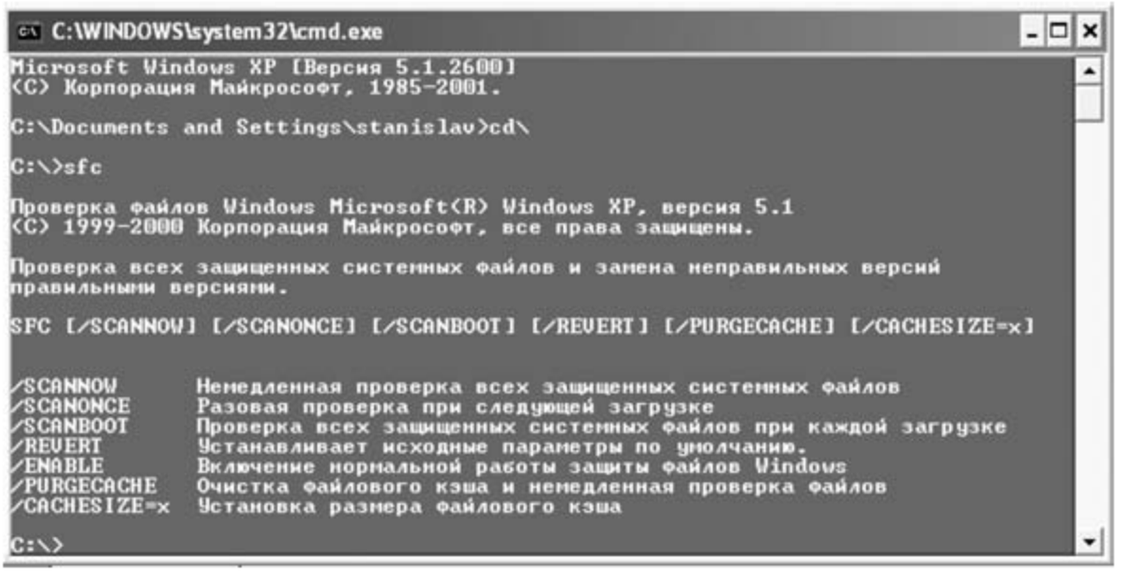


Рис. 5.3

Папка %SystemRoot%\system32\dllcache может быть достаточно объемной (до 300 Мбайт), поэтому для экономии дискового пространства ее иногда удаляют. Однако следует иметь в виду, что, если сервис WFP не сможет обнаружить нужной версии файла в папке %SystemRoot%\system32\dllcache, система потребует установить дистрибутивный CD-ROM или указать путь к папке, из которой такая версия может быть скопирована. Запустив команду sfc/scannow, можно проверить все защищенные системные файлы (рис. 5.4). Если каких-то файлов нет, утилита потребует их установить (рис. 5.5).

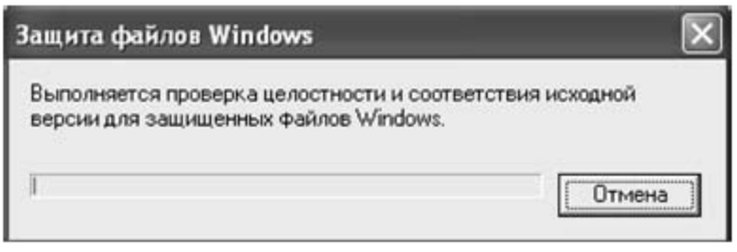


Рис. 5.4

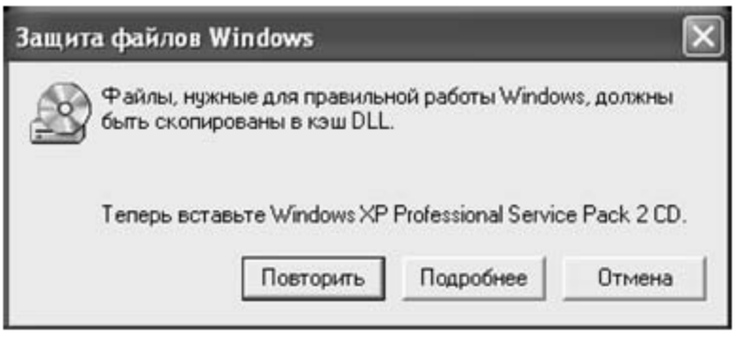


Рис. 5.5

5.4. Верификация цифровой подписи файлов

Верификация (проверка) цифровой подписи файлов (программа sigverif) позволяет идентифицировать все установленные на проверяемом компьютере файлы, не имеющие цифровой подписи, и получить об этих файлах следующую информацию: имя файла и путь к нему, дату модификации файла, тип файла и точный номер его версии.

Для устранения проблем, связанных с заменой системных файлов некорректными версиями, нужно использовать информацию, собранную программой Sigverif в файле журнала. Для этого необходимо:

1) запустить программу Sigverif (из командной строки) и в появившемся на экране окне Проверка подписи файла (File Signature Verification) (рис. 5.6) щелкнуть по кнопке Дополнительно (Advanced);

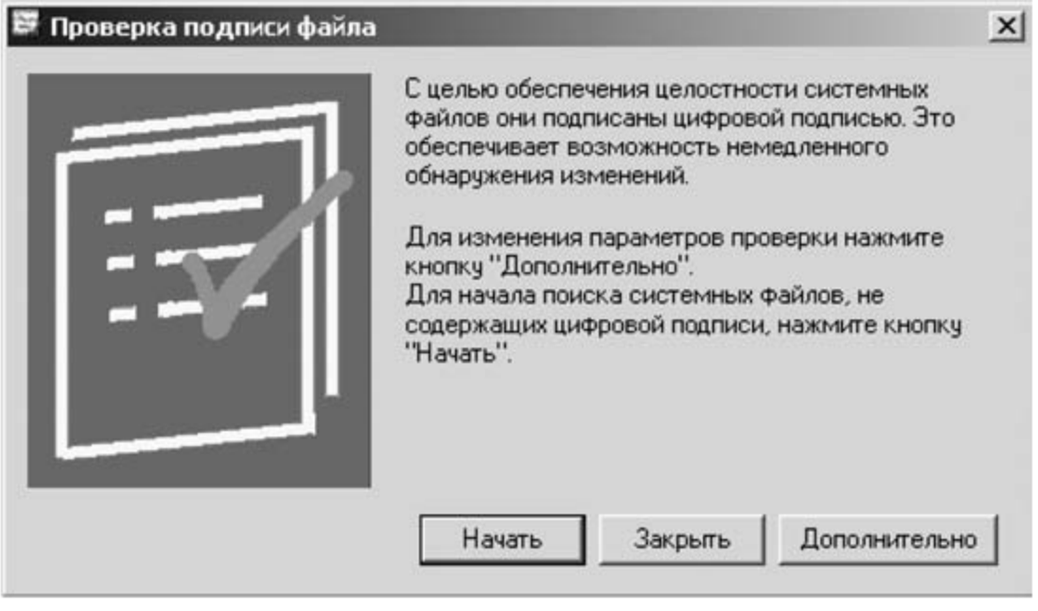


Рис. 5.6

2) в новом окне Дополнительные параметры проверки подписи файла (Advanced File Signature Verification Settings) (рис. 5.7) перейти на вкладку Ведение журнала (Logging) и установить флажок Сохранять результаты проверки подписи в журнале (Save the fib signature verification results to a log file);
3) перейти в группу Параметры журнала (Logging options) и установить желаемую опцию ведения журнала из следующих возможных:
   — добавлять к существующему журналу (Append to existing log file);
   — заменять существующий журнал (Overwrite existing log file);

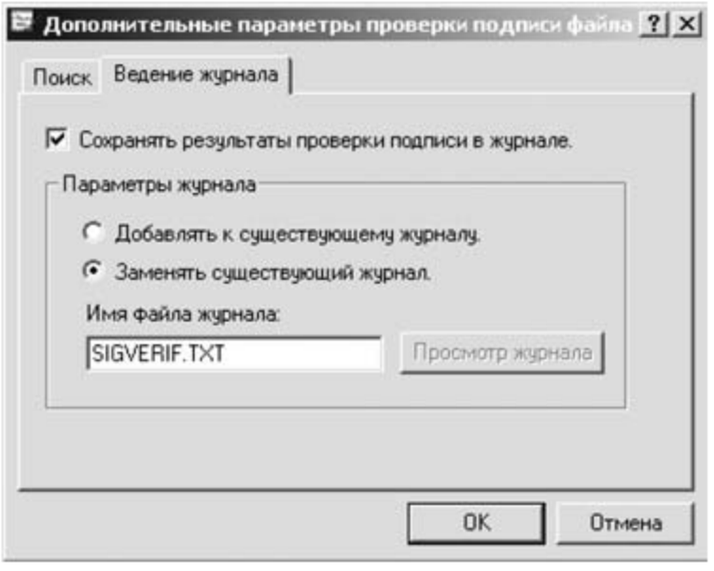


Рис. 5.7

4) в поле Имя файла журнала (Log file name) можно ввести имя файла журнала;
5) нажать кнопку OK. Произойдет возврат в окно Проверка подписи файла;
6) для начала сканирования нажать кнопку Начать (Start). Процесс сканирования индицируется индикатором Просмотр файлов (Scanning files), как показано на рис. 5.8. Его можно прервать нажатием кнопки Остановить (Stop);

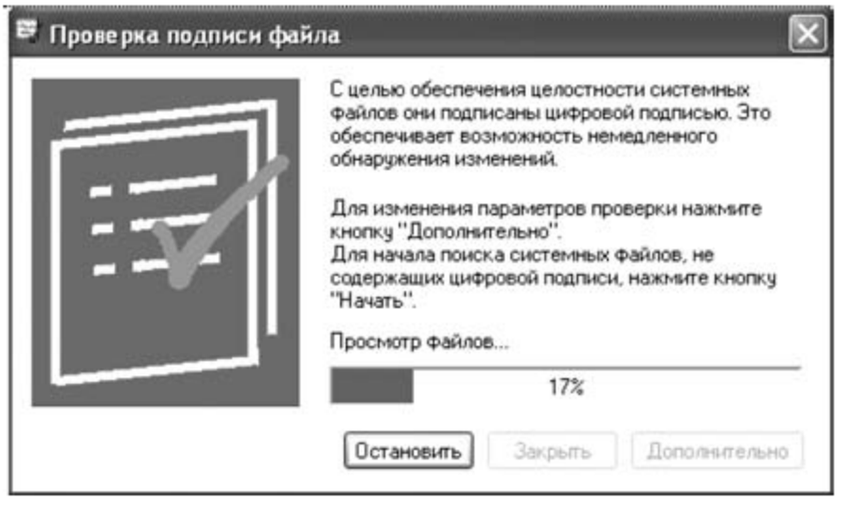


Рис. 5.8

7) после завершения сканирования на экране появится окно Результаты проверки подписи (Signature Verification Results). В нем будет отображен список файлов, не имеющих цифровой подписи. Если таких файлов нет, можно просмотреть журнал результатов проверки (рис. 5.9).

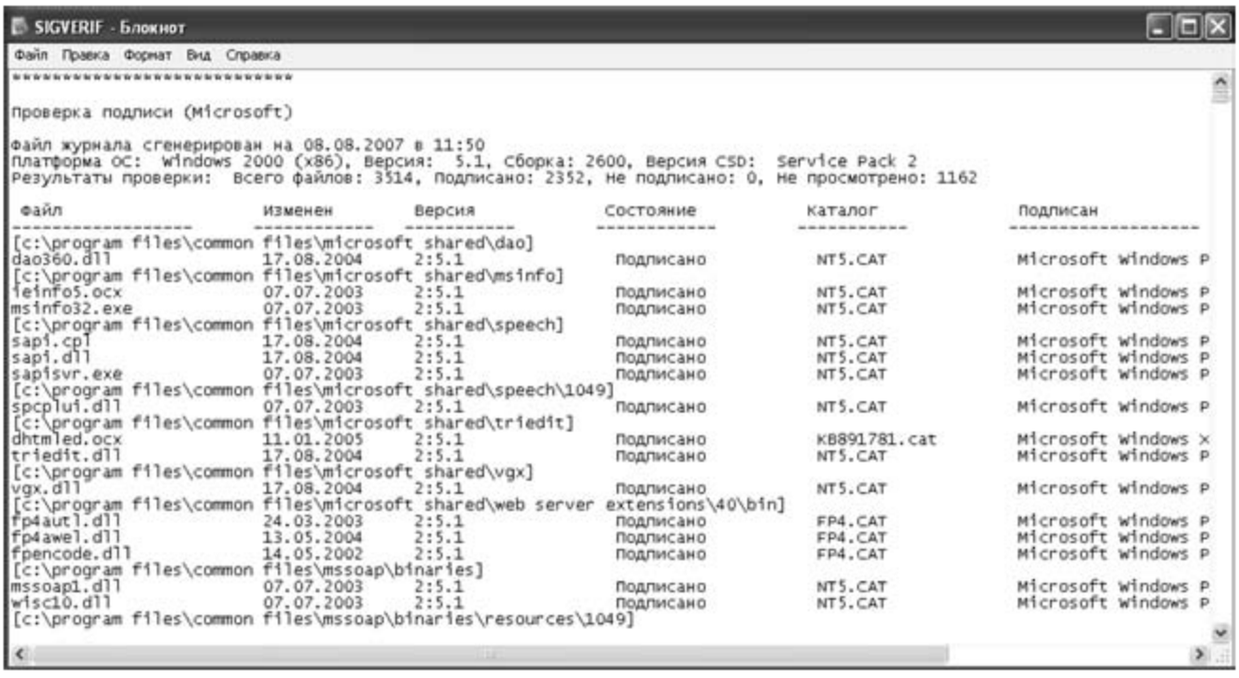


Рис. 5.9

5.5. Откат драйверов

Довольно часто такие проблемы, как аппаратные конфликты, нестабильное поведение системы, неправильная работа устройств и даже ошибки STOP [17], бывают вызваны некорректным драйвером. В таких случаях желательно быстро заменить проблемный драйвер предыдущей версией без переустановки системы.

Функция отката драйвера оказывается незаменимой при устранении неполадок, при отладке бета-версий драйверов и в других ситуациях, например если после установки обновленного драйвера при загрузке появляется сообщение STOP, попытаться загрузить систему в безопасном режиме и произвести откат драйвера.

Чтобы воспользоваться функцией отката драйверов, нужно выполнить следующие действия:

1) выбрать значок Система на панели управления, перейти на вкладку Оборудование и нажать кнопку Диспетчер устройств;
2) выполнить щелчок правой кнопкой мыши на устройстве, обновленный драйвер которого вызывает проблему, и выбрать из контекстного меню строку Свойства;
3) в раскрывшемся диалоговом окне свойств выбранного устройства перейти на вкладку Драйвер (рис. 5.10) и нажать кнопку Откатить;
4) диспетчер устройств предложит подтвердить намерение выполнить откат драйвера. Для этого следует нажать кнопку Да. Если

старая версия драйвера недоступна, функция отката драйвера выведет окно с уведомлением (рис. 5.11) и предложит воспользоваться другими средствами устранения неполадок.

Рис. 5.10

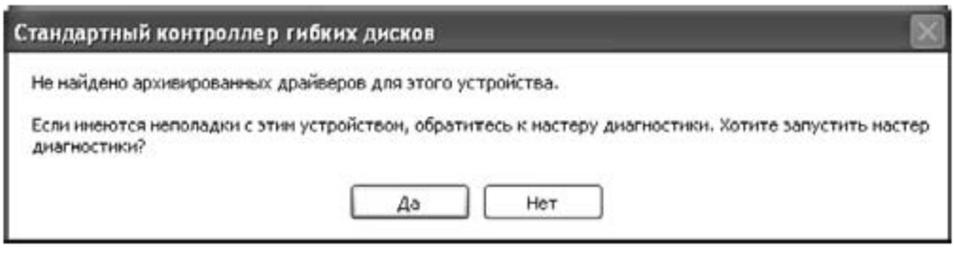


Рис. 5.11

Задания для самостоятельной работы

1. Проверить параметры подписывания драйверов на вашем компьютере. Установить опцию Предупреждать, если устанавливается драйвер без цифровой подписи.
2. Проверить наличие файла %SystemRoot%\system32\dllcache на жестком диске. Изучить опции утилиты Sfc.exe. Запустить ее, проанализировать полученные результаты.

3. Провести верификацию цифровой подписи файлов на вашем компьютере, используя утилиту sigverif. Имеются ли неподписанные файлы? Просмотреть журнал результатов проверки.
4. Попробовать заменить драйвер какого-либо устройства и выполнить функцию отката драйвера.

5.6. Безопасный режим загрузки

Если при появлении меню загрузки Windows XP нажать клавишу F8, то на экране появится меню опций отладки и дополнительных режимов загрузки. В безопасном режиме Windows 2000/XP использует параметры по умолчанию: VGA-монитор, драйвер мыши Microsoft и минимальный набор драйверов устройств, необходимый для запуска системы. Если при загрузке в безопасном режиме проблемы исчезли, следует изменить значения параметров по умолчанию и удалить лишние драйверы до полного решения проблемы.

Загрузка в безопасном режиме, концепция которого была заимствована из Windows 9x, предоставляет более удобные средства быстрого восстановления системы после ошибок, нежели Windows NT 4.0. Если несовместимый драйвер вызвал проблему при первой же перезагрузке, то в этом случае поможет опция Загрузка последней удачной конфигурации (Last Know Good Configuration). Когда пользователь выбирает из меню безопасного режима эту опцию, ОС при запуске использует информацию ключа реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet и восстанавливает всю конфигурационную информацию, сохраненную после того, как компьютер был в последний раз успешно загружен.

Если известен драйвер, вызвавший проблему, а список таких драйверов можно получить с помощью утилиты Sigverif, описанной выше, то можно попробовать другие способы быстрого восстановления. Например, можно попробовать использовать такие опции меню безопасного режима, как Безопасный режим (Safe Mode), Безопасный режим с загрузкой сетевых драйверов (Safe Mode with Networking) или Безопасный режим с поддержкой командной строки (Safe Mode with Command Prompt). После загрузки ОС можно будет удалить проблемный драйвер с помощью штатных средств Windows 2000 — Мастера оборудования (Hardware Wizard) или Диспетчера устройств (Device Manager).

Если системный и загрузочный разделы отформатированы для использования файловой системы FAT, можно попытаться загрузить компьютер с помощью загрузочной дискеты MS DOS (ли Windows 9x) и вручную удалить или переименовать файл проблемного драйвера.

5.7. Точки восстановления системы

В полностью сконфигурированной ОС крах нового приложения может привести к значительным временны́м затратам по восстановлению системы. Утрата сконфигурированной системы делает важной функцию отката Windows XP.

Перед установкой приложения можно создать точку восстановления, представляющую собой «слепок» ОС и установленных приложений, который помещается на жесткий диск. После создания точки восстановления можно устанавливать и тестировать новое приложение. Важно воздержаться от установки других приложений, пока приложение не будет полностью протестировано и не появится уверенность в том, что состояние системы не ухудшилось.

Если развитие событий пойдет по наихудшему сценарию, можно отменить установку приложения. Если это не поможет или нет уверенности в результатах удаления приложения, можно воспользоваться точкой восстановления и «откатить» систему к ее предыдущему состоянию. Этот метод позволяет восстановить ОС, но не восстанавливает жесткий диск. Файлы приложения по-прежнему остаются на нем, и их требуется удалить вручную.

Для повышения производительности системы часто рекомендуется отключить службу восстановления [5]. Это действительно целесообразно, поскольку эта функция нужна только в процессе тестирования нового приложения или обновления ОС. Однако при таком подходе перед созданием точки восстановления придется каждый раз проверять, можно ли будет ею воспользоваться.

Чтобы узнать состояние службы восстановления системы, следует щелкнуть правой клавишей мыши на значке Мой компьютер и выбрать в контекстном меню строку Свойства. Далее перейти на вкладку Восстановление системы, показанную на рис. 5.12. Если установлен флажок Отключить восстановление системы на всех дисках, то точку восстановления создать нельзя, но в этом случае экономятся ресурсы жесткого диска, оперативная память и мощность процессора. Чтобы изменить параметры отдельного диска, нужно его выделить и щелкнуть на кнопке Параметры. Появится окно (рис. 5.13), которое по-

зволит включить или выключить наблюдение и задать объем диска, резервируемый под точки восстановления.

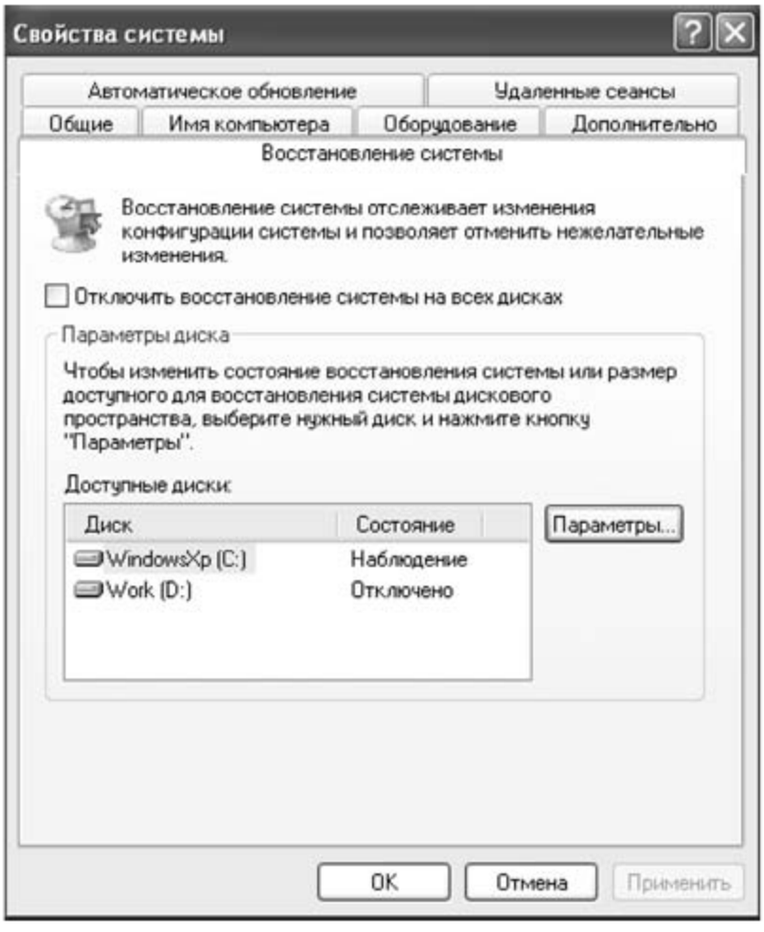


Рис. 5.12

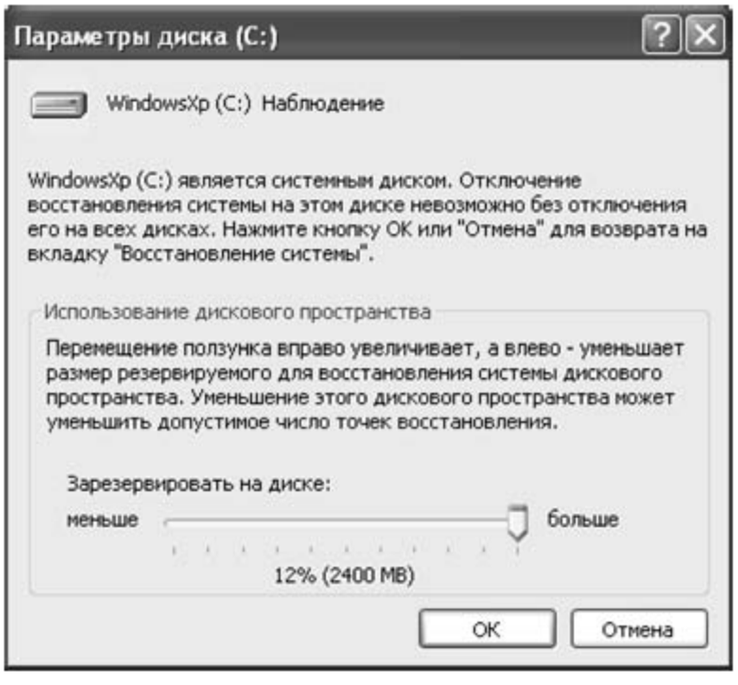


Рис. 5.13

Теоретически при решении определенных задач, таких как установка «заплат», Windows создает точки восстановления автоматически. Однако опыт показывает, что лучше создавать точки восстановления вручную. При этом пользователь будет уверен, что точка восстановления, подходящая для отката системы в случае неудачного тестирования приложения, существует, кроме того, пользователь будет точно знать ее имя.

Чтобы создать точку восстановления, нужно воспользоваться мастером восстановления системы, который запускается последовательностью команд Пуск — Программы — Специальные — Служебные — Восстановление системы. В начальном окне мастера (рис. 5.14) нужно сделать выбор между созданием новой точки восстановления и использованием существующей точки. Для создания новой точки восстановления следует установить переключатель Создать точку восстановления и щелкнуть по кнопке Далее. Теперь нужно ввести имя точки восстановления и щелкнуть по кнопке Создать (рис. 5.15). После создания точки восстановления (это займет некоторое время) мастер откроет последнее диалоговое окно с именем и датой создания точки восстановления (рис. 5.16). Щелкнув по кнопке Домой, можно вернуться в начальное окно восстановления системы, а щелкнув по кнопке Закрыть — закрыть окно.

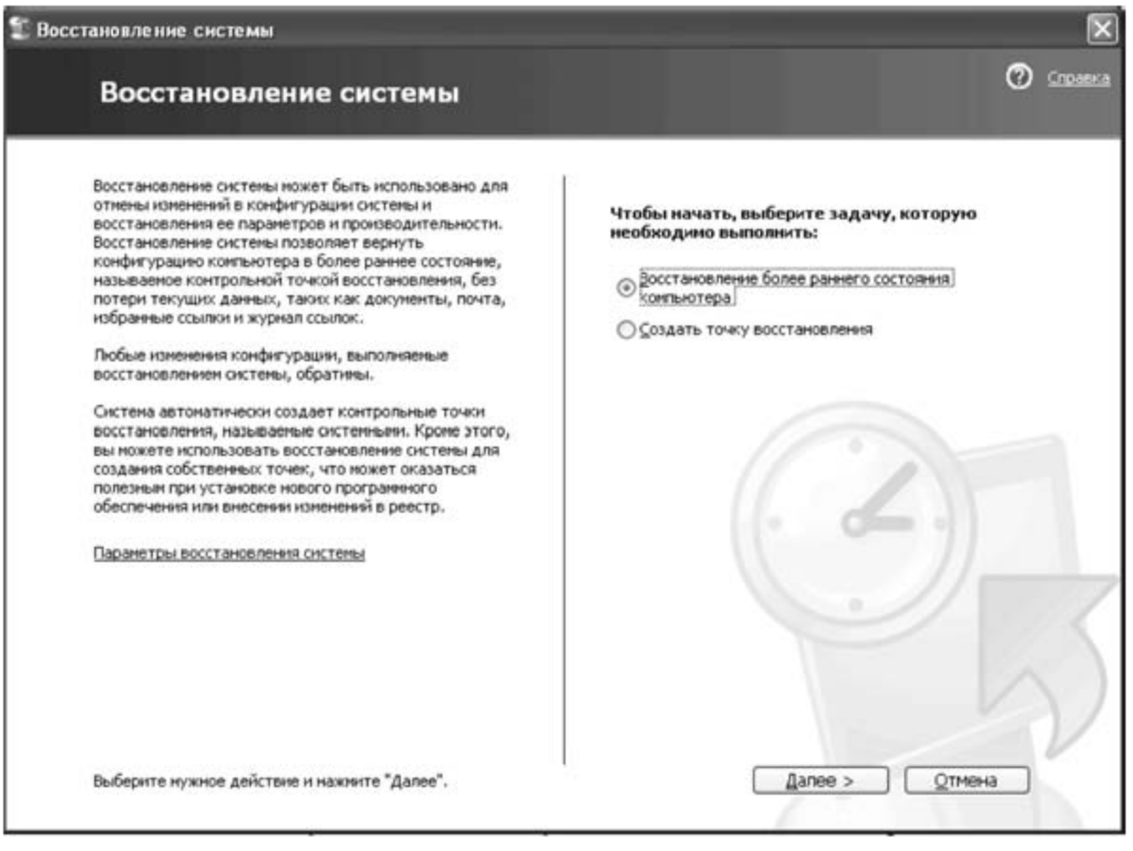


Рис. 5.14

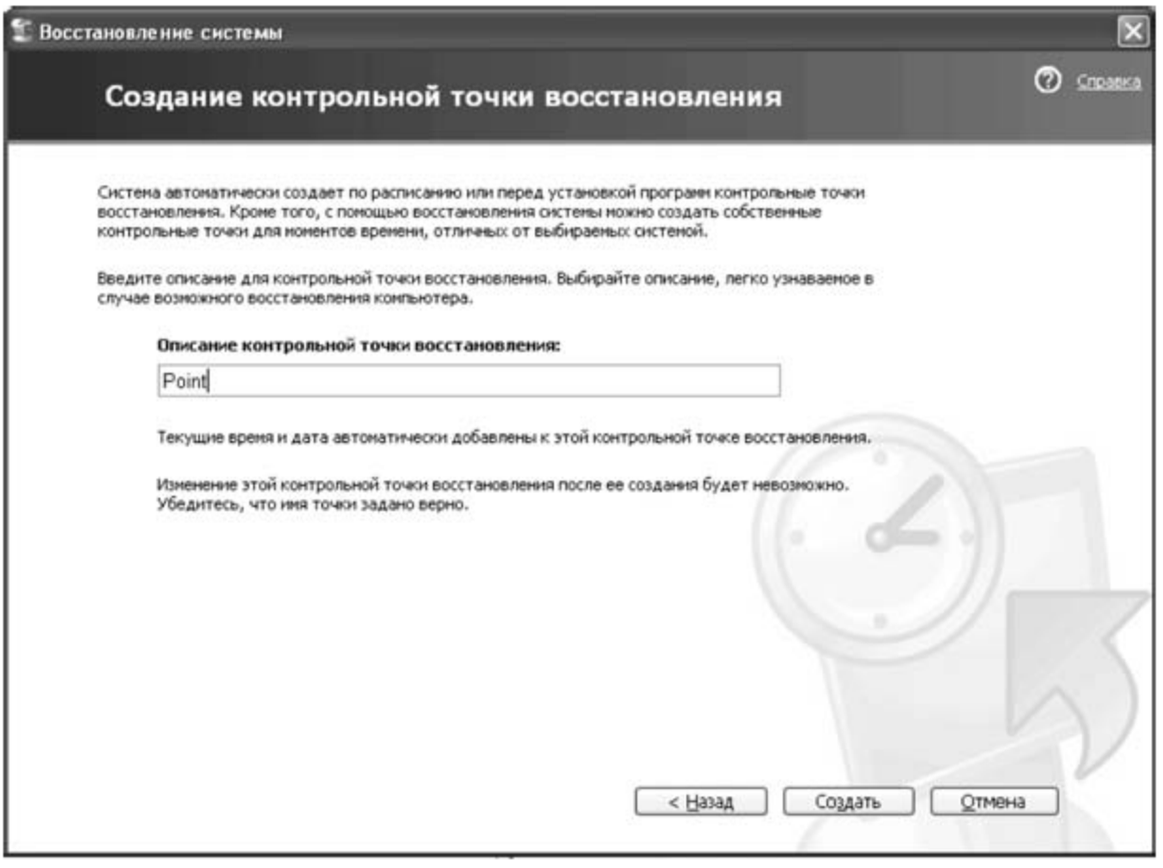


Рис. 5.15

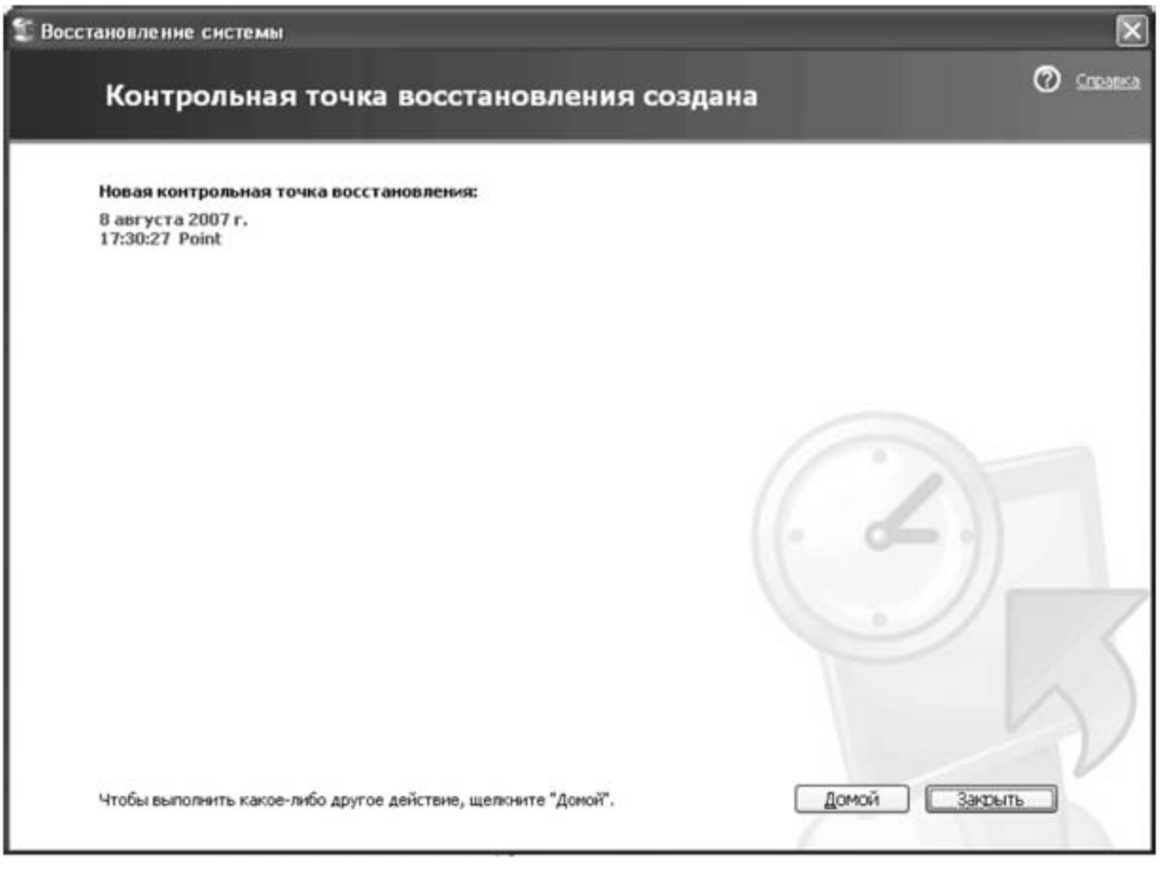


Рис. 5.16

Следует помнить, что точка восстановления — не резервная копия системы, для которой нужна полная архивация. Главная причина создания точки восстановления — подготовка места для отката системы в случае сбоя приложения, обновления или заплаты. То что жесткий диск при откате к точке восстановления не восстанавливается, означает, что использование точки восстановления и удаление приложения — разные вещи. В случае отката к точке восстановления с файлами приложения приходится разбираться вручную, поэтому лучше сначала удалить приложение, а затем воспользоваться точкой восстановления, чтобы ликвидировать вред, нанесенный реестру и другим приложениям.

Для восстановления системы нужно вызвать мастера восстановления и в его начальном диалоговом окне (см. рис. 5.14) установить переключатель Восстановление более раннего состояния компьютера и щелкнуть по кнопке Далее. На экране появится диалоговое окно Выбор контрольной точки восстановления (рис. 5.17). После выбора конкретной точки восстановления нужно щелкнуть по кнопке Далее. Мастер восстановления отобразит окно (рис. 5.18), в котором будут описаны результаты воздействия данной точки восстановления на систему. После этого можно начать процедуру восстановления системы. Перед этим следует убедиться, что все окна приложений закрыты, а затем щелкнуть по кнопке Далее. Система восстановит себя и перезагрузится. После завершения перезагрузки система сообщит, что можно отменить восстановление, если его результаты не устраивают пользователя.

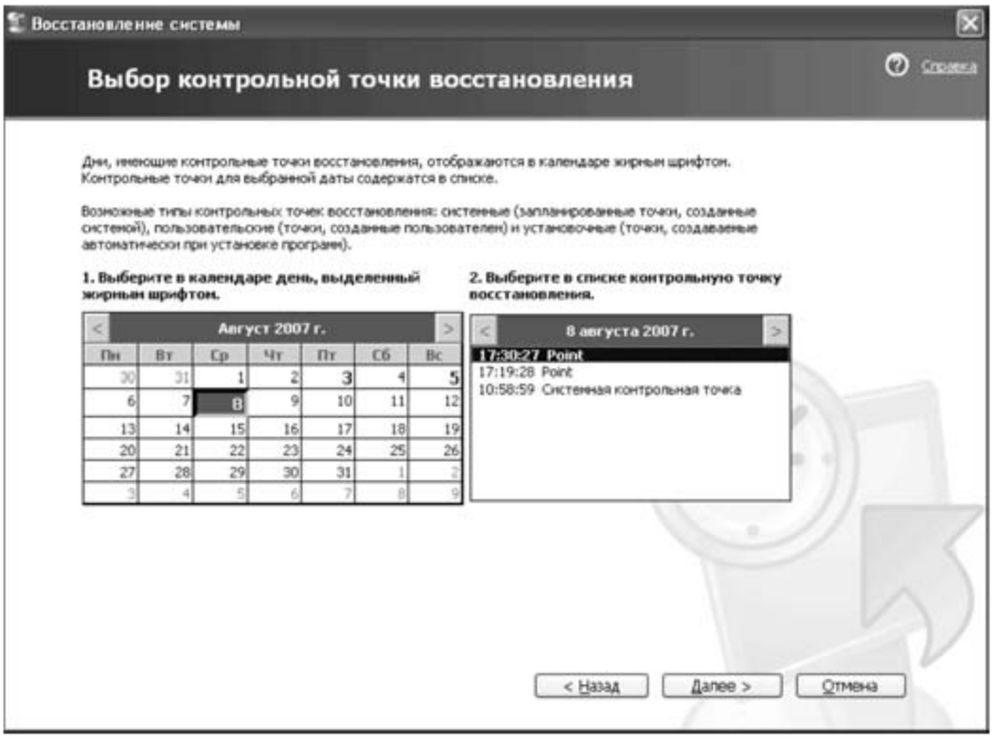


Рис. 5.17

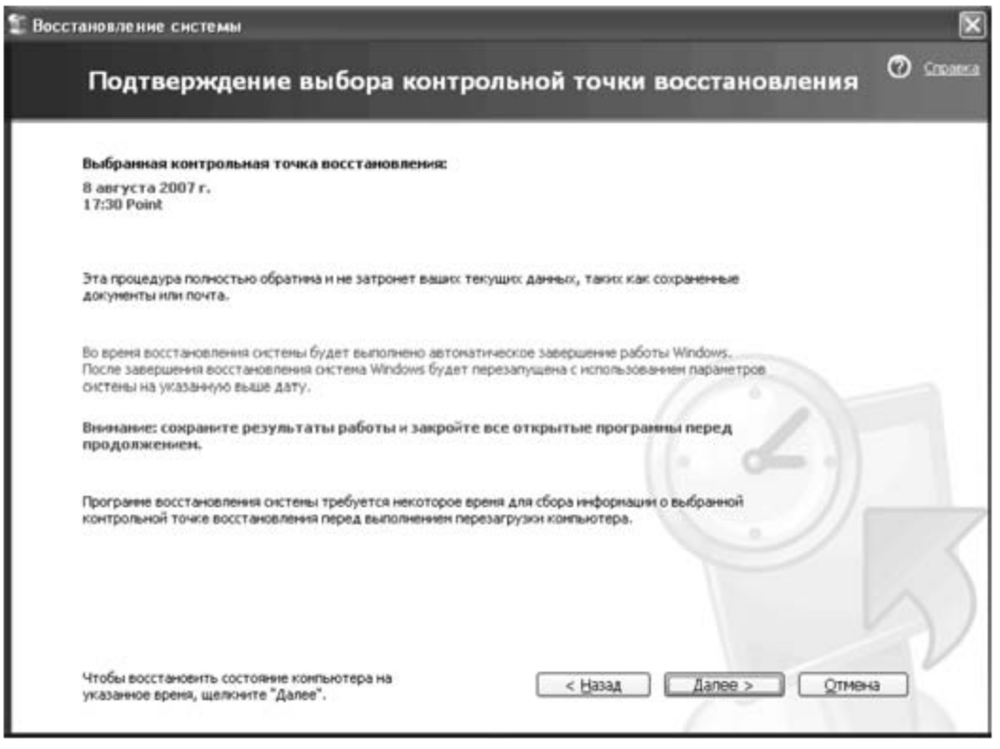


Рис. 5.18

5.8. Резервное копирование и восстановление

Для резервного копирования и восстановления данных в Windows XP/2003 используется встроенная утилита Архивация (Backup). Новая версия программы обеспечивает поддержку различных видов носителей резервной копии, что позволяет выполнять резервное копирование на любое устройство хранения информации, поддерживаемое ОС (любые жесткие диски, магнитооптические накопители и другие устройства, а не только стримеры).

К числу технологических новшеств относится технология теневых копий томов, которая позволяет создать «моментальный снимок» жесткого диска на момент начала резервного копирования. В процессе копирования будет использоваться этот моментальный снимок, остающийся неизменным, а не фактическое содержимое диска, которое в ходе резервного копирования может изменяться. Это позволяет пользователю продолжать работать в ходе выполнения резервного копирования. При этом программа архивации данных может выполнять резервное копирование открытых файлов, с которыми на данный момент времени работает пользователь (в Windows 2000 это было невозможно).

Чтобы вызвать программу архивации, нужно последовательно выполнить команды: Пуск — Программы — Стандартные — Служебные —

Архивация данных. Если программа работает в расширенном режиме, на экране появится окно, изображенное на рис. 5.19. Если программа Архивации данных работает в режиме мастера (режим легко переключается), окно программы будет выглядеть, как показано на рис. 5.20.

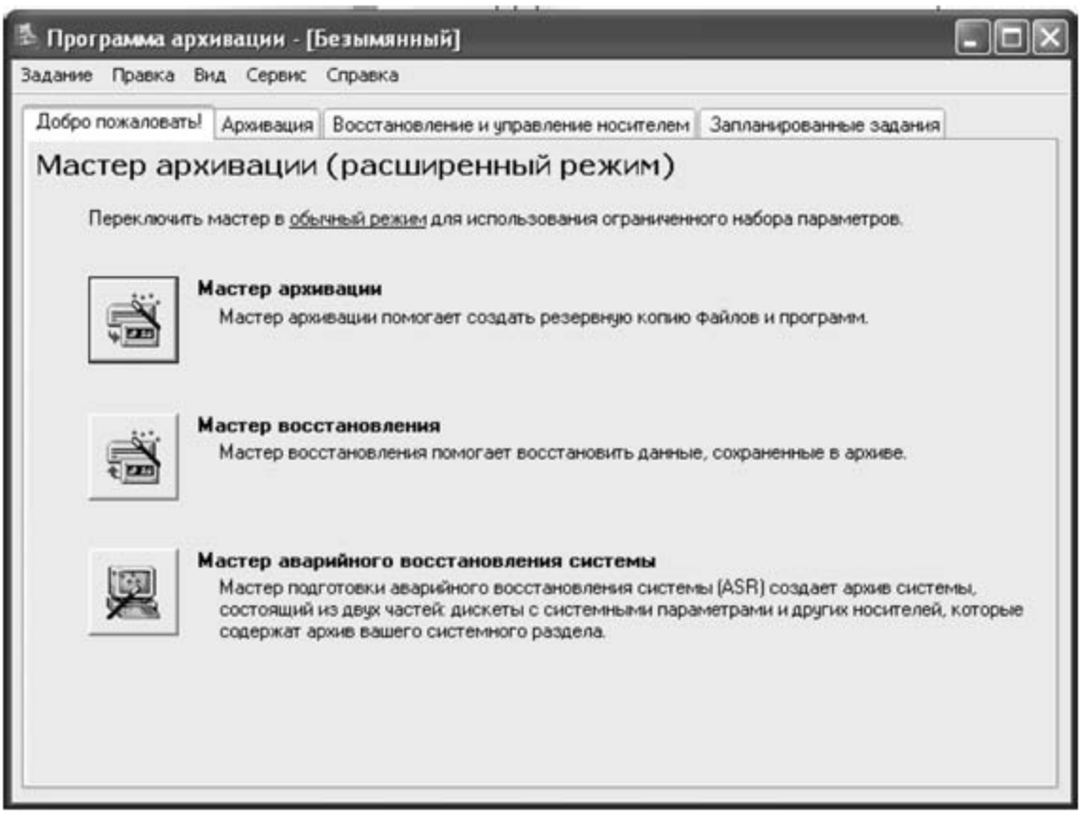


Рис. 5.19

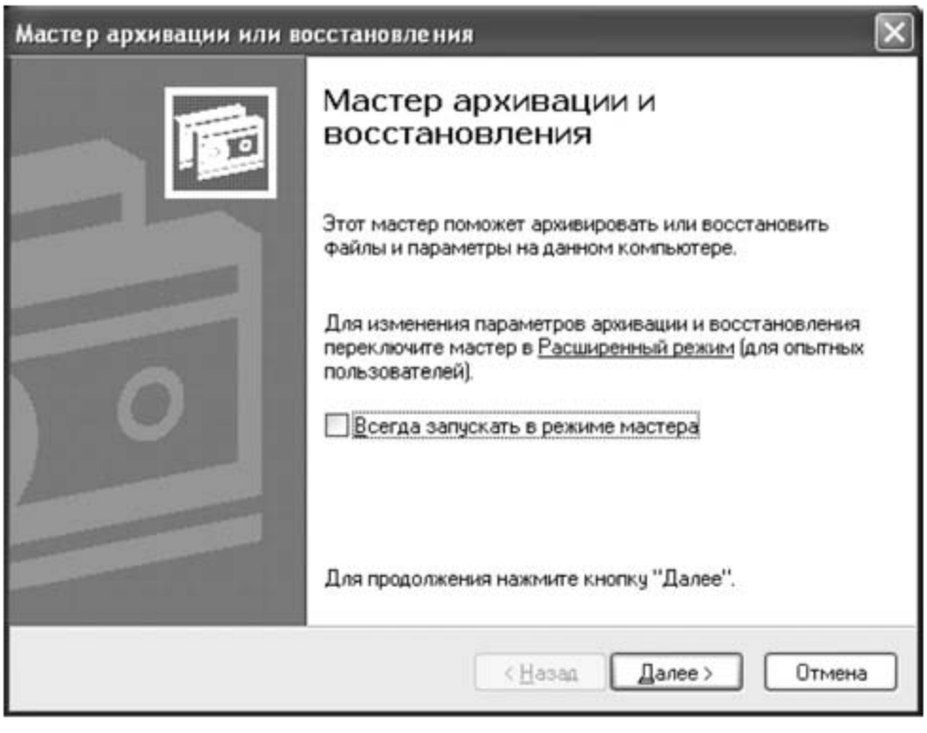


Рис. 5.20

Программа обладает большими функциональными возможностями, в том числе позволяет выполнить процедуру резервного копирования всех системных конфигурационных файлов. Чтобы упростить процедуру восстановления после сбоев, резервное копирование рекомендуется выполнять достаточно регулярно. Сделать это можно двумя способами.

Первый способ заключается в использовании Мастера архивации, который можно вызвать, нажав кнопку Мастер архивации на вкладке Добро пожаловать программы архивации (см. рис. 5.19). Нажав кнопку Далее, можно выполнить резервное копирование конфигурационных файлов, установив переключатель Архивировать только данные состояния системы (рис. 5.21) и следуя указаниям Мастера.

Второй способ связан с использованием вкладки Архивация (рис. 5.22). В этом случае из списка дисков, файлов и папок, подлежащих системному копированию, нужно выбрать опцию System State (Состояние системы). В списке Местонахождение архива нужно указать ленточное устройство, если оно имеется на компьютере, и копирование должно быть выполнено на ленту. Если такого устройства нет, по умолчанию будет установлена опция Файл. В поле Носитель архива или Имя файла нужно указать путь к файлу, в который будет выполняться резервное копирование. Дополнительные опции резервного копирования можно задать, выбрав команду Параметры из меню Сервис (рис. 5.23).

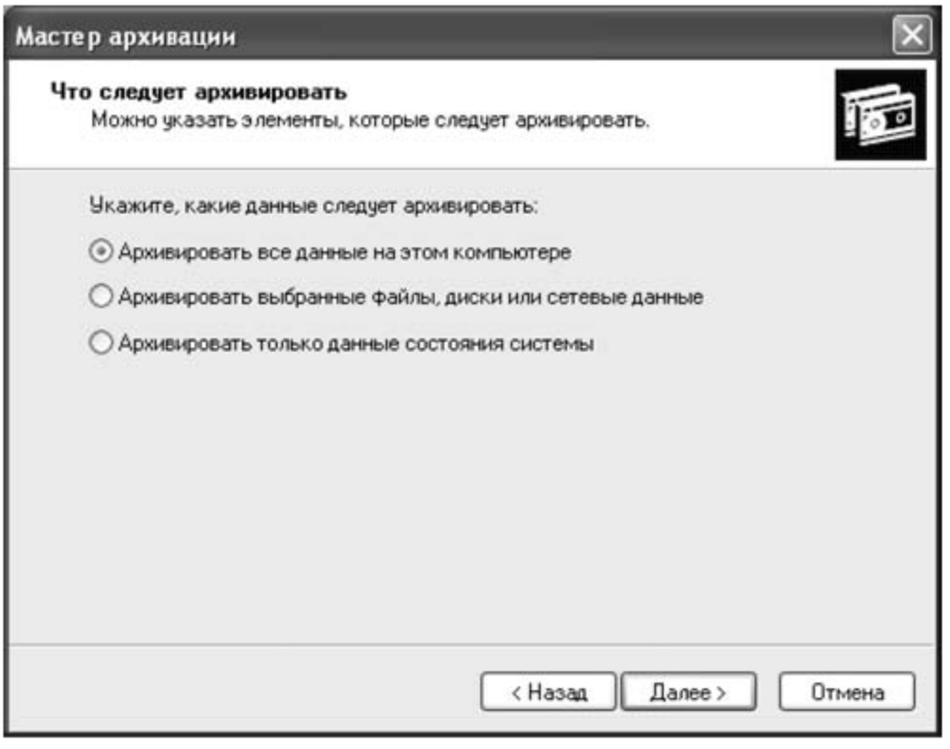


Рис. 5.21

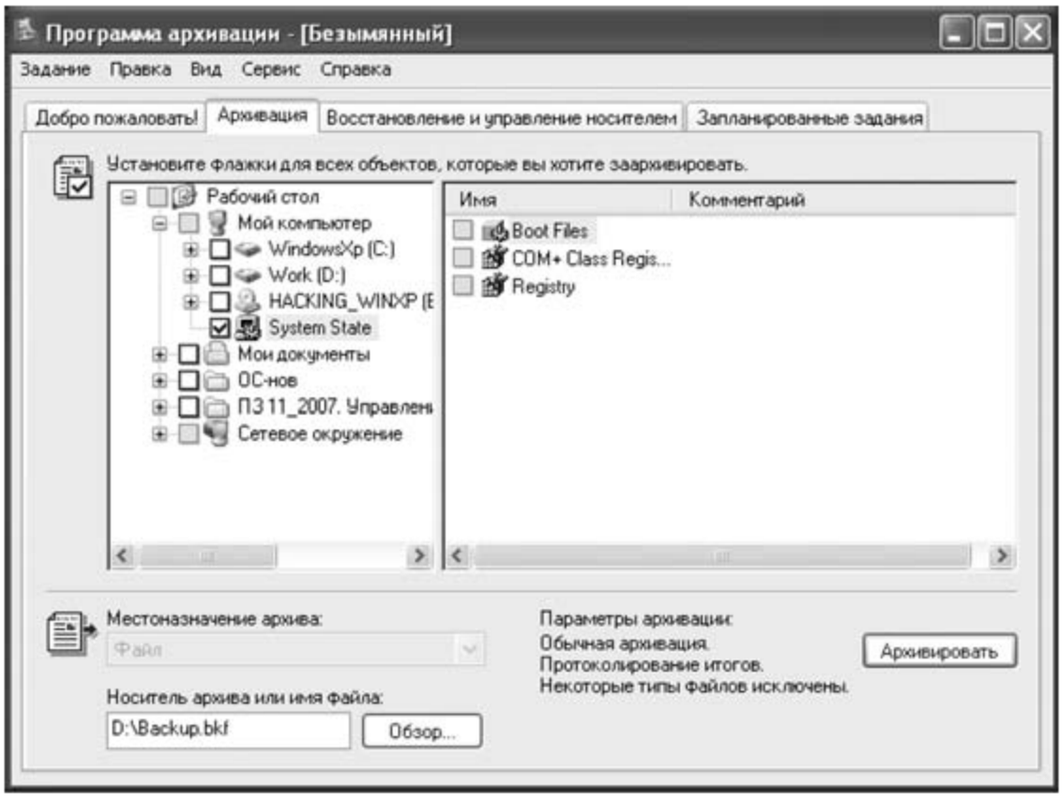


Рис. 5.22

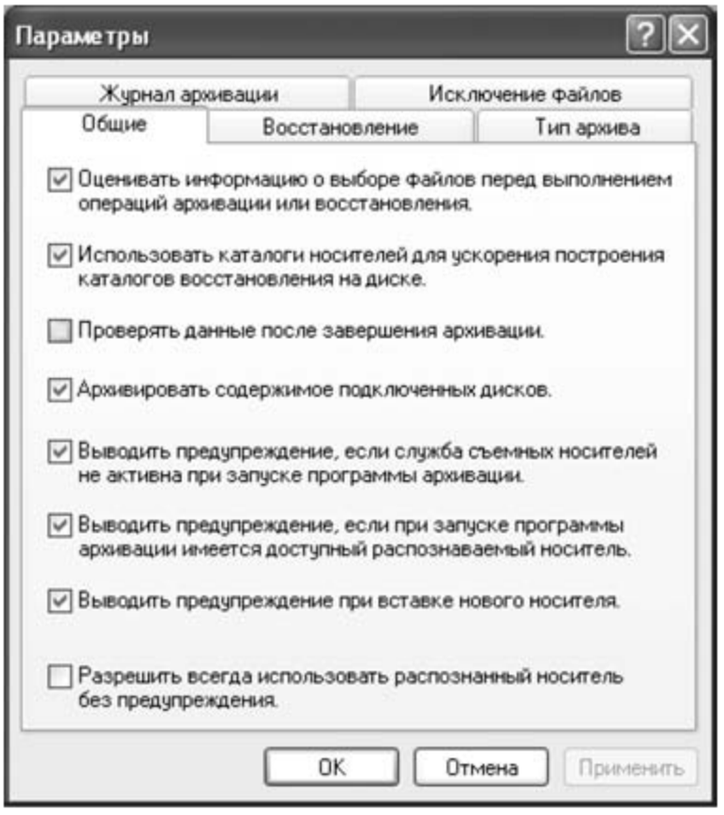


Рис. 5.23

Чтобы начать процедуру резервного копирования, следует нажать кнопку Архивировать. На экране появится окно Сведения о задании

архивации (рис. 5.24). Для установки в этом окне дополнительных опций задания на резервное копирование, нужно нажать кнопку Дополнительно. Раскроется окно Дополнительные параметры архивации (рис. 5.25). Обратите внимание на состояние флажка Автоматически архивировать защищенные системные файлы вместе с состоянием системы. Программа архивации не позволяет выполнить выборочное резервное копирование отдельных компонентов набора System State.

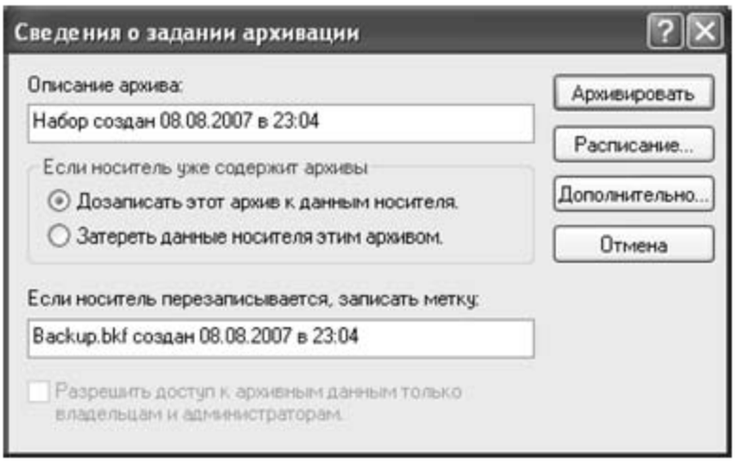


Рис. 5.24

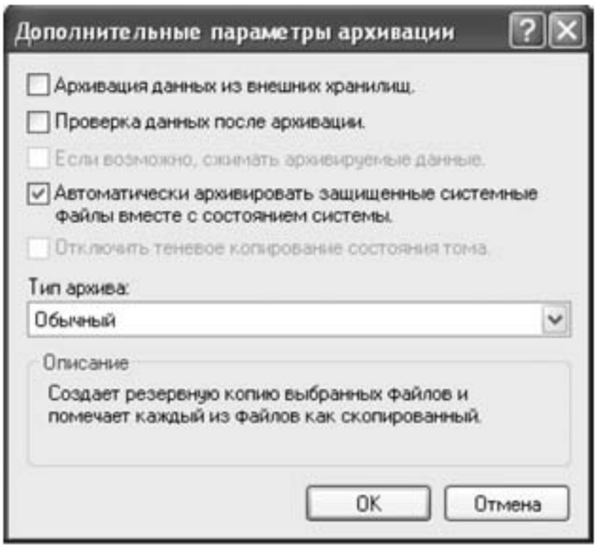


Рис. 5.25

Однако Windows XP/2003 предоставляет возможность одновременно с резервным копированием системных конфигурационных файлов выполнить резервное копирование всех защищенных файлов ОС (по сути, сделать архивную копию самой системы, что не всегда нужно). По умолчанию эта опция активизирована, однако ее можно блокировать, сбросив данный флажок. Размер архива в этом случае будет значительно меньше.

Ход архивации система отображает на экране (рис. 5.26), о завершении архивации выдается сообщение. Отчет о результатах архивации можно просмотреть, используя Блокнот (рис. 5.27).

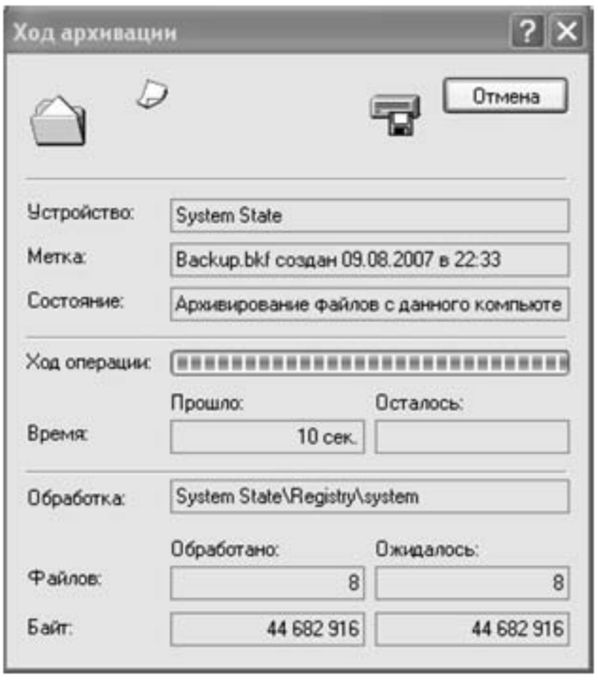


Рис. 5.26

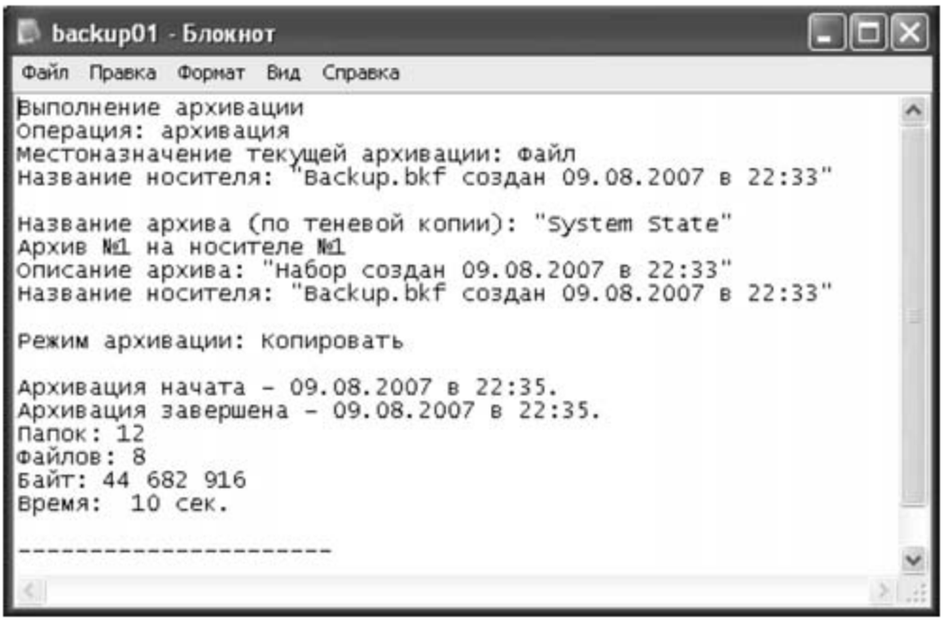


Рис. 5.27

При выполнении резервного копирования данных из набора System State система сохраняет копии файлов реестра в папке %SystemRoot%\repair\regback. В случае удаления или повреждения файлов реестра резервные копии его файлов, сохраненные в этой папке, могут быть использованы для восстановления системы без необходимости полной процедуры восстановления системных конфигурационных данных.

Если попытки восстановить поврежденную систему завершаются неудачей, работоспособная копия системных конфигурационных данных может быть использована для восстановления работоспособности. Чтобы восстановить системные конфигурационные данные, нужно вызвать программу архивации в расширенном режиме и на вкладке Восстановление и управление носителем выбрать опцию System State (Состояние системы), после чего нажать кнопку Восстановить (рис. 5.28). Через некоторое время будут восстановлены системные файлы, а также другие запрошенные данные.

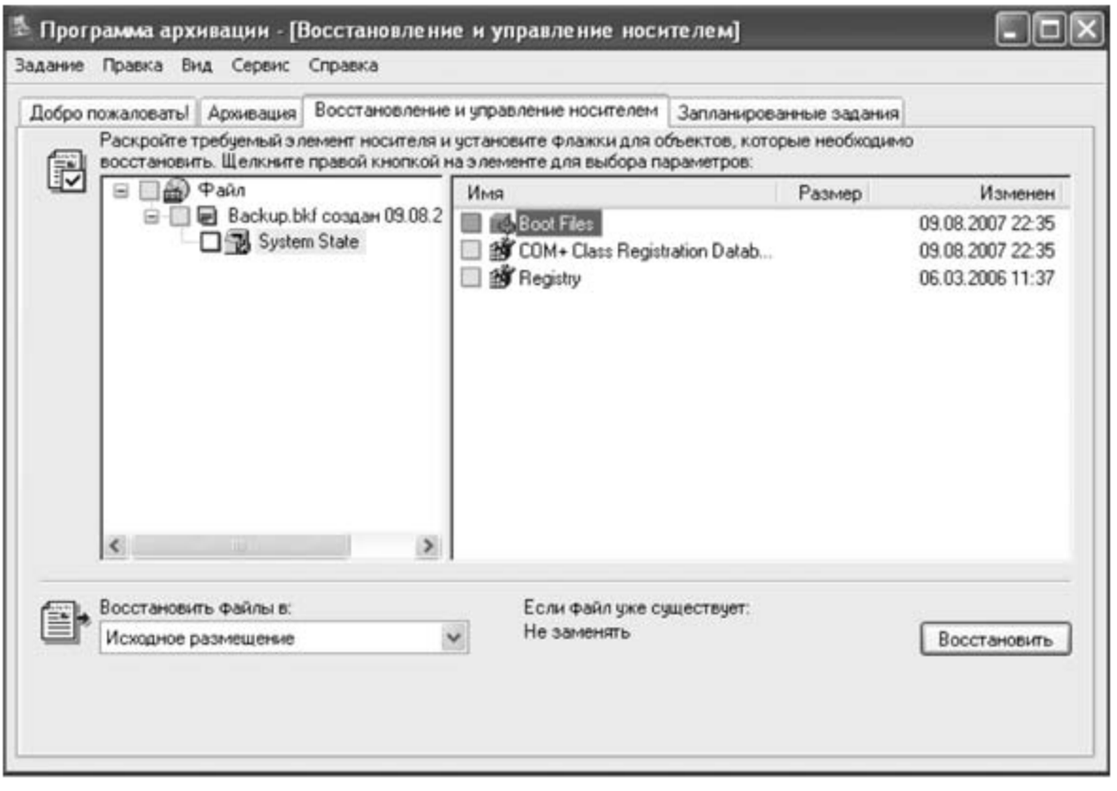


Рис. 5.28

Задания для самостоятельной работы

1. Загрузить компьютер в безопасном режиме. Как изменились возможности компьютера? Когда используется опция Загрузка последней удачной конфигурации?
2. Определить состояние службы восстановления компьютера. Создать точку восстановления системы компьютера.
3. Ознакомиться с возможностями системной программы архивации. Выполнить резервное копирование системных конфигура-

ционных файлов из набора System State. Найти архивный файл и определить его объем. Как изменится объем файла, если одновременно с резервным копированием системных конфигурационных файлов выполнить резервное копирование всех защищенных файлов ОС?

5.9. Аварийное восстановление системы

Кроме традиционных функциональных возможностей по резервному копированию и восстановлению данных, версия программы архивации, входящая в состав Windows XP, включает новую функцию подготовки к аварийному (автоматическому) восстановлению системы (Automated System Recovery, ASR). Аварийное восстановление системы представляет собой двухступенчатый процесс, который позволяет пользователю восстановить поврежденную копию Windows XP, используя для этого резервную копию конфигурационных данных операционной системы и информацию о дисковой конфигурации, сохраненную на дискете.

Для подготовки к аварийному восстановлению системы необходимо выполнить следующие действия:

1) освободить достаточный объем дискового пространства для выполнения резервного копирования (если есть стример, подготовить его);
2) запустить программу архивации в расширенном режиме и нажать кнопку Мастер аварийного восстановления системы. В первом окне программы-мастера подготовки к автоматическому восстановлению системы нажать кнопку Далее;
3) на следующей странице мастера (рис. 5.29) указать тип носителя, на который будет производиться резервное копирование, и указать путь к резервной копии. Нажать кнопку Далее;
4) в последнем окне мастера аварийного восстановления нажать кнопку Готово. Программа архивации начнет сканирование системы и составит список файлов (рис. 5.30), которые необходимо включить в состав резервной копии ASR. После ряда информационных сообщений на экране появится окно Ход архивации, сообщающее о ходе процесса создания аварийной копии системы (рис. 5.31);
5) когда процесс резервного копирования завершится (рис. 5.32), мастер аварийного восстановления предложит вставить в дис-

ковод чистую дискету, на которой будет сохранена информация о конфигурации дисковой подсистемы, в том числе сигнатуры дисков, таблица разделов, данные о томах, информация о конфигурации компьютера, а также список файлов, подлежащих восстановлению. При выполнении процедуры аварийного восстановления с этой дискеты будет считываться информация о конфигурации дисковой подсистемы и другие данные;

6) для просмотра отчета о ходе резервного копирования нужно нажать кнопку Отчет в окне Ход архивации. Отчет открывается в Блокноте (рис. 5.33).

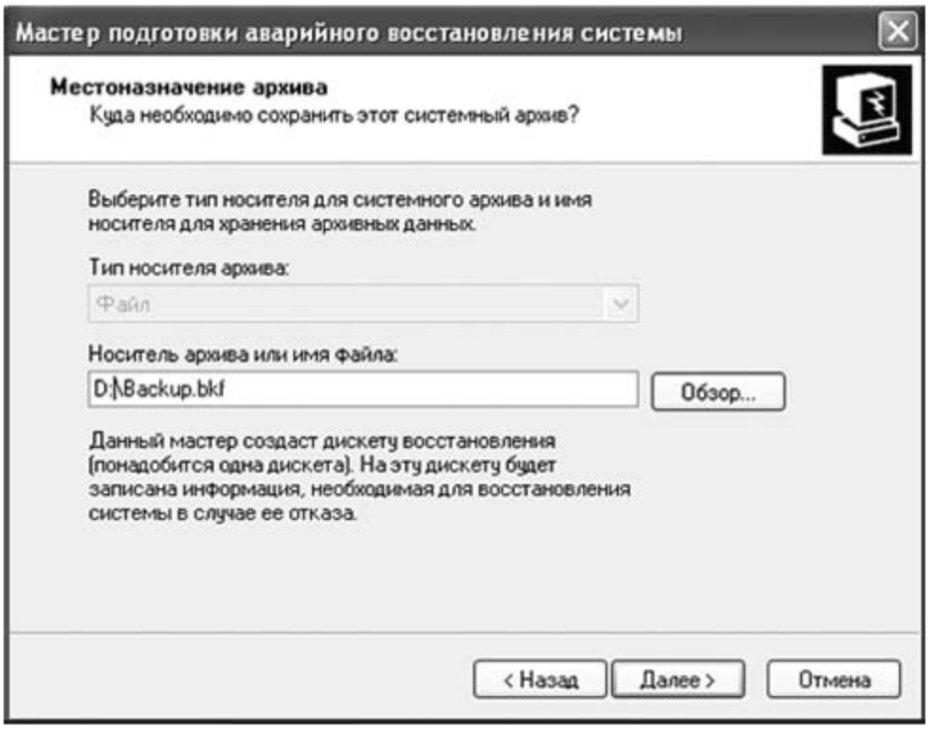


Рис. 5.29

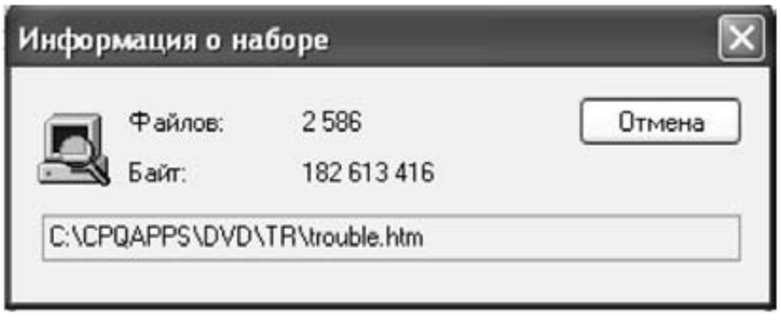


Рис. 5.30

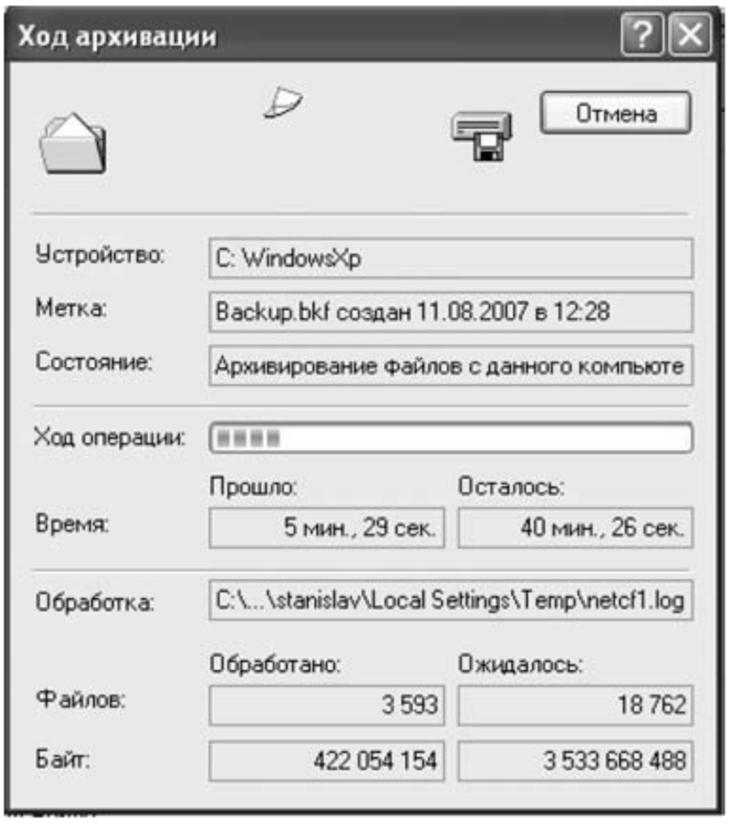


Рис. 5.31

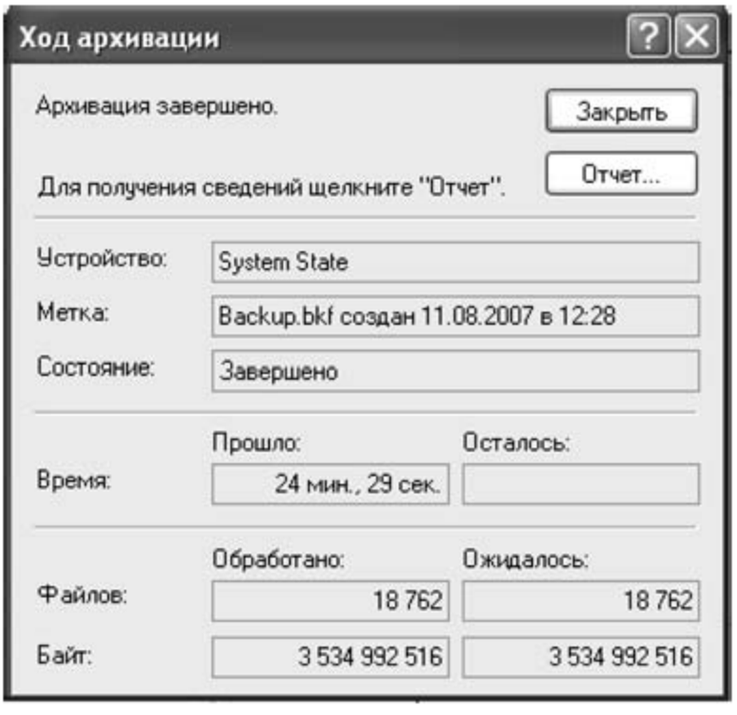


Рис. 5.32

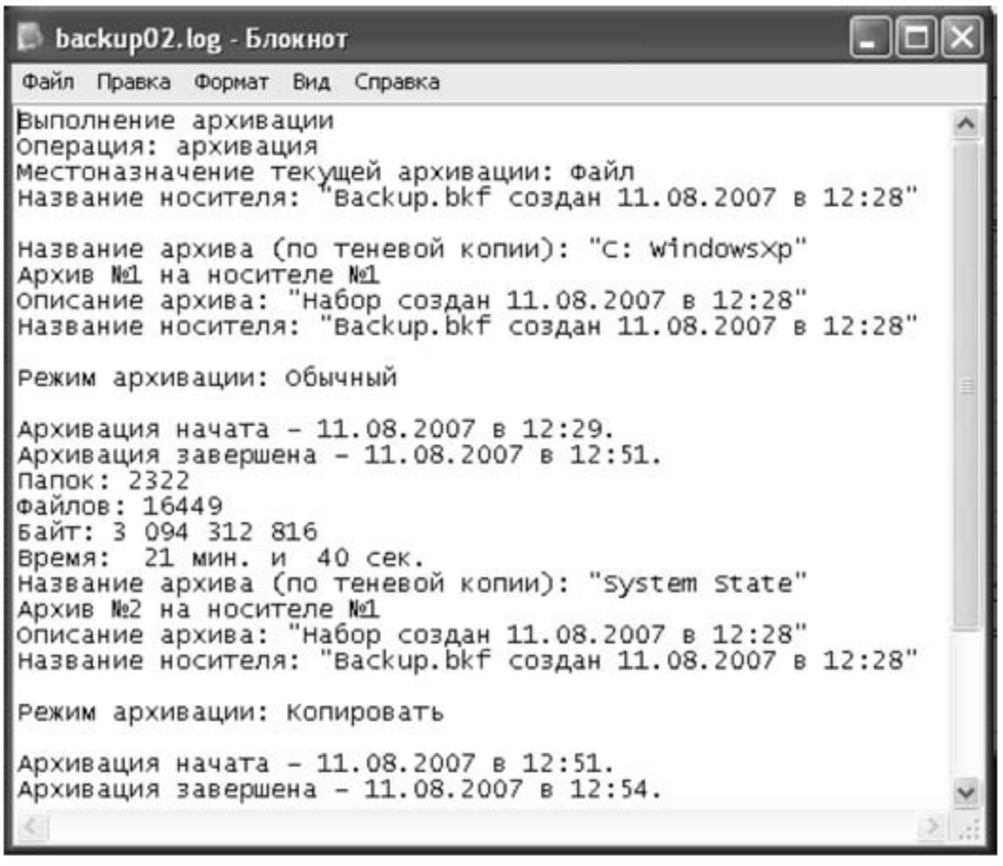


Рис. 5.33

Процесс аварийного восстановления поврежденной системы с использованием процедуры ASR запускается из программы установки ОС Windows XP Setup, что требует наличия дистрибутивного компакт-диска. Процедура ASR восстановит конфигурацию дисковой подсистемы с использованием данных, сохраненных на дискету ASR, переформатирует раздел %SystemDrive% (на котором установлена копия ОС, подлежащая восстановлению), затем переустановит на этот раздел Windows XP и восстановит конфигурационную информацию системы с резервной копии.

Следует понимать, что подготовка к ASR не является заменой регулярных процедур резервного копирования, так как она не выполняет резервного копирования файлов приложений и пользовательских данных. В процессе восстановления поврежденной системы ASR выполняет переформатирование раздела %SystemDrive%, не восстанавливая при этом файлов приложений и пользовательских данных, которые могут находиться на этом диске.

Чтобы провести процесс аварийного восстановления с помощью резервной копии ASR, следует выполнить следующие действия:

1) подготовить дискету ASR с конфигурационной информацией, соответствующей конфигурации компьютера на тот момент, когда последний раз выполнялась процедура подготовки ASR,

и компакт-диск Windows XP (имеется в виду, что резервная копия данных ASR создана на жестком диске);

2) запустить программу Windows Setup (можно просто перезагрузить компьютер с дистрибутивного диска CD_ROM);
3) нажать любую клавишу, когда на экране появится соответствующее сообщение (Press any key to boot from CD...);
4) после запуска программы Windows Setup при появлении сообщения Нажмите F2 для запуска автоматического восстановления системы (ASR) нажать клавишу F2;
5) вставить в дисковод дискету ASR, когда программа Setup предложит это сделать;
6) на экране появится сообщение: «Подготовка к автоматическому восстановлению системы». Для его отмены нажмите клавишу <ESC>;
7) если пользователь намерен продолжать процедуру, следует не реагировать на это предложение, и программа Setup последовательно отобразит следующие сообщения:
   — Запуск автоматического восстановления Windows;
   — Копирование файлов;
   — Программа установки запускает Windows.

   После этого начинается переформатирование раздела %System-Drive% и проверка остальных разделов с целью определения, не нуждаются ли они в восстановлении;
8) после форматирования и завершения проверки всех разделов ASR построит список файлов для копирования и предложит вставить носитель с резервной копией ASR. Если при подготовке к ASR резервное копирование выполнялось в файл, то этот шаг будет пропущен. Далее процедура ASR выполнит автоматическую установку Windows XP, а затем восстановит системную конфигурацию.

Если кроме всех компонентов, необходимых для выполнения аварийного восстановления системы, имеется полная резервная копия всех данных, то описанная процедура предоставляет довольно надежный способ восстановления поврежденной системы.

Выше была отмечена необходимость использования дискеты ASR для выполнения процедуры восстановления системы (нужно сказать, что при создании этой дискеты система ASR рекомендует сохранить ее). Однако возможна ситуация утраты дискеты. Ее можно восстановить. Дело в том, что файлы, которые копируются на дискету, включаются также в состав резервной копии ASR. Поэтому имеется возможность восстановления дискеты. Для этого нужно выполнить следующие действия:

1) отформатировать дискету и вставить ее в компьютер;
2) запустить программу архивации в расширенном режиме. Перейти на вкладку Восстановление и управление носителем и выбрать команду Каталогизировать архивный файл из меню Сервис (рассматривается случай, когда резервная копия была создана на жестком диске);
3) в левой части окна развернуть архив, соответствующий диску ASR, который требуется восстановить;
4) развернуть папку %SystemRoot%\repair и пометить для восстановления файлы asr.sif, asrpnp.sif, setup.log (рис. 5.34). В списке Восстановить файлы в выбрать опцию Альтернативное размещение, а в поле Альтернативное размещение указать путь к корневому каталогу дискеты (A:). Программа попросит подтвердить восстановление (рис. 5.35). Щелкнуть по кнопке OK;
5) нажать кнопку Восстановить, выбранные файлы будут скопированы на дискету. Далее дискету нужно извлечь и использовать ее в ходе восстановления ASR.

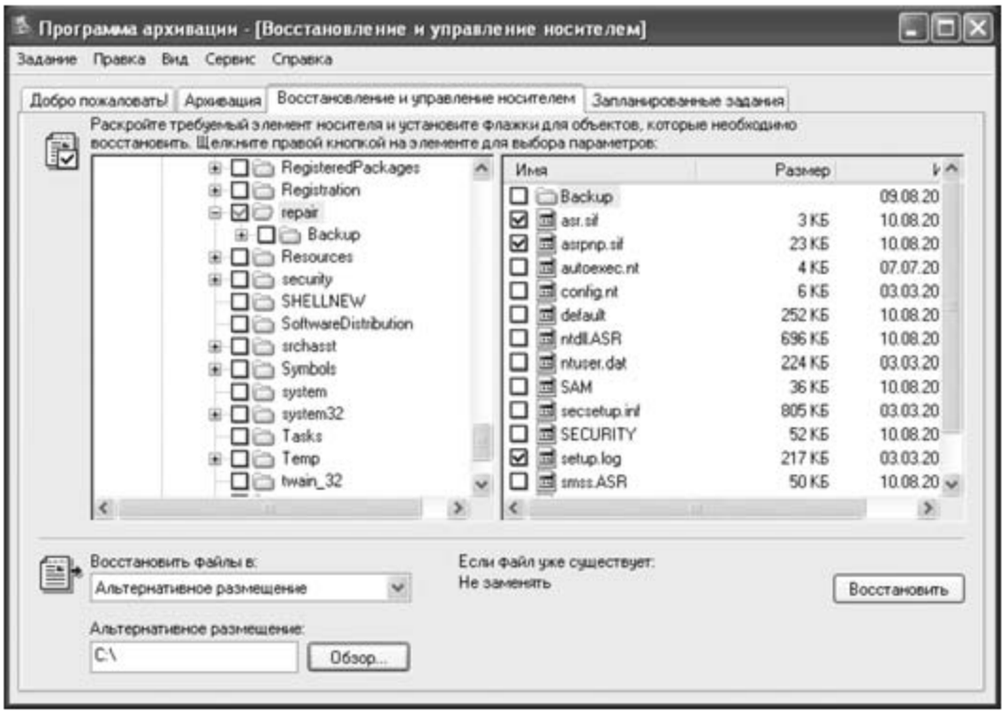


Рис. 5.34

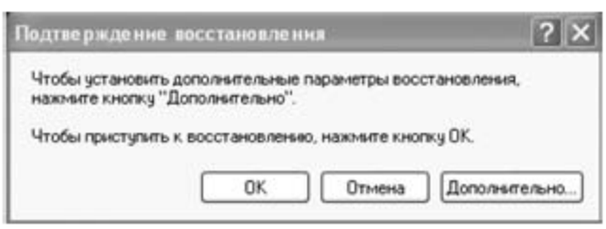


Рис. 5.35

Задания для самостоятельной работы

1. Выполнить процедуру аварийного восстановления системы. Для архивации выбрать системные файлы диска C:.
2. Восстановить дискету ASR (считаем ее утраченной), как это рассмотрено выше.

5.10. Консоль восстановления

Консоль восстановления Windows XP (Recovery Console) представляет собой исполняющую среду с интерфейсом командной строки, которая предоставляет администраторам и пользователям с административными правами необходимый минимум средств, позволяющих выполнить восстановительные процедуры в системе, имеющей проблемы с загрузкой. Используя консоль восстановления, можно запускать и останавливать сервисы, форматировать диски, выполнять чтение и запись данных на локальные жесткие диски, устранять проблемы с поврежденной главной загрузочной записью (MBR) и поврежденными загрузочными секторами. Администратор также выполняет другие административные задачи.

Существуют два способа запуска консоли восстановления. Первый способ предполагает запуск консоли восстановления из программы Windows Setup. Это можно сделать, если имеется загрузочное устройство CD-ROM или загрузочная дискета и доступ к дистрибутивным файлам для запуска программы Windows Setup. После запуска программы Windows Setup и завершения процесса начального копирования файлов появится экран, в котором Setup приглашает к инсталляции системы и предлагает на выбор: установить Windows, восстановить поврежденную копию Windows или завершить программу установки. Нужно нажать клавишу R (восстановление).

После этого будут предложены две опции по восстановлению поврежденной системы: с помощью консоли восстановления или с помощью диска аварийного восстановления. Для использования консоли нужно нажать клавишу C. После этого будет предложено указать установленную копию Windows 2000, которую требуется восстановить, а затем ввести пароль администратора для этой системы.

Второй способ запуска консоли восстановления предполагает предварительную установку консоли на жесткий диск и включение ее как одной из доступных опций в меню загрузки системы. В этом случае нужно выполнить следующие действия:

1) зарегистрироваться в Windows как администратор;
2) вставить дистрибутивный компакт-диск Windows XP в устройство CD-ROM;
3) нажать кнопку Нет (No), если будет предложено обновить ОС;
4) в режиме командной строки перейти на дистрибутивный диск Windows XP и ввести команду имя_CD-ROM\i386\winnt32.exe/cmdcons. Откроется окно, показанное на рис. 5.36. Щелкнуть по кнопке Да;
5) следовать инструкциям, появляющимся на экране.

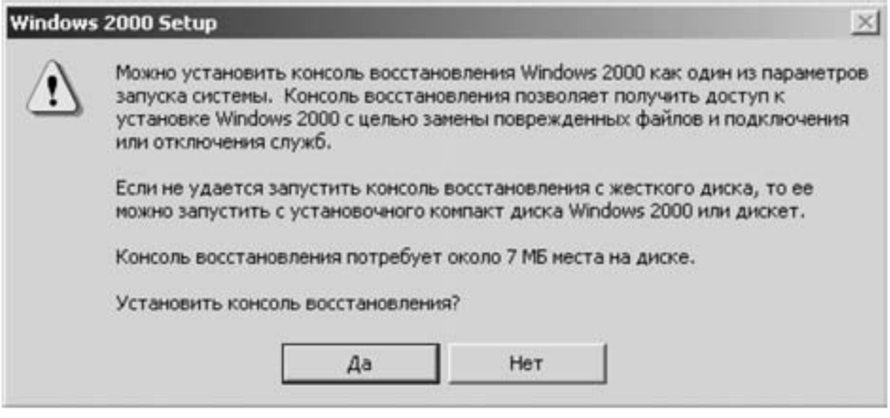


Рис. 5.36

Установка консоли потребует около 7 Мбайт дискового пространства на системном разделе. Интерфейс консоли восстановления представляет собой полноэкранный интерфейс командной строки (как в MS DOS). Для вывода доступных команд можно воспользоваться командой help. Консоль восстановления запоминает предыдущие введенные команды и позволяет их выбирать с помощью клавиш «вверх» и «вниз». Для редактирования предыдущей команды можно использовать клавишу Backspace. Для выхода из консоли используется команда exit.

Если требуется удалить консоль восстановления из списка опций, доступных в меню загрузки, нужно выполнить две операции:

1) удалить из корневого каталога системного раздела папку \cmdcons и файл cmldr;

2) открыть для редактирования файл Boot.ini и удалить в нем строку, соответствующую опции запуска консоли восстановления.

5.11. Диск аварийного восстановления

Диск аварийного восстановления (Emergency Repair Disk, ERD) создается после установки Windows NT/2000 (для Windows XP не ис-

пользуется). После каждого применения пакета обновлений, изменения данных системы, замены драйверов нужно повторно создавать ERD. Для изготовления ERD необходимо:

1) приготовить свободную дискету емкостью 1,44 Мбайт;
2) запустить программу архивации (Пуск — Программы — Стандартные — Служебные — Архивация данных). В окне программы выбрать из меню Сервис (Tools) команду Создание диска аварийного восстановления (Create an Emergency Repair Disk);
3) на экране появится окно Диск аварийного восстановления (рис. 5.37). Установить флажок Архивировать реестр в папку восстановления (Also backup the registry to the repair directory). После установки этого флажка резервное копирование реестра будет произведено в папку %SystemRoot%\repair. Процедура аварийного восстановления использует информацию из этой папки;

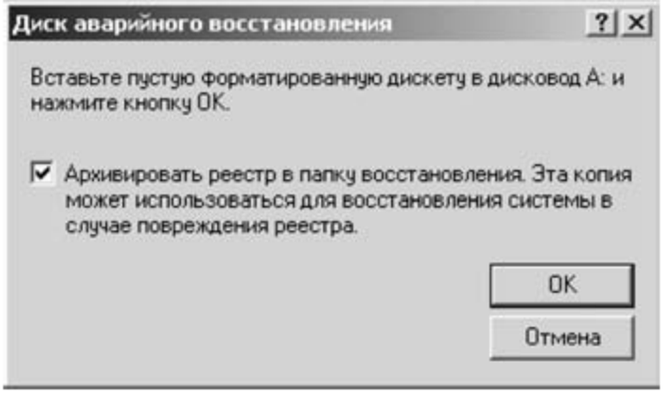


Рис. 5.37

4) нажать OK и программа архивации создаст диск аварийного восстановления.

Подготовленный ERD можно использовать для восстановления системных файлов после старта компьютера с установочных дискет или установочного компакт-диска Windows 2000. Для замены поврежденных файлов установочный диск нужен обязательно.

В начале аварийного восстановления надо выбрать один из двух режимов восстановления:

1. Ручное восстановление (Manual Repair) — нажать клавишу M. Выбирать такой режим рекомендуется только администраторам или опытным пользователям. В этом режиме можно выбрать следующие параметры:

■ анализ среды загрузки (Inspect Startup Environment) для проверки начального окружения на правильность файлов Windows 2000 в системном разделе. Отсутствующие или поврежденные файлы бу-

дут скопированы с установочного компакт-диска, включая файлы Ntldr и Ntdetect.com. Отсутствующий файл Boot.ini будет создан заново;

■ проверка системных файлов Windows 2000 (Verify Windows 2000 System files) для проверки системных файлов Windows 2000 с использованием контрольных сумм и сравнения их с файлами установочного компакт-диска Windows 2000. Если будет обнаружено несовпадение файла, появится сообщение с именем поврежденного файла и предложением его замены. В процессе восстановления проверяются также наличие и целостность загрузочных файлов, таких как Ntldr и Ntoskrnl.exe;

■ анализ загрузочного сектора (Inspect Boot Sector) для проверки, что загрузочный сектор на системном разделе ссылается на Ntldr. Процесс аварийного восстановления может только заменить загрузочный сектор на системном разделе первого жесткого диска и восстановить загрузочный сектор на системном разделе загрузочного диска.

2. Быстрое восстановление (Fast Repair) — нажать клавишу F. Это самый простой способ проведения аварийного восстановления. После выбора этого режима процесс аварийного восстановления попытается решить проблемы, связанные с системными файлами, загрузочным сектором на системном диске и средой загрузки. Файлы реестра будут проверены и восстановлены с перезагрузкой каждого раздела реестра. Если раздел не был восстановлен, он будет автоматически скопирован из каталога восстановления в папку %SystemRoot%\System32\Config.

При заражении загрузочного сектора вирусом следует загрузить компьютер с антивирусной загрузочной дискеты. Антивирусную программу можно взять с установочного компакт-диска Windows 2000 в папке \3RDPARTY\CA_ANTIV.

Если аварийное восстановление не помогло исправить ошибки, то можно провести повторную установку ОС поверх существующей. Если и это не помогает, потребуется полная переустановка системы.

5.12. Загрузочная дискета

Если опция запуска консоли не включена в меню загрузчика, то для доступа к этому средству потребуется запустить программу Windows Setup. Однако если устройство CD-ROM не является загрузочным, запуск консоли восстановления возможен только с помощью четырех установочных дискет Windows 2000, которые тоже желательно заранее подготовить.

Загрузка с дискет занимает довольно много времени, поэтому лучше использовать загрузочную дискету.

Для ее изготовления необходимо отформатировать дискету в Windows 2000/XP и скопировать на нее следующие файлы:

- Ntldr;
- Ntdetect.com;
- Boot.ini;
- Bootsect.dos — если используется мультизагрузочная система и нужно обеспечить возможность загрузки с дискеты также Windows 9x или DOS;
- Ntbootdd.sys — если в файле Boot.ini применяется синтаксис scsi ().

Загрузочная дискета поможет выполнить загрузку компьютера в следующих случаях:

1) повреждены главная загрузочная запись и/или загрузочный сектор раздела на системном разделе;
2) возникли проблемы с диском, на котором находится системный раздел;
3) выполняется переконфигурирование жестких дисков и нужно обеспечить возможность запуска Windows 2000 в случае возникновения проблем.

Задания для самостоятельной работы

1. Установить на компьютере консоль восстановления. Проверить, изменилось ли меню загрузки системы.
2. Запустить консоль восстановления. Используя команду help, ознакомиться со списком команд, доступных при работе с консолью.
3. Изготовить загрузочную дискету и проверить ее работоспособность.

ГЛАВА 6

СИСТЕМНЫЙ РЕЕСТР И СИСТЕМНЫЕ СЛУЖБЫ

6.1. Назначение и структура реестра

Реестр Windows XP (Windows registry) представляет собой реляционную базу данных, в которой аккумулируется вся необходимая для нормального функционирования компьютера информация о настройках ОС, а также об используемом совместно с Windows программном обеспечении и оборудовании [7]. В процессе развития ОС Windows реестр заменил конфигурационные файлы (INI-файлы) и снял неудобства и ограничения, связанные с их использованием. В отличие от ОС предыдущих поколений информация в реестре Windows XP хранится в бинарном (двоичном) представлении, что позволяет не только помещать в реестр значительно больший объем информации, но и увеличить скорость работы с ним.

С реестром взаимодействуют следующие компоненты ОС [12, 17]:

1) программы установки (Windows Setup). Каждый раз при запуске программы Windows Setup или установочных программ для аппаратных и программных средств происходит добавление в реестр новых конфигурационных данных. Правильно разработанные программы установки считывают информацию реестра, чтобы определить, присутствуют ли в системе компоненты, обязательные для успешного завершения установки. Реестр позволяет приложениям совместно использовать конфигурационную информацию и предоставляет им больше возможностей взаимодействия между собой;
2) распознаватель (Recognizer). При запуске компьютера распознаватель аппаратных средств помещает в реестр список обнаруженных устройств. На компьютерах с процессорами Intel распознавание аппаратных средств осуществляется программой Ntdetect.com и ядром ОС Ntoskrnl.exe;

3) ядро ОС (Ntoskrnl.exe). При старте системы ядро извлекает из реестра сведения о загружаемых драйверах устройств и порядке их загрузки. Кроме того, оно передает в реестр информацию о себе (номер версии и др.);
4) драйверы устройств. Драйверы обмениваются с реестром параметрами загрузки и конфигурационными данными. Каждый драйвер устройства должен сообщить об используемых им системных ресурсах, включая аппаратные прерывания и каналы DMA. Приложения и драйверы устройств могут считывать эту информацию из реестра, предоставляя пользователям интеллектуальные программы инсталляции и конфигурирования;
5) административные средства. Эти средства, в том числе утилиты панели управления и оснастки, собранные в меню Администрирование, представляют собой удобные и безопасные (в части внесения ошибок) средства модификации реестра. Редактор реестра также полезен для просмотра реестра и внесения изменений в конфигурацию системы;
6) пользовательские профили (user profiles). Windows NT/2000/XP обеспечивают возможность создания множества пользовательских профилей. Вся информация, относящаяся к конкретному пользовательскому имени и ассоциированным с ним правам, хранится в реестре;
7) аппаратные профили (hardware profiles), или профили оборудования. Реестр в отличие от INI-файлов позволяет хранить множественные аппаратные конфигурации. Аппаратный профиль представляет собой набор инструкций, с помощью которого можно указать операционной системе, драйверы каких устройств должны загружаться при запуске компьютера.

Работа с реестром (его редактирование) позволяют пользователю [18]:

■ решать проблемы, возникающие в процессе эксплуатации прикладного программного обеспечения, гибко настраивать режимы работы приложений;

■ устранять неполадки в работе оборудования, вызванные некорректным использованием ресурсов ОС или драйверов различных устройств;

■ настраивать параметры и ограничения пользовательской среды Windows, изменять заданные по умолчанию характеристики ОС;

■ управлять быстродействием компьютера;

■ перераспределять ресурсы ОС по усмотрению администратора компьютера;

■ управлять конфигурацией компонентов Windows и системных сервисов, что способствует оптимизации работы ОС в зависимости от назначения компьютера и решаемых пользователем задач.

При редактировании системного реестра Windows XP в специальных программах он представляется в виде единой базы данных, имеющей жесткую иерархическую структуру. Однако на физическом уровне реестр неоднороден и состоит из множества файлов, каждый из которых отвечает за собственный объем представленной в этой базе информации[1]. Следует учесть и тот факт, что некоторые из отображаемых в реестре сведений вообще не хранятся в виде файлов на диске, а помещаются в оперативную память компьютера в процессе загрузки ОС и утрачиваются в момент отключения питания. Такие разделы реестра называются энергозависимыми.

В частности, к энергозависимым разделам реестра относится ветвь HKEY_LOCAL_MACHINE\HARDWARE[2], в которой находятся сведения об оборудовании компьютера и системных ресурсах, назначенных устройствам: запросах на прерывание (IRQ), каналах прямого доступа к памяти (DMA) и диапазонах памяти ввода-вывода (I/O Range).

Другие части реестра, хранящие данные о базовой конфигурации ОС, ее настройках и параметрах, содержатся в системной папке %SystemRoot%\System32\Config. Файлы, хранящие данные о профилях пользователей, находятся в папке % SystemRoot%\Profiles. Данные, относящиеся к конкретным настройкам системы для каждого пользователя и данные об их персональной конфигурации рабочей среды, представлены в папках %Drive%\Documents and Settings\%UserName%, где %Drive% — имя раздела диска, на котором установлена ОС, а %UserName% — папка, имя которой соответствует имени одного из зарегистрированных в системе пользователей.

Дополнительные сведения о локальных пользователях системы по умолчанию содержатся в папке %Drive%\Documents and Settings\LocalService, а данные о настройках системы для удаленных пользователей — в папке %Drive%\Documents and Settings\NetworkService.

Следует отметить, что для нормальной работы с реестром Windows знания его устройства на физическом уровне не требуется, поскольку программы, предназначенные для взаимодействия с реестром, позволяют его редактировать как единый файл. На логическом уровне реестр представляется иерархической структурой из четырех ступеней.

---

[1] Постоянные компоненты реестра называются ульями (hives) или кустами.

[2] Не является ульем.

Верхний уровень образуют так называемые *ветви* (Hive Keys), которые принято обозначать аббревиатурой HKEY_, где за символом подчеркивания следует название ветви. Всего в реестре пять ветвей: HKEY_CLASSES_ROOT; HKEY_CURRENT_USER; HKEY_LOCAL_MACHINE; HKEY_USERS и HKEY_CURRENT_CONFIG.

Второй ступенью являются *разделы*, или *ключи* (Keys). Они отображаются в программе Редактора реестра в виде подпапок ветвей HKEY_. Функционально ключи можно разделить на две условные категории: определяемые системой (их менять нельзя) и определяемые пользователем (эти имена могут быть изменены администратором, и такие изменения не приведут к каким-либо фатальным последствиям).

Ступенью ниже следуют *подразделы* (Subkeys). Их имена также могут быть определены системой или пользователем. Последней ступенью в иерархической структуре системного реестра являются параметры (Values) — элементы реестра, содержащие саму информацию, определяющую работу ОС и компьютера в целом. Параметры представляют собой цепочку имя параметра — тип данных — значение.

Типы данных, определенные для параметров реестра, приведены ниже в табл. 6.1. Для значений параметров реестра вне зависимости от того, к какому типу данных они относятся, в программе Редактор реестра имеется набор встроенных мастеров, позволяющих легко изменять данные любого типа.

Таблица 6.1

Наименование	Тип данных	Описание
REG_BINARY	Двоичный	Аппаратные компоненты используют информацию в виде двоичных данных. Редакторы реестра отображают ее в шестнадцатеричном формате
REG_DWORD	Числовой	Числа (4 байта), параметры драйверов устройств и сервисов. Редакторы реестра отображают ее в двоичном, шестнадцатеричном и десятичном формате
REG_DWORD_LITTLE_ENDIAN	»	Эквивалент REG_DWORD с младшим байтом в начале числа
REG_DWORD_BIG_ENDIAN	»	Эквивалент REG_DWORD со старшим байтом в начале числа
REG_SZ	Строковый	Описания компонентов

Окончание

Наименование	Тип данных	Описание
REG_EXPAND_SZ	Строковый	Расширяемая строка данных. Текст, содержащий переменную, которая может быть заменена при вызове со стороны приложения
REG_MULTI_SZ	Многострочковый	Списки текстовых строк в формате, удобном для восприятия
REG_LINK	Строковый	Символическая ссылка Unicode. Предназначен для внутреннего использования.
REG_NONE	Нет типа	Параметр не имеет определенного типа данных
REG_RESOURCE_LIST	Строковый	Список аппаратных ресурсов
REG_RESOURCE_REQUIREMENTS_LIST	»	Список необходимых аппаратных ресурсов
REG_FULL_RESOURCE_DESCRIPTOR	»	Дескриптор (описатель) аппаратного ресурса

6.2. Средства управления реестром

Основным инструментом администрирования реестра Windows является программа Редактор реестра. Кроме Редактора реестра в Windows XP имеются средства консоли управления — специальные средства, позволяющие управлять многими аппаратными, программными и сетевыми компонентами Windows. Часть программных средств, позволяющих изменять настройки реестра, расположена в Панели управления Windows XP. Практически все параметры ОС, связанные с окружением пользовательской среды, ее возможностями и ограничениями, можно изменить при помощи Редактора системных политик. Возможно и управление реестром Windows из командной строки посредством консольного интерпретатора команд CMD. Можно также создать командный (пакетный) файл, включающий набор команд среды CMD. Сам Редактор реестра поддерживает собственный набор команд, последовательность которых может быть записана в файл. Путем использования таких файлов та или иная информация может быть автоматически добавлена в реестр.

По умолчанию Редактор реестра (утилита Reggdit.exe) в процессе установки ОС копируется в каталог %SystemRoot%. Для его запуска нужно выполнить команды Пуск — Выполнить, а затем в поле Открыть набрать команду regedit и щелкнуть на кнопке OK. Рабочее окно Редактора реестра разделено на две части (рис. 6.1). В левой (Панель разделов) отображается дерево ветвей, разделов и подразделов, из которых состоит реестр Windows XP. В правой части отображаются параметры, назначенные для выбранного элемента реестра (в данном случае — Select).

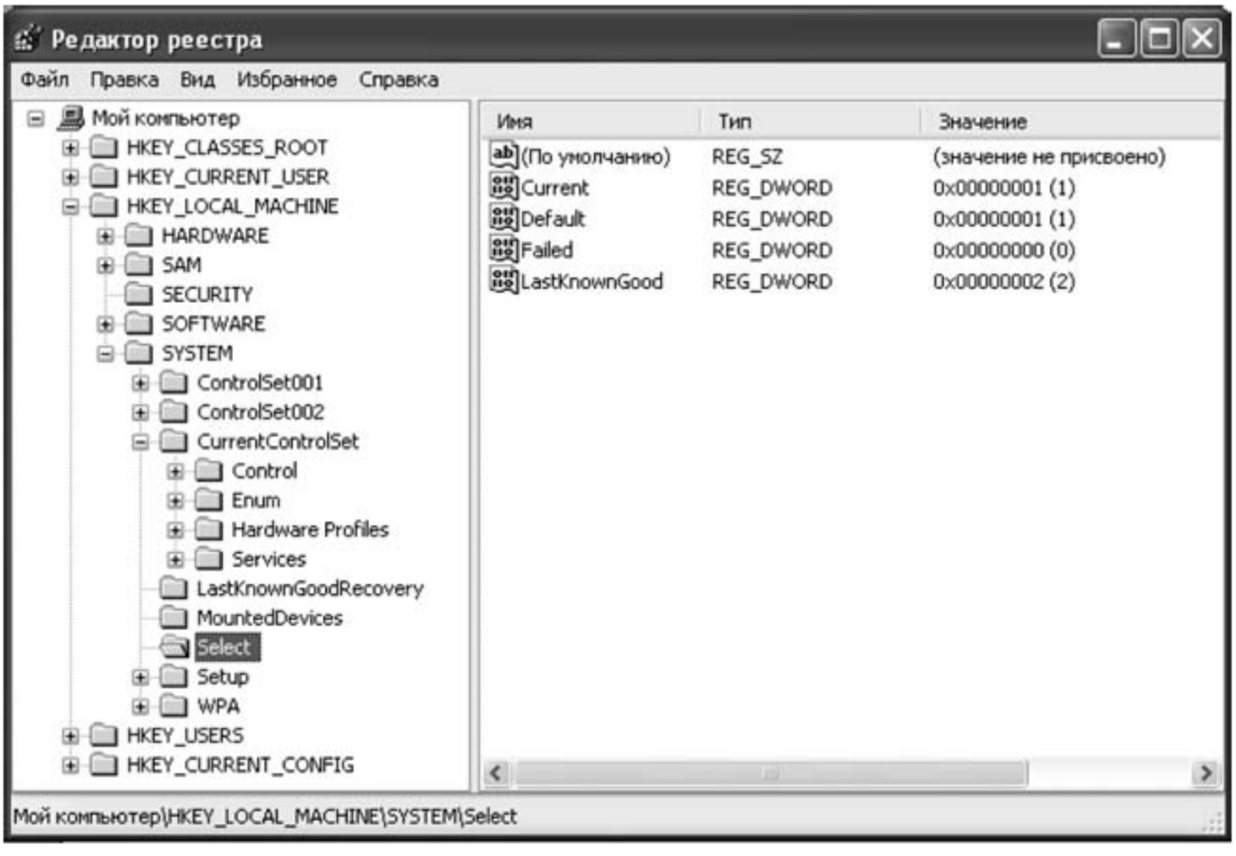


Рис. 6.1

Перемещаясь при помощи мыши по иерархической структуре реестра в левой части окна, в правой части можно просматривать параметры каждого из разделов. Чтобы изменить значение какого-либо параметра, нужно дважды щелкнуть мышью по значку. В открывшемся окне (рис. 6.2) можно установить требуемое значение параметра.

В нижней части окна расположена строка состояния, которая отображает размещение выделенного элемента реестра. Включение и отключение строки состояния осуществляются установкой или снятием флажка Строка состояния в меню Вид. Строка меню содержит пять подменю: Файл, Правка, Вид, Избранное и Справка.

Меню Файл осуществляет операции экспорта-импорта файла реестра и его отдельных компонентов, вывод содержимого какого-либо фрагмента реестра на печать или открытие для редактирования файла реестра других компьютеров, расположенных в локальной сети.

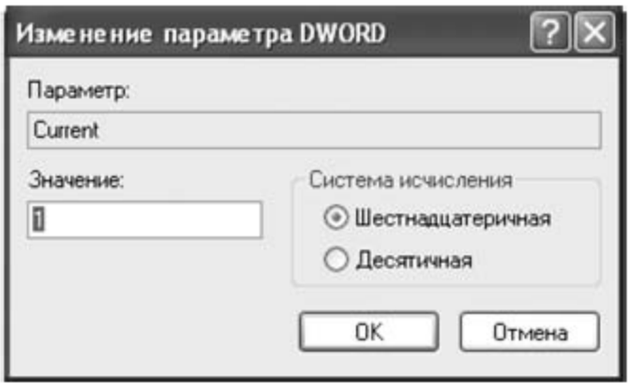


Рис. 6.2

Меню Правка позволяет выполнять операции создания, удаления, редактирования, переименования логических элементов реестра, выполнять процедуры поиска, настраивать режимы безопасности.

Меню Вид управляет настройками интерфейса Редактора реестра.

Меню Избранное устанавливает закладки на различные разделы и подразделы реестра для последующего быстрого перехода к ним.

Меню Справка вызывает встроенную справочную систему Редактора реестра.

Большое удобство для работы пользователя предоставляют встроенные мастера редактора, позволяющие удобно редактировать параметры, хранящие значения с различными типами данных. Таких мастеров четыре — по количеству типов данных, характерных для значений параметров реестра. Соответствующий мастер вызывается автоматически двойным щелчком по параметру реестра. Вызванный мастер позволяет изменить значение параметра, причем значение может быть изменено только с сохранением текущего типа данных. Пример окна мастера для настройки параметра DWORD показан на рис. 6.2.

Работа с реестром при помощи программы Редактор реестра состоит из следующих действий:

- просмотра разделов и подразделов (в данном случае механизм работы сходен с порядком работы пользователя в программе Проводник);
- поиска информации в реестре (меню Правка — Найти — Найти далее);
- быстрого перехода к выбранному разделу реестра (используется механизм закладок, меню Избранное — Добавить в избранное);

■ добавления новых разделов и параметров реестра[1] (меню Правка — Создать — Раздел) или (Правка — Создать, далее из меню выбирается тип создаваемого параметра, например Строковый параметр);

■ переименования, удаления и копирования раздела или параметра реестра (щелчок правой клавишей мыши по объекту переименования и использование контекстного меню).

Для того чтобы назначить пользователям или группам пользователей разрешения на доступ к какому-либо разделу реестра, нужно выполнить следующие действия (обладая правами администратора):

1) открыть редактор реестра;
2) выделить ветвь, раздел или подраздел, для которого требуется настроить разрешения;
3) в меню Правка выбрать команду Разрешения. На экране появится диалоговое окно настройки параметров безопасности для выбранного раздела реестра (рис. 6.3);

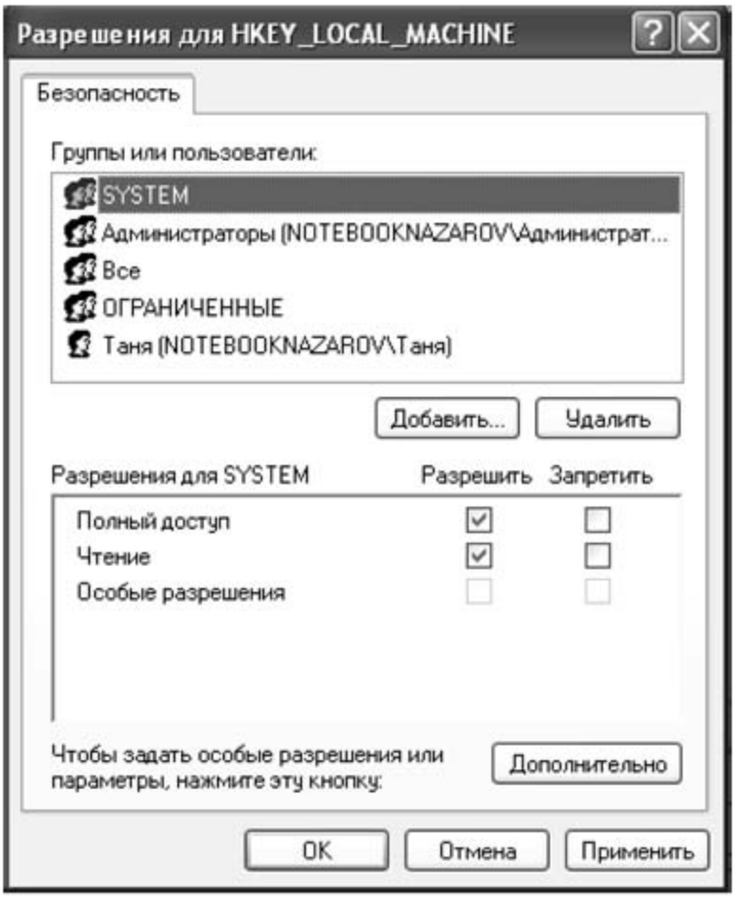


Рис. 6.3

[1] Следует соблюдать осторожность, так как изменения могут повлиять на работу приложений, оборудования компьютера и ОС.

4) щелкнуть мышью по кнопке Дополнительно. Откроется окно Дополнительные параметры безопасности (рис. 6.4);

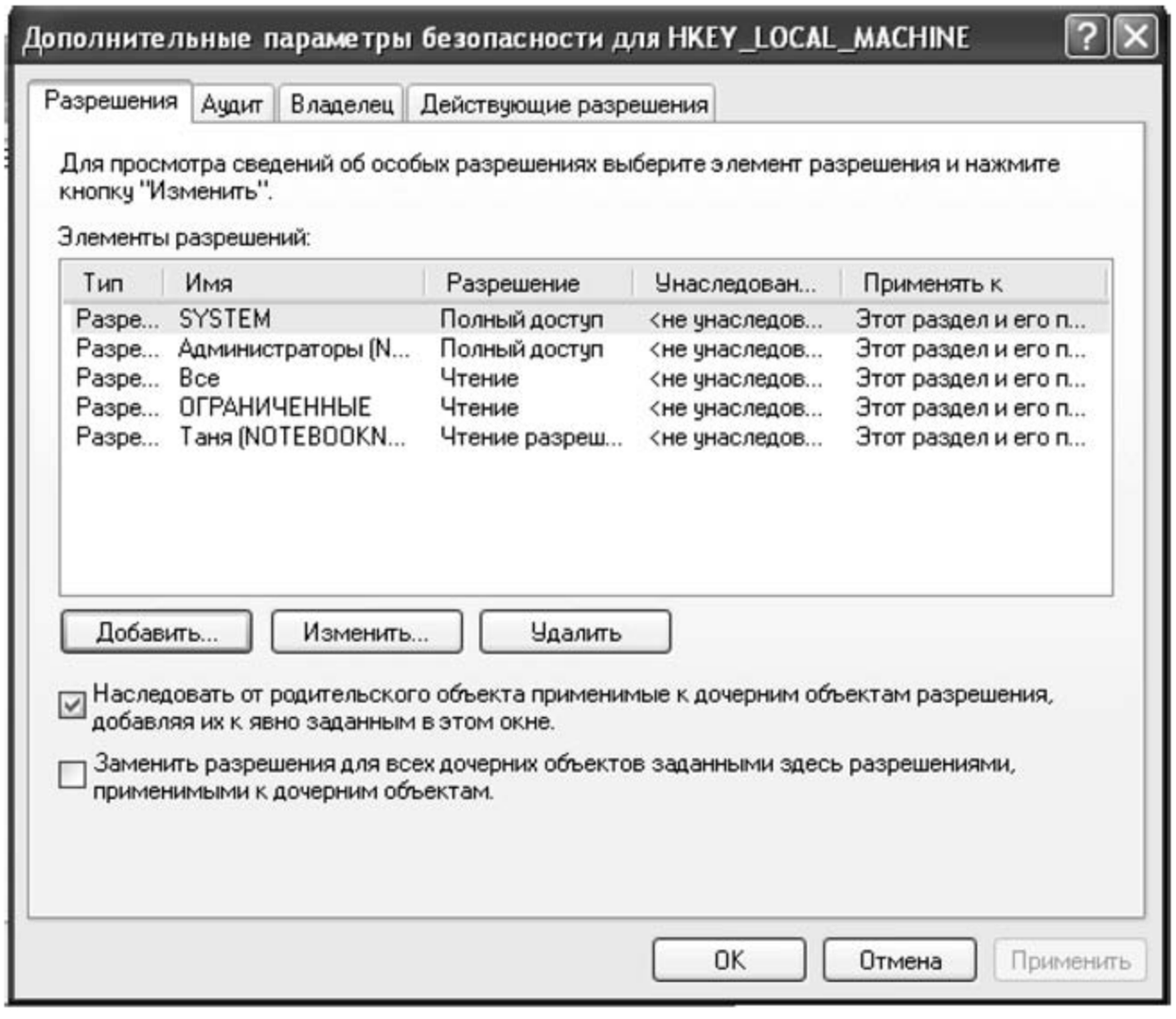


Рис. 6.4

5) на вкладке Разрешения этого окна в списке Элементы разрешений выбрать компьютер, пользователя или группу. Чтобы добавить новый элемент в список нужно воспользоваться кнопкой Добавить;
6) если необходимо настроить все разрешения для выбранного в списке пользователя или группы вручную, запретив автоматическое наследование разрешений от родительского объекта, следует сбросить флажок Наследовать от родительского объекта применимые к дочерним объектам разрешения, добавляя их к явно заданным в этом окне;
7) для изменения списка разрешений для выбранного пользователя или группы щелкнуть по кнопке Изменить. В списке Разрешения открывшегося окна установить (сбросить) флажки, соответ-

ствующие разрешениям для данного пользователя или группы (рис. 6.5). Если флажок затенен, это означает, что разрешение наследуется. Чтобы изменить настройки такого разрешения, нужно выполнить действия, указанные в п. 6.

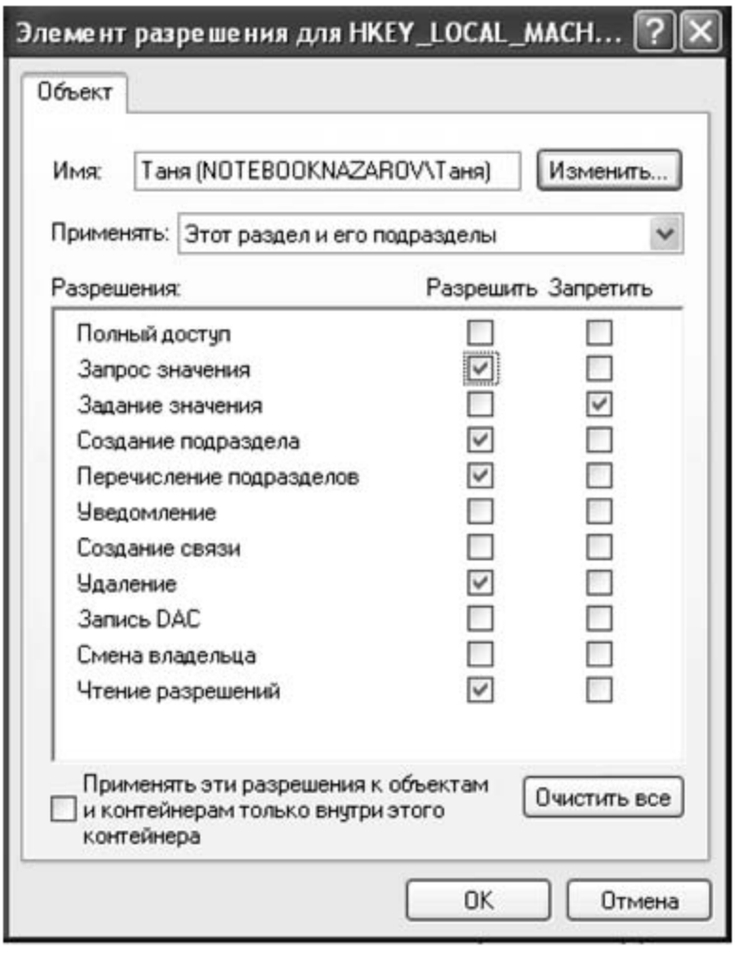


Рис. 6.5

Для того чтобы дочерние объекты реестра автоматически наследовали разрешения от родительских объектов, в диалоговом окне Дополнительные параметры безопасности следует установить флажок Заменить разрешения для всех дочерних объектов заданными здесь разрешениями, применимыми к дочерним объектам. Изменения в настройках разрешений вступят в силу после нажатия на кнопку Применить, а затем OK.

Контроль над действиями пользователей с реестром администратор может осуществлять посредством аудита. Для этого администратор должен выполнить следующие действия:

1) открыть Редактор реестра;
2) выделить ветвь, ключ или подраздел, для которого нужно настроить параметры аудита;

3) в меню Правка выбрать команду Разрешения. На экране появится окно настройки параметров безопасности для выбранного раздела реестра;
4) щелкнуть по кнопке Дополнительно. Откроется окно Дополнительные параметры безопасности;
5) перейти на вкладку Аудит;
6) двойным щелчком выбрать в списке Элементы аудита пользователя или группу. Если список пуст, необходимо выполнить следующие действия:

■ щелкнуть мышью по кнопке Добавить. На экране появится окно Выбор: Пользователи или Группа,

■ для выбора типа объекта щелкнуть на кнопке Типы объектов и выбрать требуемый вариант (в нашем случае — Пользователи) и щелкнуть по кнопке OK. Произойдет возврат в окно Выбор (Пользователи или Группа), в котором можно ввести имена пользователей, после чего щелкнуть по кнопке Проверить имена OK. На экране появится окно Элементы аудита для выбранной ветви реестра (рис. 6.6) (в нашем случае HKEY_LOCAL_MACHINE);

7) выбрав пользователя, следует установить флажки Успех или Отказ в группе Доступ.

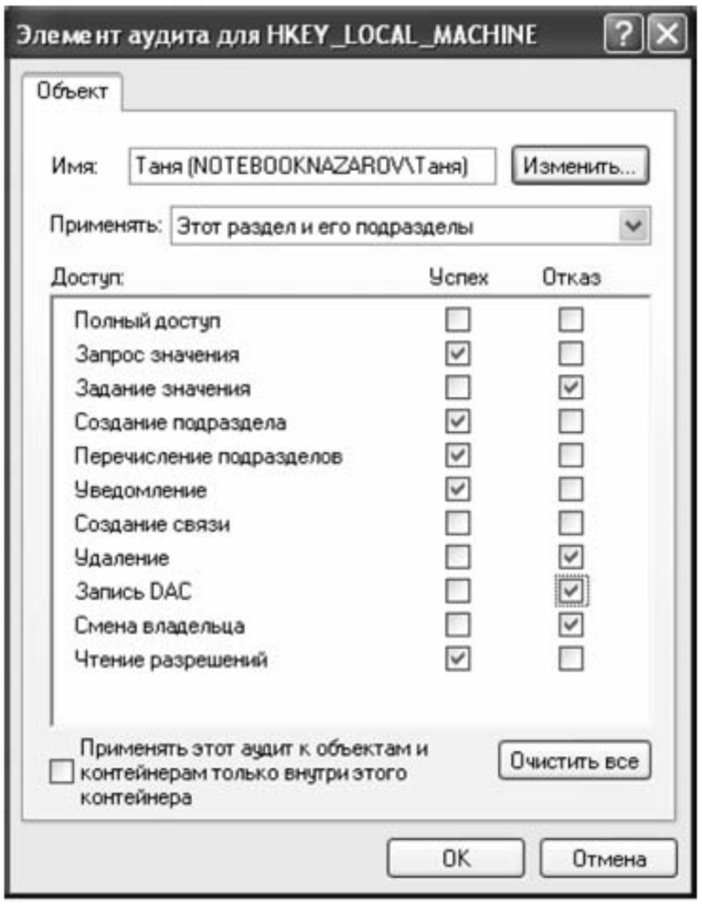


Рис. 6.6

Значения параметров для различных событий аудита реестра приведены в табл. 6.2. Результаты аудита можно просматривать в системном журнале Безопасность с помощью оснастки Просмотр событий.

Таблица 6.2

Событие	Тип аудита
Запрос значения	Аудит всех попыток чтения параметров из раздела реестра
Задание значения	Аудит всех попыток задания значений параметров в разделе реестра
Создание подраздела	Аудит всех попыток создания подразделов в данном разделе реестра
Перечисление подразделов	Аудит всех попыток определения подразделов данного раздела реестра
Уведомление	Аудит событий уведомления из подраздела реестра
Создание ссылки	Аудит попыток создания символьной ссылки в указанном реестре
Удаление	Аудит попыток удаления объектов реестра
Запись DAC	Аудит всех попыток записи в избирательную таблицу управления доступом для данного раздела
Смена владельца	Аудит всех попыток смены владельца выбранного раздела
Чтение разрешений	Аудит всех попыток открытия необязательной таблицы управления доступом для данного раздела

Задания для самостоятельной работы

1. Запустить Редактор реестра. Просмотреть ветвь HKEY_LOCAL_MACHINE. Найти параметры, относящиеся к ОС (список загруженных драйверов, сведения о загрузке Windows) и оборудованию компьютера (тип шины компьютера, объем доступной памяти и др.).
2. Войти в систему с правами администратора. Назначить пользователям или группе пользователей разрешения на доступ к какому-либо разделу реестра.
3. Установить аудит над действиями пользователей, которым дано разрешение на доступ к реестру.

6.3. Резервное копирование и восстановление реестра

Реестр является жизненно важным компонентом системы, и единственная ошибка в его редактировании может привести к проблемам в работе с ОС, включая и загрузку. Если реестр окажется поврежденным, а в распоряжении пользователя не окажется работоспособной и пригодной к использованию резервной копии, то во многих случаях не остается ничего другого как переустановить ОС. Поэтому прежде чем начать редактирование реестра, необходимо выполнить его резервное копирование.

Windows XP предоставляет множество методов резервного копирования, часть из которых рассматривается далее.

6.3.1. Экспорт файла реестра или его компонентов

Чтобы экспортировать в файл системный реестр целиком или какой-либо из его компонентов, потребуется выполнить следующие действия:

1) открыть программу Редактор реестра;
2) выбрать в панели Редактора реестра ветвь или раздел (подраздел), который необходимо экспортировать. Если нужно записать в файл реестр целиком, нужно выделить щелчком мыши корневой объект дерева реестра — Мой компьютер;
3) в меню Файл выбрать команду Экспорт и в открывшемся диалоговом окне сохранения файла указать место его сохранения и имя (рис. 6.7);
4) щелкнуть по кнопке Сохранить.

Альтернативой предложенному способу экспорта реестра может стать использование консольной команды REG EXPORT. Синтаксис команды приведен на рис. 6.8

6.3.2. Импорт файла реестра или его компонентов

Обладая полномочиями администратора, можно импортировать данные реестра или его компоненты из ранее сохраненного REG-файла. Для этого достаточно дважды щелкнуть по значку соответствующего REG-файла или использовать консольную команду REG IMPORT. Можно воспользоваться и Редактором реестра, для этого нужно выполнить следующие действия:

1) запустить программу Редактор реестра;
2) в меню Файл выбрать команду Импорт и в открывшемся диалоговом окне Импорт файла реестра указать место его расположения и имя (рис. 6.9);
3) щелкнуть мышью на кнопке Открыть.

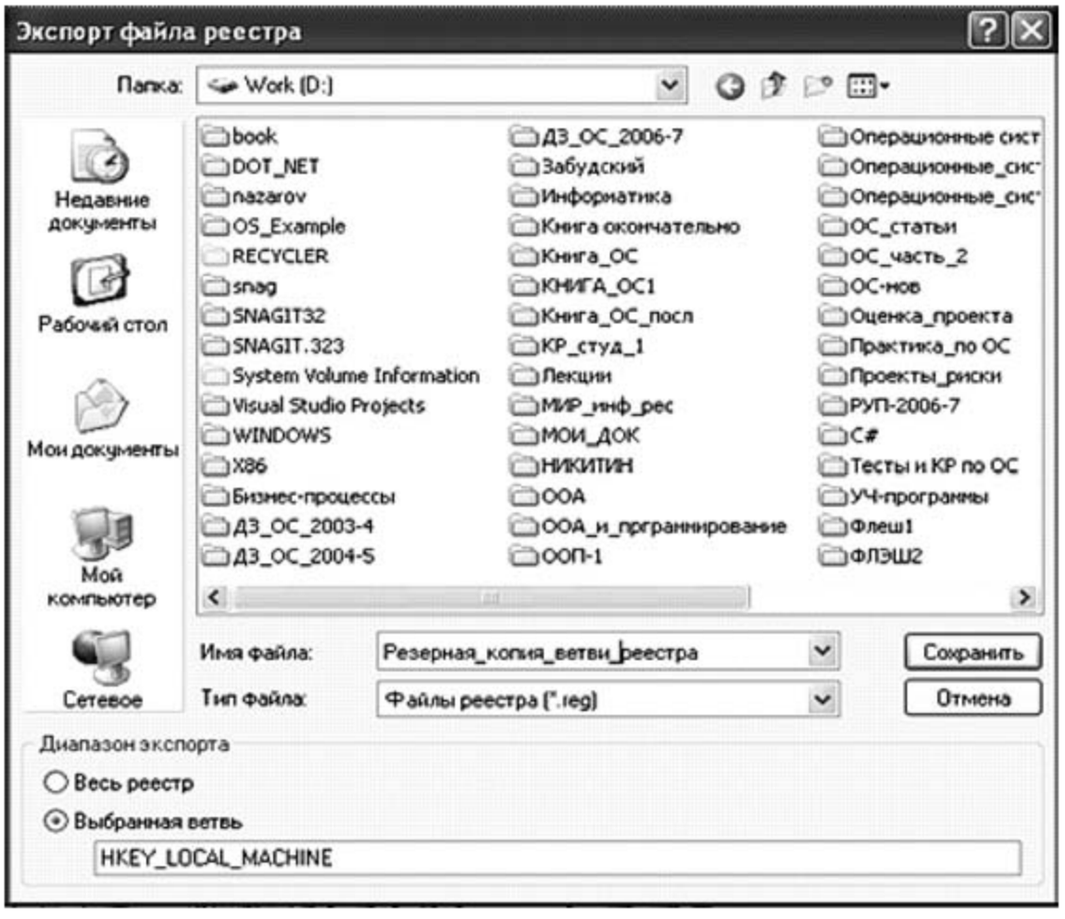


Рис. 6.7

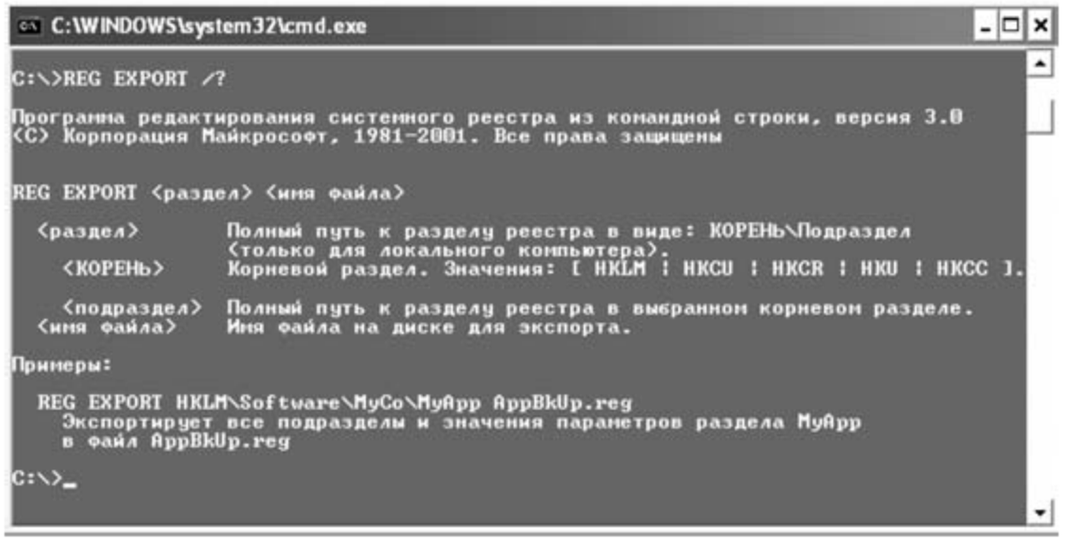


Рис. 6.8

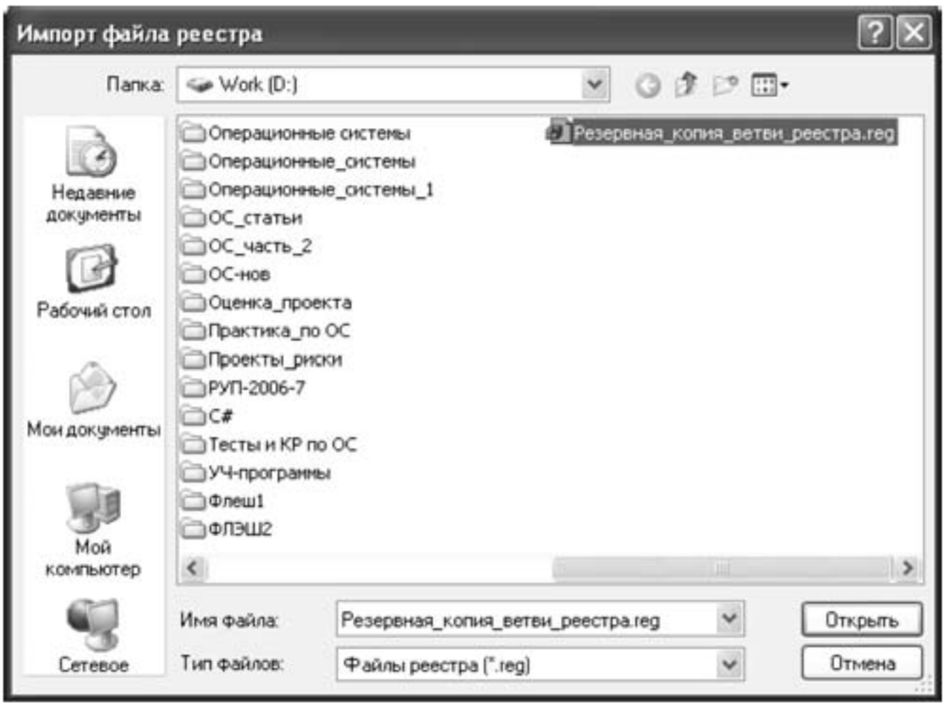


Рис. 6.9

6.3.3. Альтернативные методы резервного копирования реестра Windows XP

В подразделе 5.7 рассмотрена программа Восстановление системы и процедура создания точек восстановления системы. При создании таких точек программа автоматически размещает в резервной копии ключи реестра, отвечающие за пользовательские настройки системы, а также список установленного на компьютере программного обеспечения и оборудования. В процессе аварийного восстановления системы эти данные загружаются в реестр из резервной копии.

Для создания резервной копии реестра на каком-либо внешнем носителе информации (стримере, устройстве для записи компакт-дисков, DVD и др.) можно воспользоваться программой архивации (см. подраздел 5.8). Имеется также возможность копирования и восстановления реестра с помощью Консоли восстановления. Для создания резервной копии системного реестра нужно выполнить в ней следующую последовательность команд:

```
md tmp
copy c:\windows\system32\config\system c:\windows\tmp\system.bak
copy c:\windows\system32\config\software c:\windows\tmp\software.bak
copy c:\windows\system32\config\sam c:\windows\tmp\sam.bak
copy c:\windows\system32\config\security c:\windows\tmp\security.bak
copy c:\windows\system32\config\default c:\windows\tmp\default.bak
```

Не рекомендуется создавать файлы резервной копии реестра с помощью Консоли восстановления за пределами папки установки Windows, поскольку впоследствии доступ к ним может быть затруднен. Восстановить ранее сохраненный таким образом реестр можно обратным перемещением файлов в папку C:\windows\system32\config.

6.4. Очистка реестра

Записи приложений содержатся в разделах HKEY_CURRET_USER\Software (персональные параметры пользователя) и HKEY_LOCAL_MACHINE\SOFTWARE (параметры, общие для всех пользователей компьютера). После удаления многие приложения оставляют в реестре свои параметры — как правило, системные или пользовательские. Это может привести к следующим проблемам [5]:

1) неэффективному использованию системных ресурсов. Windows «верит» всему, что записано в реестре. Если реестр содержит запись о том, что для обработки файла или решения задачи в системе имеется определенное приложение, то Windows будет искать его, потребляя системные ресурсы, даже если приложения на самом деле нет;

2) нарушению надежности системы. Проблемы с устойчивостью системы начинают проявляться, когда компьютер пытается использовать несуществующий ресурс. Вообще система успешно восстанавливается после одиночных и даже двойных ошибок. Но по мере накопления в реестре некорректных записей, правильно реагировать на ошибки становится труднее, и начинают наблюдаться проблемы (компьютер ведет себя странным образом, вплоть до краха системы);

3) нарушению надежности приложений. Приложения получают данные из целого ряда источников — от других приложений, драйверов устройств, ОС. Что произойдет, если одно приложение посредством Windows посылает запрос другому приложению, но оказывается, что оно удалено из системы? Если запрашивающее приложение не обладает хорошим механизмом обработки ошибок, оно потерпит крах по причине некорректной записи в реестре;

4) нарушению безопасности. Безопасность компьютера может пострадать из-за «мусора» в реестре. Вирусам проще скрыть себя среди устаревших записей, а антивирусным приложениям труднее их обнаружить. Более того, устаревшие записи способствуют проникновению в систему вредоносных программ. Например, загрузка из-за некор-

ректной записи в реестре в память ненужной библиотеки DLL делает возможными атаки на систему через эту библиотеку.

Специальные средства очистки реестра помогают в поиске устаревших данных установки приложений, однако не гарантируют обнаружения всех этих данных. Большинство средств очистки ищут некорректные параметры, но с этими параметрами часто связано множество записей. Для самостоятельного поиска таких записей можно воспользоваться рядом критериев:

- именем разработчика;
- именем приложения;
- именем исполняемого файла приложения;
- именами DDL;
- GUID компонентов (Globally Unique Identifier — глобально уникальный идентификатор);
- URL приложения;
- уникальными данными настройки приложения.

Поскольку реестр столь важен для работы Windows, следует не допускать ошибок при его редактировании. Поэтому прежде чем начать работу, необходимо сохранить все ветви реестра и заархивировать их.

Хорошие утилиты для очистки реестра способны значительно упростить обнаружение некорректных записей и снизить вероятность удаления нужных. Утилита RegClean (http://www.majorgeeks.com/download458.html) является продуктом компании Microsoft. При первом запуске она долго сканирует реестр и выдает сообщение о готовности к устранению ошибок (рис. 6.10). Недостатком этой утилиты является отсутствие списка сделанных исправлений, достоинством — создание файла UNDO.REG, с помощью которого можно вернуть реестр к предыдущему состоянию. Для этого достаточно дважды щелкнуть на этом файле.

Утилита Registry Mechanic (http://www.winguides.com/regmech/) позволяет создать резервную копию реестра до начала сканирования. После запуска утилиты можно увидеть большой список сканируемых объектов (рис. 6.11). Особенность утилиты — возможность задания такого режима работы, при котором реестр сканируется при каждом запуске Windows. Сканирование, выполняемое утилитой Registry Mechanic, отличается чрезвычайной полнотой, и, по мнению А.П. Шалина, [18], вряд ли есть лучшая утилита. Для восстановления реестра достаточно щелкнуть по кнопке Backup.

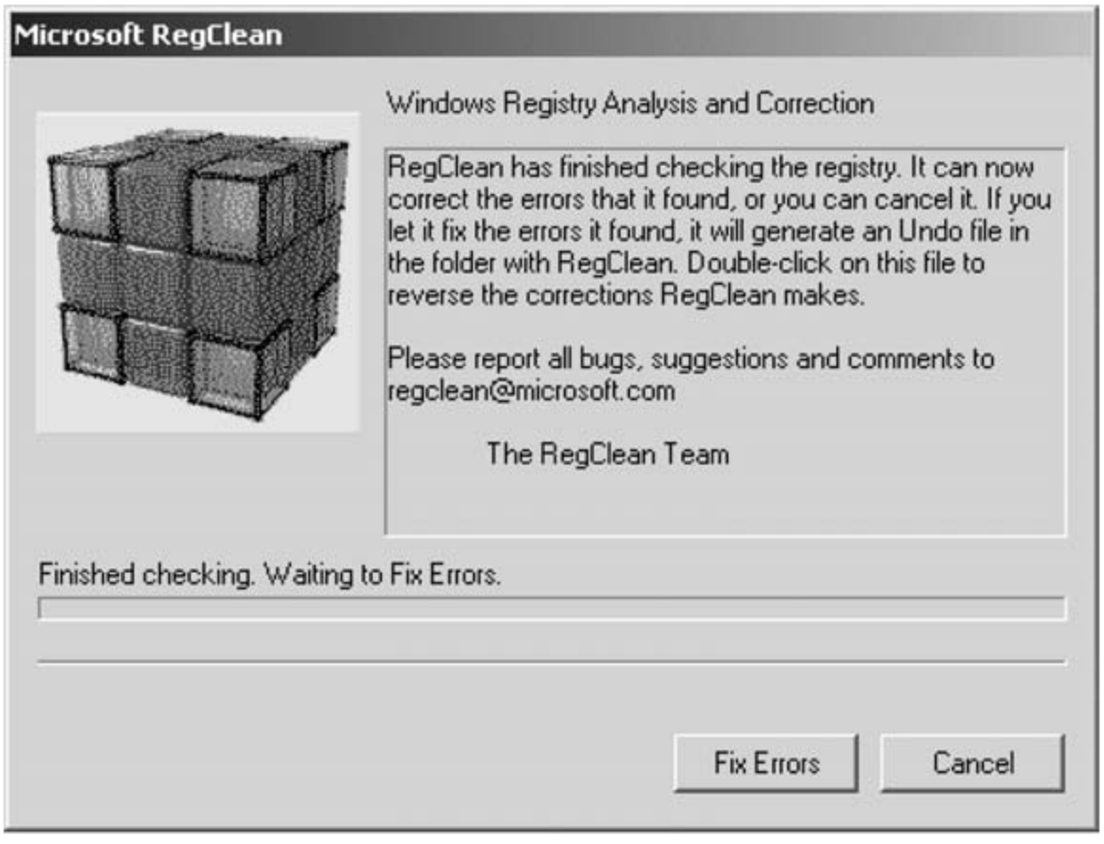


Рис. 6.10

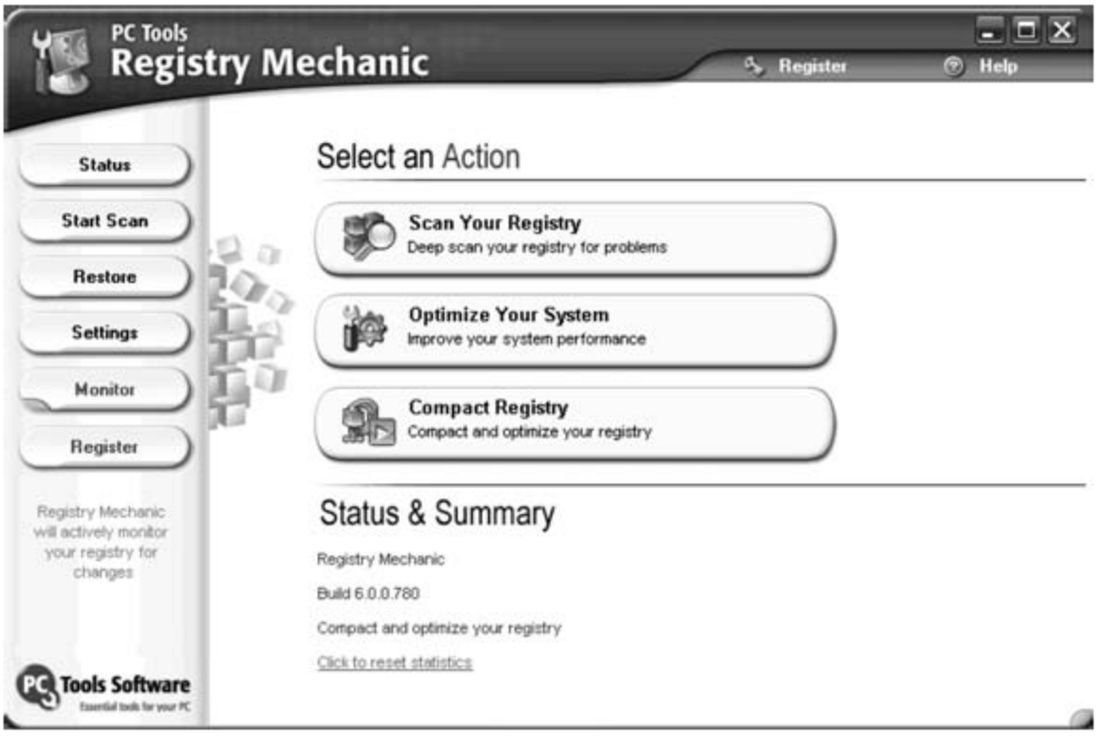


Рис. 6.11

Утилита TweakNow RegCleaner (http://www.tweaknow.com/RegCleaner.html) обнаруживает в реестре наиболее распространенные ошибки

(рис. 6.12). Ее возможностей вполне достаточно для домашних и офисных компьютеров. Хотя эта утилита находит в реестре меньше ошибок по сравнению с Registry Mechanic, но ее быстродействие значительно выше. Одна из функций TweakNow RegCleaner — индикация безопасности или небезопасности удаления записей реестра. Другие утилиты подобной функции не предоставляют. Утилита TweakNow RegCleaner автоматически создает резервную копию реестра перед удалением любых некорректных записей. При этом поддерживается лишь одна резервная копия, что при наличии нескольких вариантов очистки вынуждает пользователя тестировать их поочередно.

Рис. 6.12

6.5. Редактирование реестра

6.5.1. Удаление недействительных записей из списка установленных программ

Утилита Установка и удаление программ на панели управления предназначена для установки, удаления или модификации приложений, установленных в Windows. Несмотря на удобство и простоту ее использования, в работе этой программы могут возникать неполадки, вызванные некорректно или не полностью удаленными приложения-

ми. В частности, если приложение удалено некорректно, то ссылка на него по-прежнему будет фигурировать в списке Установленные программы.

При этом любая попытка воспользоваться мастером установки удаления приложений будет приводить к появлению сообщений об ошибке, информирующих пользователя об отсутствии файлов, требующихся для корректного удаления приложения и невозможности завершить процедуру удаления. Несуществующие приложения по-прежнему останутся в списке.

Чтобы устранить эту проблему и удалить несуществующие приложения из списка установленных программ, нужно выполнить следующие действия:

1) запустить Редактор реестра и раскрыть ключ HKEY_LOCAL_MA-CHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall;
2) в составе этого ключа найти вложенный ключ, соответствующий приложению, которое требуется удалить из списка установленных программ. Если наименование программы не следует с очевидностью из имени вложенного ключа, нужно просмотреть содержимое всех вложенных ключей списка. В составе каждого из вложенных ключей имеется параметр DisplayName, значение которого представляет собой строки, отображаемые мастером установки и удаления программ;
3) обнаружив ключ, в составе которого имеется параметр DisplayName, задающий имя приложения, которое должно быть удалено из списка установленных программ, нужно удалить этот ключ вместе со всем его содержимым;
4) закрыть редактор реестра и убедиться в том, что мастер установки и удаления программ больше не отображает несуществующего приложения.

Следует отметить, что существует вероятность того, что некоторые из файлов некорректно удаленного приложения все же останутся в системе. Чтобы полностью удалить такое приложение, необходимо удалить из системы все его файлы и все параметры реестра, относящиеся к этому приложению.

6.5.2. Ускорение работы системы с памятью

Если на компьютере установлена оперативная память достаточно большого объема (не менее 256 Мбайт) и не предполагается запускать

одновременно множество приложений, то можно значительно повысить производительность компьютера, ликвидировав файл подкачки. Однако нужно понимать, что это может привести к снижению надежности функционирования компьютера. Чтобы решить, стоит ли отказываться от файла подкачки, нужно определить, сколько памяти потребляет система в каждый момент времени. Если свободная память составляет хотя бы 40% ее общего объема, то отключение подкачки возможно [5, 6]. Если же после этого система будет давать сообщения о недостатке оперативной памяти, следует снова включить подкачку или установить оперативную память большего объема.

Для отключения файла подкачки нужно выполнить следующие действия:

1) запустить Редактор реестра и найти ключ HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Control\Session Manager\ Memory Management (рис. 6.13);

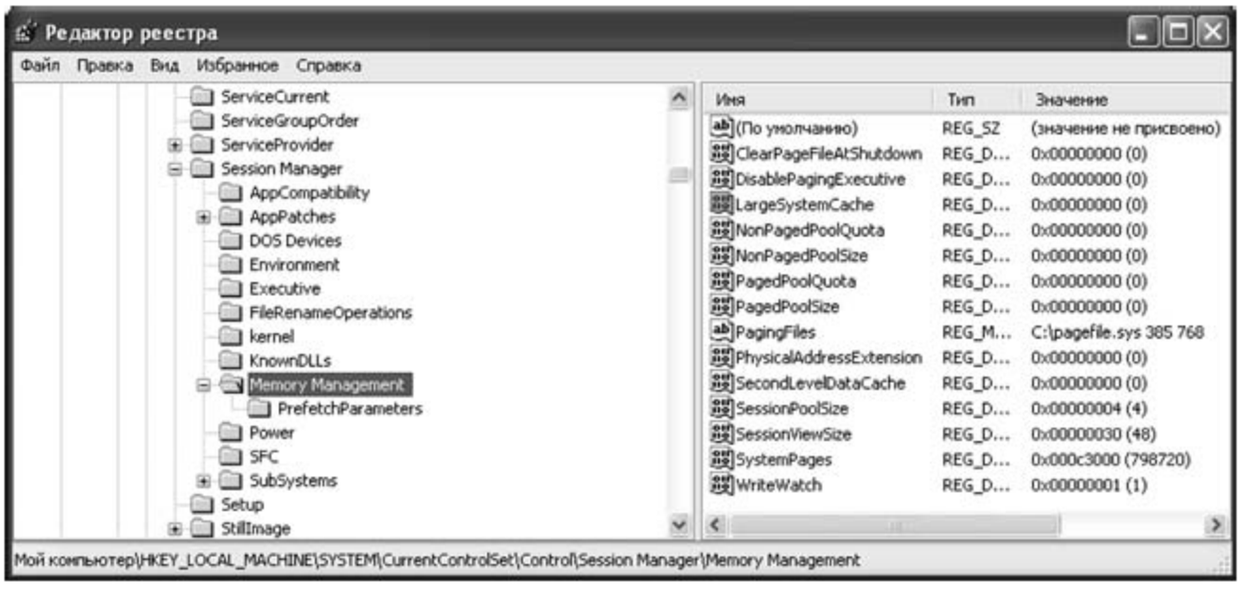


Рис. 6.13

2) установить значение параметра DisblePagingExecutive равным 1. Это будет указанием системе на то, что данные следует (в том числе ядро системы и драйверы) хранить в оперативной памяти, а не сбрасывать их на жесткий диск (рис. 6.14);
3) установить значение параметра LargeSystemCache равным 1. В этом случае ОС будет выделен участок оперативной памяти минимальным объемом в 4 Мбайт (по умолчанию выделяется 8 Мбайт) для кэширования ядра и окружения системы, что увеличит скорость доступа к этим компонентам. Это укажет Windows на то, что ее ядро следует хранить в оперативной памяти;

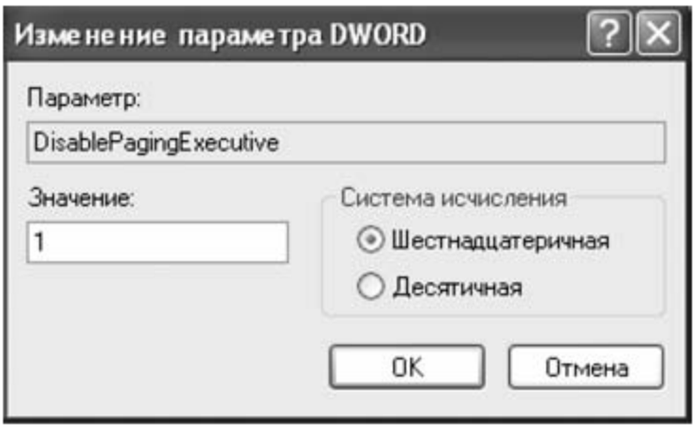


Рис. 6.14

4) создать новое значение типа DWORD с именем IOPageLockLimit. Этот параметр управляет объемом данных при их постраничной передаче и таким образом влияет на скорость выполнения операций ввода-вывода. Установить его равным 10 000, если объем оперативной памяти не менее 256 Мбайт. Если объем памяти не менее 512 Мбайт, значение параметра может быть равным 40 000.

Далее следует опробовать новые параметры, чтобы удостовериться в отсутствии ошибок, связанных с недостатком оперативной памяти. После этого нужно снова открыть окно редактора реестра, найти параметр PagingFiles, удалить из него текст и перегрузить компьютер. Это приведет к установке нулевого размера файла подкачки. После перезагрузки нужно удалить с жесткого диска файл PageFile.sys.

Одним из важных факторов, определяющим работу Windows, является объем перемещаемой и неперемещаемой памяти. Для задания объема неперемещаемой памяти нужно установить значение параметра NonPagedPoolSize (в байтах). После перезагрузки компьютера можно проверить новое значение величины невыгружаемой памяти на вкладке Быстродействие Диспетчера задач (рис. 6.15).

6.5.3. Повышение производительности системы

В составе ОС Windows XP имеется специальная утилита отладки Doctor Watson, предназначенная для отслеживания сбоев в приложениях и выдачи соответствующих системных предупреждений. Утилита загружается автоматически с загрузкой системы и чаще всего себя никак не проявляет. Поэтому можно отказаться от услуг утилиты и вы-

свободить память, занимаемую этой утилитой. Для этого необходимо выполнить следующие действия:

1) запустить Редактор реестра;
2) выбрать ветвь HKEY_LOCAL_MACHINE;
3) перейти к подразделу SOFTWARE\Microsoft\WindowsNT\Current-Version\AeDebug (рис. 6.16);
4) установить значение 0 для параметра Auto.

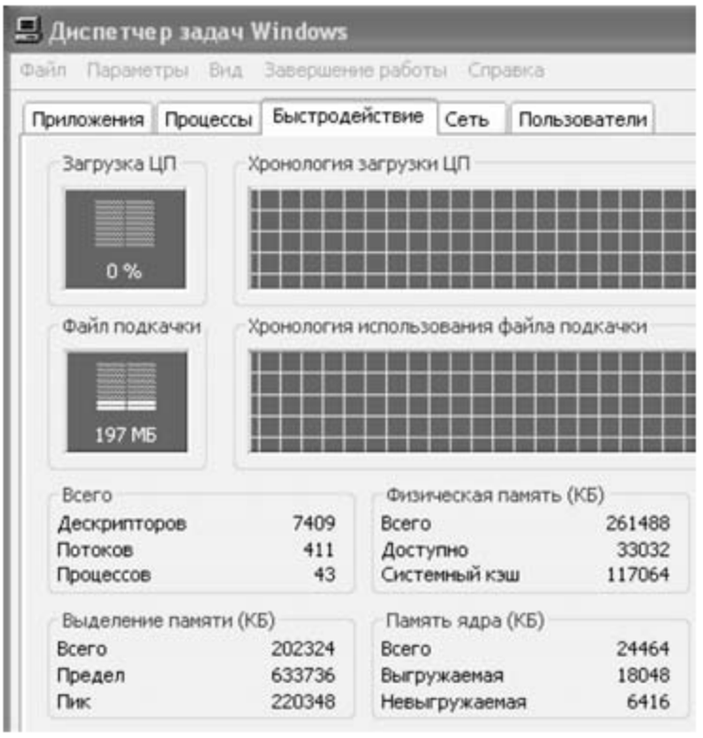


Рис. 6.15

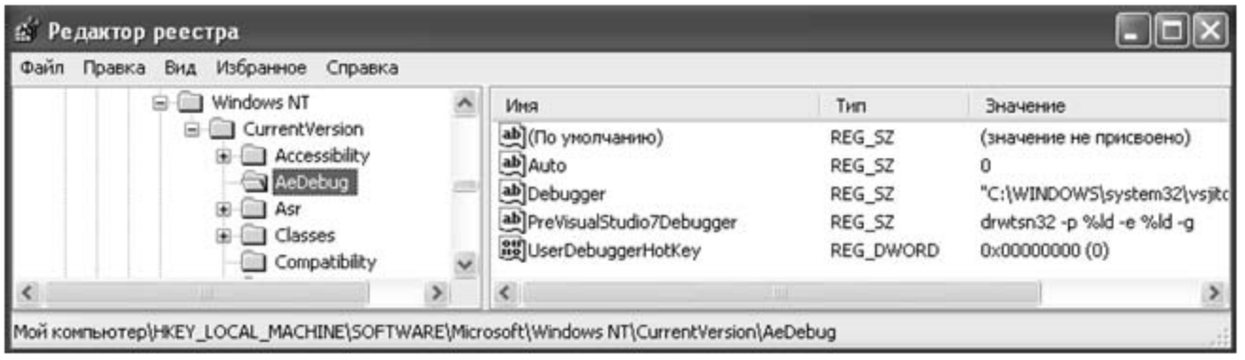


Рис. 6.16

Хранящиеся в разделе диска NTFS папки, которые содержат значительное количество файлов, открываются в Проводнике очень долго, поскольку при каждом обращении к папкам Windows автоматически обновляет системную метку последнего доступа к файлам, что при большом числе файловых объектов занимает много времени. Чтобы отключить эту функцию, нужно в ветви HKEY_LOCAL_MA-

CHINE найти подраздел SYSTEM\CurrentControlSet\Control\File-System. Далее следует создать параметр типа DWORD и присвоить ему значение 1 (рис. 6.17).

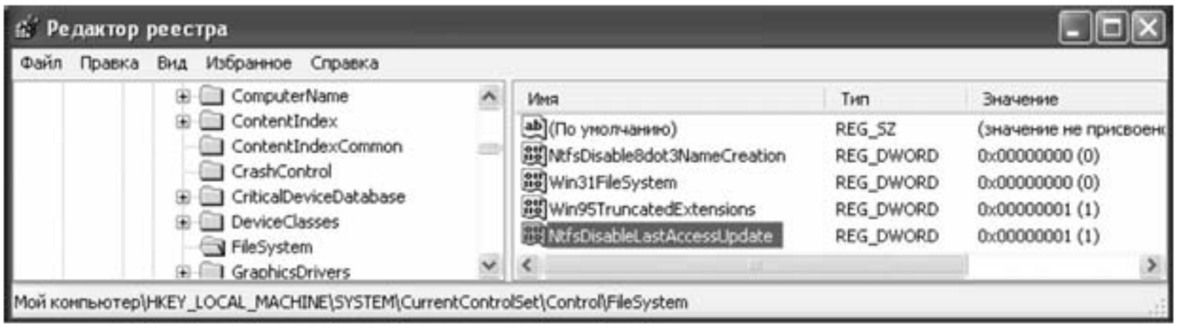


Рис. 6.17

Значительного ускорения работы пользовательской оболочки Windows XP можно добиться, отключив стандартными средствами все визуальные эффекты (тень под курсором мыши, фоновые рисунки, темы, анимацию и т.д.). Для получения доступа к опциям визуальных настроек следует выполнить команды Панель управления — Система — Дополнительно — Параметры быстродействия. Затем на вкладке Визуальные эффекты нужно убрать или поставить флажки напротив нужных опций. Также в этом отношении эффективно уменьшение времени задержки при разворачивании меню и отключение анимации при сворачивании-разворачивании окон.

Используя системный реестр, это можно сделать следующим образом. Выбрать ветвь HKEY_CURRENT_USER и перейти к подразделу ControlPanel\Desktop. Найти параметр MenuShowDelay (по умолчанию он равен 400) и установить его значение равным 0 (рис. 6.18). Однако в этом случае список будет появляться мгновенно даже при быстром перемещении курсора над кнопкой. Это может быть неудобно, поэтому нужно определить оптимальное значение экспериментальным путем. Далее перейти в подраздел Windows Metrics и установить значение параметра MinAnimate равным 0 (рис. 6.19) и перезагрузить компьютер.

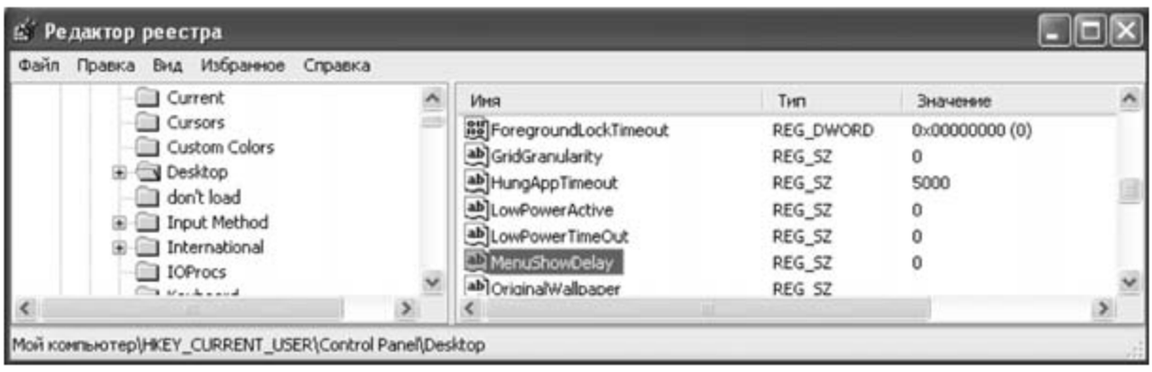


Рис. 6.18

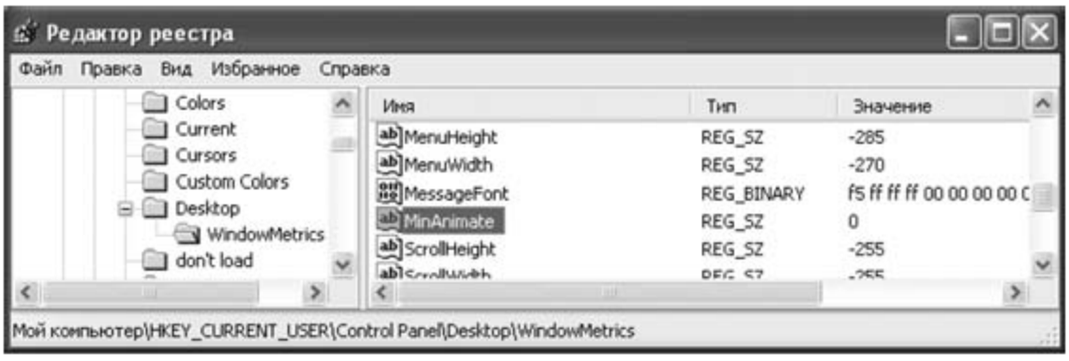


Рис. 6.19

Задания для самостоятельной работы

1. Запустить Редактор реестра. Выполнить резервное копирование одной из ветвей реестра. Используя консольную команду REG EXPORT, выполнить копирование другой ветви реестра.
2. Получив полномочия администратора, импортировать данные реестра или его компоненты из ранее сохраненного REG-файла.
3. Провести очистку реестра, используя утилиты, загруженные с указанных в подразделе 6.4 сайтов.
4. Удалить недействительные записи из списка установленных программ, как это рассмотрено в подразделе 6.5.1.
5. Удалить файл подкачки по методике, рассмотренной в подразделе 6.5.2. Повысилась ли производительность системы? Попробуйте запустить несколько приложений. Нет ли отказов системы по причине недостаточного объема памяти? Объяснить полученный результат. Восстановить файл подкачки.
6. Попробуйте увеличить объем неперемещаемой памяти. Изменилась ли производительность системы? Объяснить полученный результат.
7. Отключить все визуальные эффекты (тень под курсором мыши, фоновые рисунки, темы, анимацию и т.д.). Изменилась ли производительность системы? Объяснить полученный результат.

6.6. Системные службы

Архитектура Windows XP предусматривает наличие в составе ОС определенного количества сервисов (системных служб) — специаль-

ных программ, которые запускаются при загрузке системы и обеспечивают выполнение каких-либо специализированных системных задач. Следует отметить, что не все сервисы (их более 80), настраиваемые по умолчанию в процессе установки ОС, нужны пользователю. Управление системными службами возможно с помощью реестра, однако в составе Windows XP есть более удобный инструмент — программа Настройка системы. Для ее запуска нужно последовательно выполнить команды Пуск — Выполнить, а затем в открывшемся окне Запуск программы набрать msconfig и щелкнуть по кнопке OK (рис. 6.20). Все изменения, внесенные в конфигурацию ОС с помощью данного приложения, будут автоматически внесены в реестр и другие системные файлы.

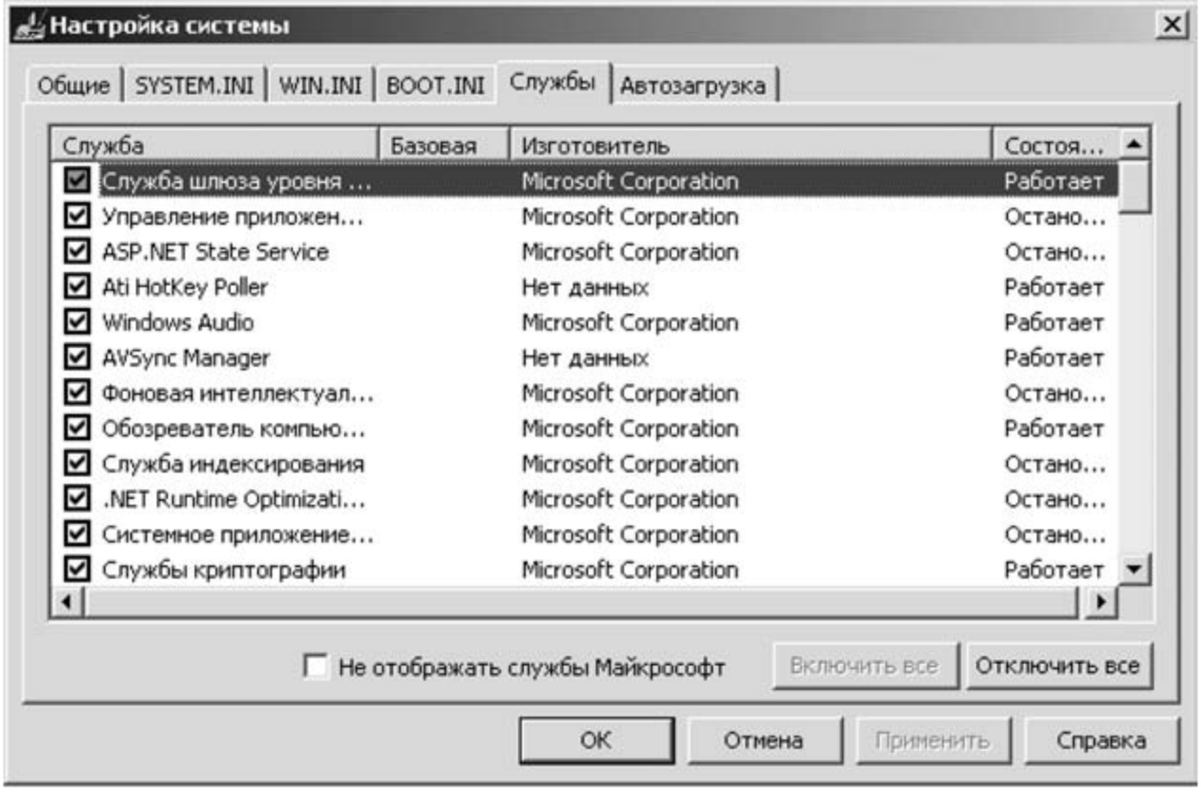


Рис. 6.20

Службы Windows XP являются системными процессами, которые осуществляют поддержку оборудования и программного обеспечения компьютера на низком, близком к аппаратному, уровне. Пользователь может определить список запускаемых по умолчанию сервисов в зависимости от назначения и способов использования компьютера. Для этого нужно выполнить следующие действия:

1) запустить программу Настройка системы и перейти на вкладку Службы;

2) сбросить флажки напротив тех служб, которые не будут использоваться;
3) закончив редактирование списка, щелкнуть по кнопке OK и перезагрузить компьютер.

Следует иметь в виду, что снятие флажка у той или иной службы приводит только к отключению, но не удалению этой службы. Более совершенные методы работы со службами обеспечиваются консолью Службы, которая находится в папке Администрирование панели управления. Типичный вид консоли приведен на рис. 6.21.

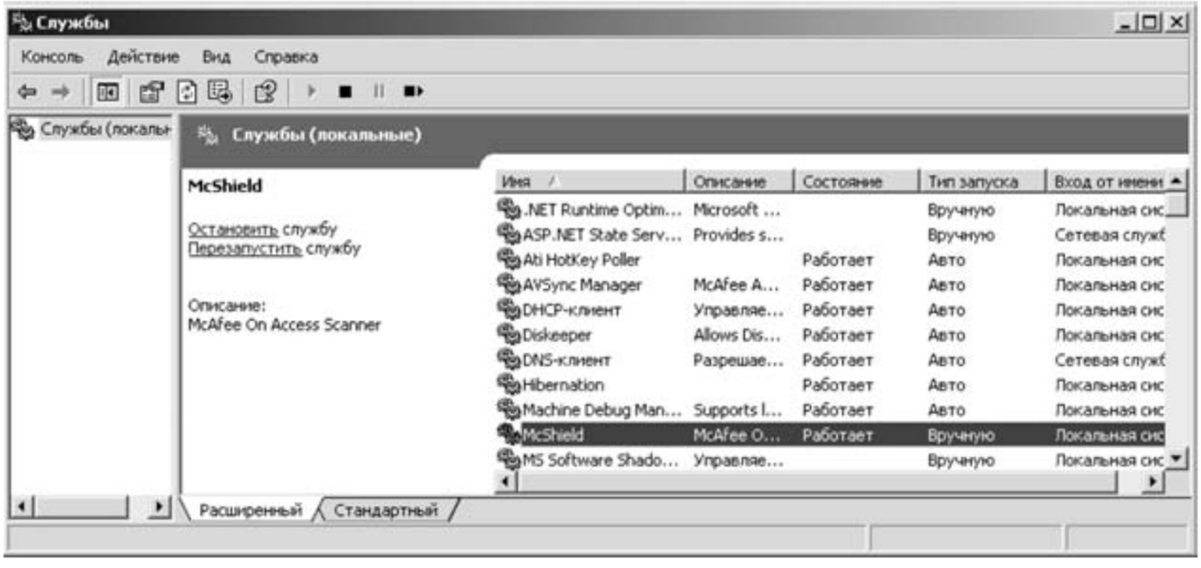


Рис. 6.21

В консоли отображается состояние и тип запуска службы, определяющий, запускается ли она автоматически или вручную. Остановленная служба не потребляет ресурсов компьютера. Службы, помеченные как Авто, запущены постоянно, а службы с типом запуска Вручную запускаются по требованию приложения. Тип запуска Отключено означает, что службы не запускаются ни при каких условиях. Хотя службы при ожидании не обязательно расходуют ресурсы процессора и их нелегко удалить с жесткого диска, оперативную память они все же потребляют. Эту память можно сэкономить, ликвидировав ненужные службы.

Для получения максимальной производительности можно остановить некоторые службы или установить тип запуска Вручную. Не рекомендуется устанавливать тип запуска Отключено, если нет уверенности, что данная служба не используется. Прежде чем начинать экспериментировать, нужно выяснить, какие службы запущены на компьютере. Это можно сделать, выполнив команду Net Start в Командной строке, в результате чего отобразится полный список всех запущенных служб (рис. 6.22).

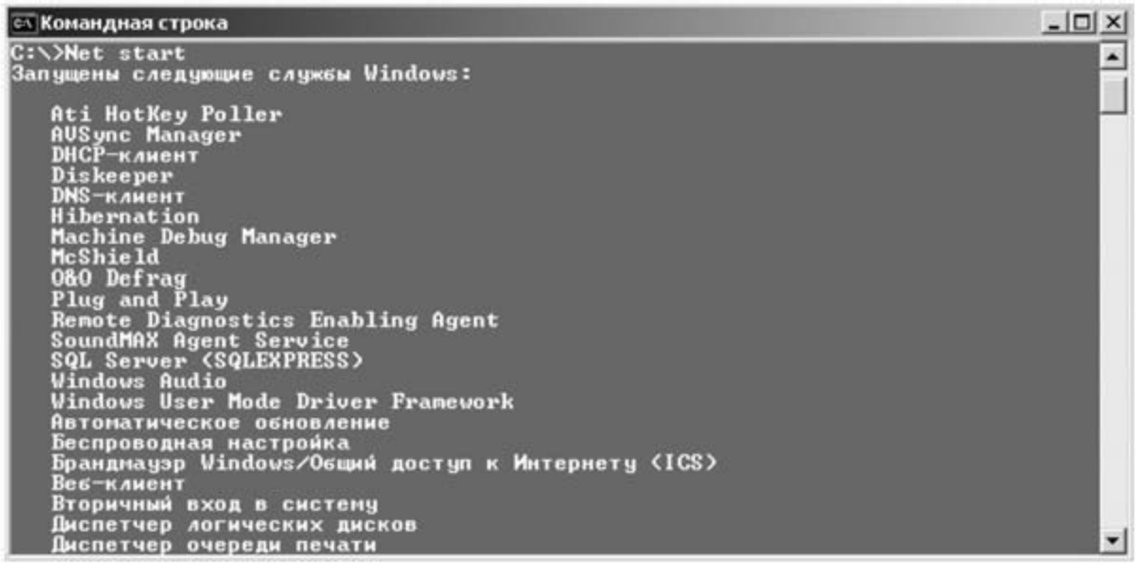


Рис. 6.22

Возможность отключения конкретной службы зависит от конфигурации ПК и от сценария его использования. Например, если в ПК не используется беспроводный адаптер, то такая служба, как Беспроводная настройка, абсолютно бесполезна. Большое количество служб можно отключить, если компьютер не входит с состав локальной сети или не имеет выхода в Интернет. Например, для автономного компьютера можно остановить следующие службы:

- обозреватель компьютеров;
- Net Meeting Remote Desktop Sharing;
- сетевой вход в систему;
- службу сетевого расположения NLA.

Если пользователь не работает в Интернете и на компьютере нет модема, можно отключить:

- сетевые подключения;
- DHCP-клиент;
- DNS-клиент;
- сервер;
- модуль поддержки NetBIOS через TCP/IP;
- службу сетевого расположения NLA;
- службы IPSEC;
- диспетчер автоподключений удаленного доступа;
- диспетчер подключений удаленного доступа;
- брандмауэр Windows/общий доступ к Интернету (ICS);
- телефонию.

Для работы с системными службами пользователь обычно использует консоль Службы, которая позволяет выполнять следующие действия:

просмотреть описание службы, которое появляется после щелчка мышью на требуемой службе; запустить, приостановить, прекратить, продолжить или перезапустить выделенную службу, выбрав соответствующую команду в меню Действие. Многие службы связаны друг с другом, поэтому остановка или блокировка одной из них может привести к невозможности запуска или выполнения некоторых других.

Для отслеживания внутренних связей между различными службами можно поступить следующим образом:

1) открыть консоль службы;
2) выделить требуемую службу и щелкнуть на ней правой кнопкой мыши;
3) в контекстном меню выбрать строку Свойства;
4) в открывшемся окне (рис. 6.23) перейти на вкладку Зависимости, демонстрирующую список сервисов, от которых зависит данная служба, и список служб, которые зависят от данной службы.

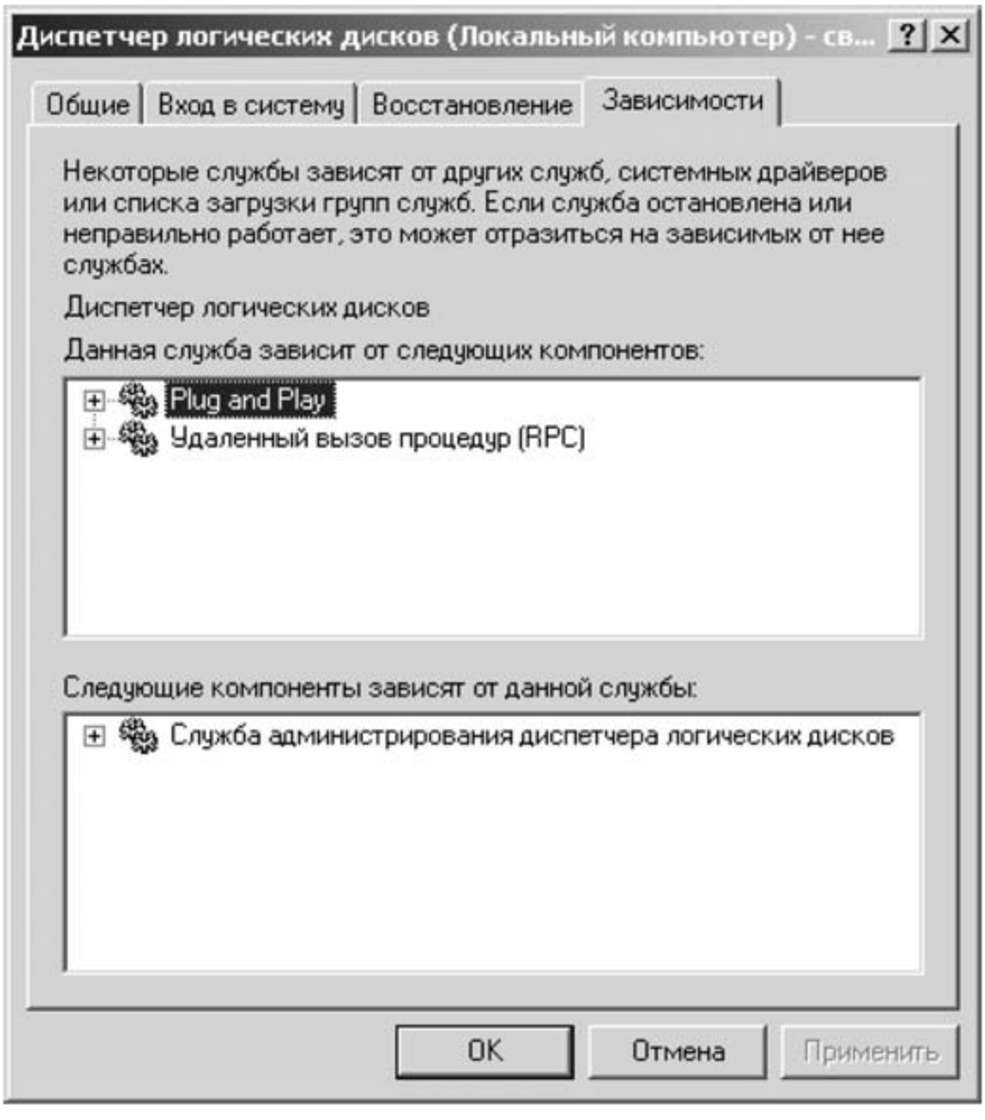


Рис. 6.23

Системная служба Windows XP, как и любая другая программа, может вызывать сбои в процессе своей работы. Чтобы настроить правила обработки ошибок при возникновении подобных сбоев, нужно выполнить следующие действия:

1) открыть консоль Службы;
2) выделить требуемую службу и щелкнуть на ней правой кнопкой мыши;
3) в контекстном меню выбрать строку Свойства;
4) в открывшемся окне (рис. 6.24) перейти на вкладку Восстановление;

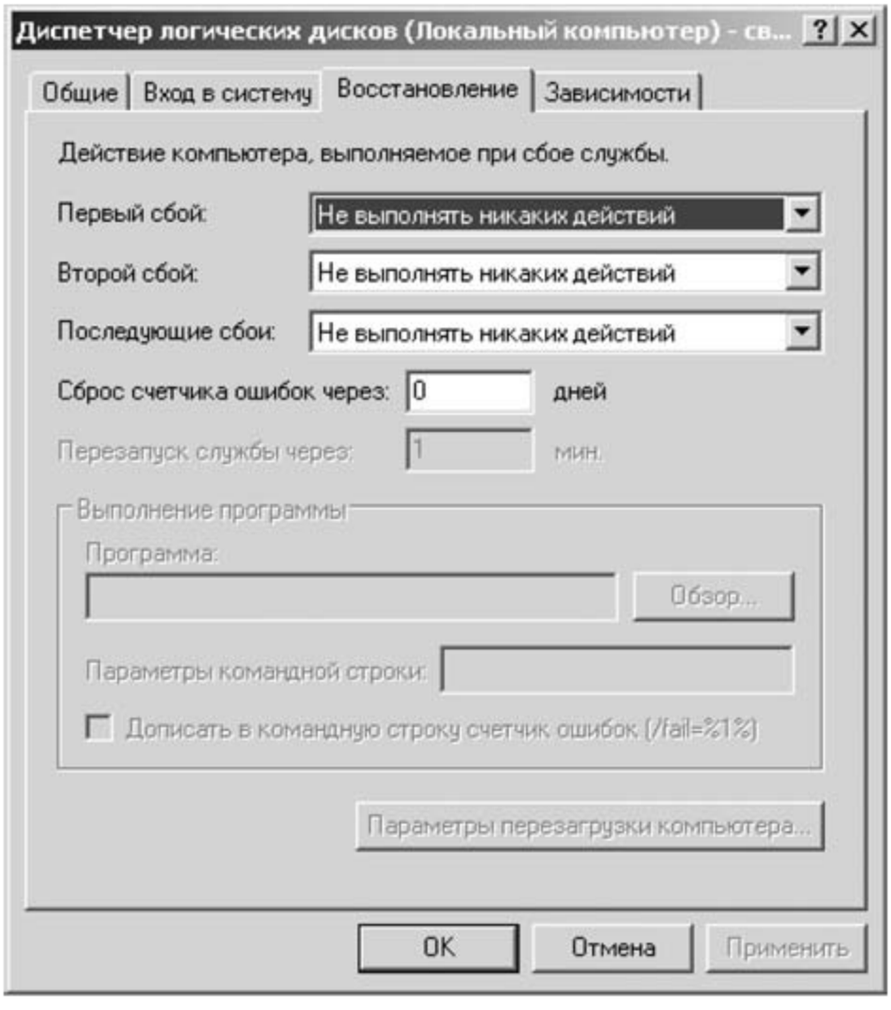


Рис. 6.24

5) в раскрывающихся списках Первый сбой, Второй сбой и Последующие сбои выбрать тип реакции обработчика ошибок из предложенных вариантов: Не выполнять никаких действий, Перезапуск службы, Запуск программы или Перезагрузка компьютера;
6) при выборе пункта Запуск программы в поле Программа указать имя программы и путь, в поле Параметры командной строки ввести ключи, необходимые для запуска программы и, если необхо-

димо, установить флажок Дописать в командную строку счетчик ошибок;

7) если в одном из списков выбран пункт Перезагрузка компьютера, можно настроить режим перезагрузки щелчком на кнопке Параметры перезагрузки компьютера, указав время, через которое будет выполнена перезагрузка в случае возникновения сбоя, и создав административное сообщение, которое будет автоматически разослано всем компьютерам локальной сети перед выполнением перезагрузки (рис. 6.25);

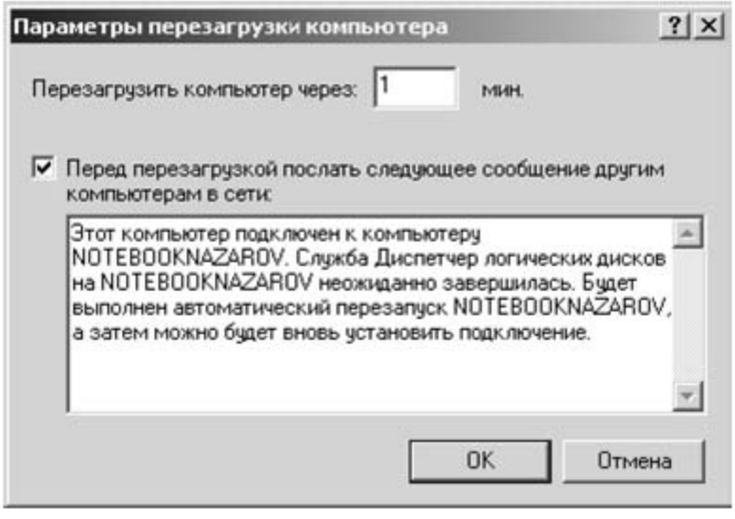


Рис. 6.25

8) если выбран пункт Перезапуск службы, нужно указать промежуток времени, по истечении которого служба будет автоматически перезапущена;
9) если это необходимо, в поле Сброс счетчика ошибок следует указать количество дней, по истечении которых счетчик будет обнулен;
10) щелкнуть по кнопке OK, чтобы внесенные изменения вступили в силу.

По умолчанию большинство сервисов запускаются автоматически при открытии сеанса Windows с полномочиями системы или учетной записи пользователя, открывшего сеанс. Можно перенастроить этот режим таким образом, что служба будет запускаться при загрузке системы от имени учетной записи другого пользователя. Для этого нужно:

1) выделить требуемую службу, щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню строку Свойства;
2) в открывшемся диалоговом окне перейти на вкладку Вход в систему (рис. 6.26);

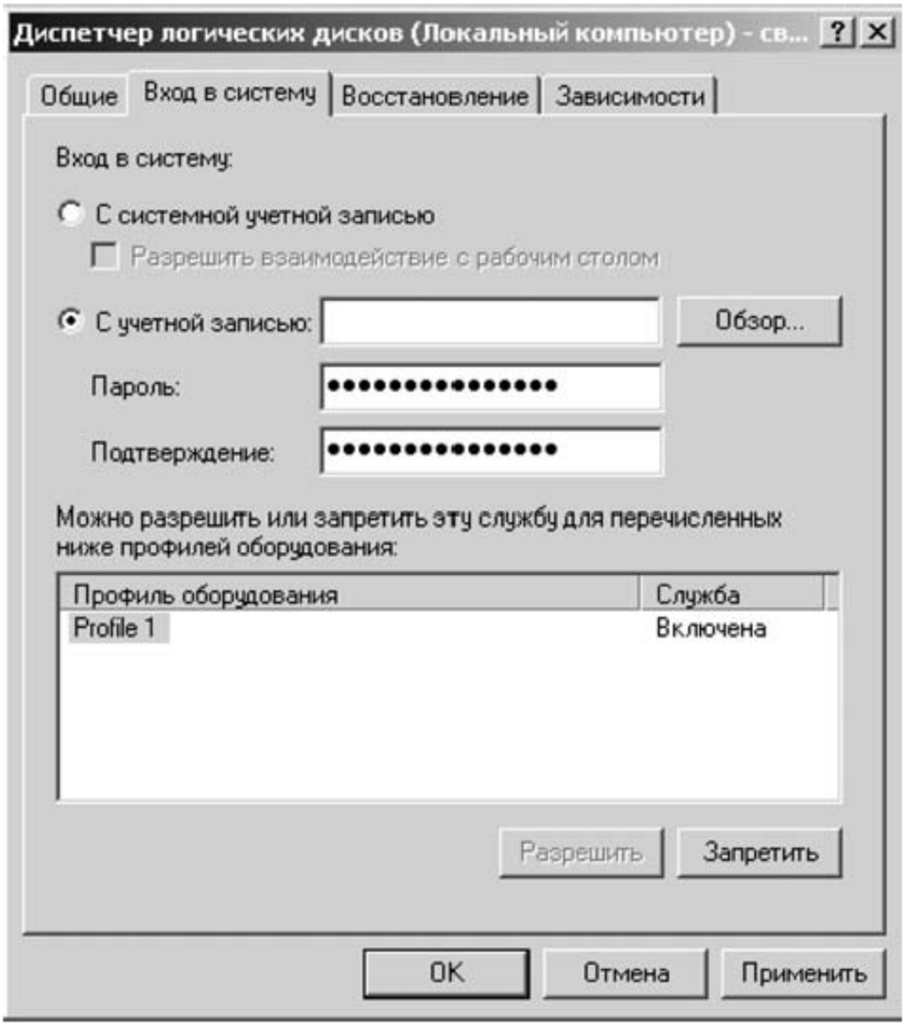


Рис. 6.26

3) если требуется, чтобы при загрузке Windows служба запускалась с полномочиями операционной системы, установить переключатель Вход в систему в положение С системной учетной записью. Чтобы запускать сервис от имени другого пользователя, установить переключатель в позицию С учетной записью и ввести название учетной записи, а в полях Пароль и Подтверждение следует ввести пароль, присвоенный данному профилю;
4) чтобы разрешить или запретить автоматический запуск службы для различных учетных записей, выберите соответствующий профиль в нижней части окна и щелкните по кнопке Разрешить или Запретить;
5) если установить флажок Разрешить взаимодействие с рабочим столом, служба автоматически подключит интерфейс, с которым сможет взаимодействовать любой пользователь, вошедший в систему после запуска службы;
6) щелкнуть мышью по кнопке OK, чтобы внесенные изменения вступили в силу.

Задания для самостоятельной работы

1. Запустить программу msconfig. Просмотреть системные службы, установленные на компьютере. Освоить работу с консолью Службы.
2. Исходя из специфики работы на вашем компьютере, определить, есть ли необходимость во всех запущенных службах. Удалить ненужные службы.

ГЛАВА 7

ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ СИСТЕМЫ

7.1. Защита от вторжений. Брандмауэры

Учитывая все возрастающее количество программ, разрабатываемых для атак на ОС, важнейшей проблемой стало обеспечение безопасности компьютера. Прошло время, когда источником вредоносного программного обеспечения были документы на дискетах и приложения к электронным письмам. Сейчас вирусы и «черви» могут проникнуть в компьютер вообще без каких-либо действий пользователя. Инфицированная машина сама может превратиться в источник распространения вирусов.

Поскольку взаимодействие компьютера с внешним миром осуществляется через порты, а их достаточно много (65 536 в IBM-совместимом компьютере), то целесообразна идея закрытия большинства из них, кроме немногих (одного-двух), абсолютно необходимых. Определить, насколько компьютер открыт для внешнего мира, можно с помощью специальных тестов, позволяющих оценить уровень уязвимости компьютера. Ниже приведены сайты, которые помогут в решении этой задачи:

- http://security.symantec.com;
- http://scan.sygate.com;
- www.grc.com;
- www.dslreports.com/scan.

Для проверки компьютера нужно зайти на один из этих сайтов (например, на сайт security.Symantec.com, рис. 7.1) и выполнить представленные там инструкции.

Возможные варианты сканирования компьютера после щелчка по кнопке GO показаны на рис. 7.2. После выбора одного из вариантов начнется сканирование с обещанием детального анализа полученных результатов, когда сканирование будет закончено. Обобщенные результаты сканирования показаны на рис. 7.3 (в данном случае сканиро-

вание не было проведено до конца). Далее можно получить детальную информацию по каждому представленному результату, для чего нужно щелкнуть по надписи Show Details. Например, на рис. 7.4 представлена развернутая информация по проверке защищенности компьютера от хакеров. Результаты анализа говорят о том, что неавторизованные пользователи могут соединиться с компьютером и информация, хранящаяся на нем, не защищена от хакеров.

Рис. 7.1

Идеальных ОС не существует, в их числе и Windows XP. Поэтому Microsoft выпускает ежемесячные обновления безопасности, а также срочные внеплановые обновления. Веб-сайт Windows Update позволяет познакомиться со всеми обновлениями, критически важными (critical update), и обновлениями механизмов ОС (features updates). Критически важные обновления призваны решать проблемы, связанные с безопасностью, например проблему защиты от широко распространенного эксплойта для Windows XP, известного под именем «червя» W32.Blaster.Worm, который распространялся через уязвимость в системе RPC (Remote Procedure Call — вызов удаленных процедур).

В Windows XP имеется полезная служба автоматического обновления; установив расписание для ежедневной автоматической проверки и установки новых обновлений, можно отказаться от интерактивных

посещений веб-сайта Windows Update. Для настройки параметров автоматического обновления нужно щелкнуть правой кнопкой мыши по значку Мой компьютер и выбрать в контекстном меню строку Свойства, а затем перейти на вкладку Автоматическое обновление (рис. 7.5).

Рис. 7.2

Установив открытые порты компьютера, как это рассмотрено выше, можно их заблокировать, оставив минимальное количество открытых, с помощью специальной программы — брандмауэра. Когда удаленный компьютер попытается через заблокированный порт получить доступ к компьютеру, на котором установлен брандмауэр, он не сможет этого сделать, поскольку посылаемые им данные будут игнорироваться.

При попадании данных в заблокированный порт в зависимости от настройки брандмауэр отвечает, что порт закрыт, или вообще ничего не отвечает, делая компьютер невидимым извне. Компьютер, на котором установлен брандмауэр, работающий в режиме невидимости, для любого удаленного компьютера, пытающегося к нему подключиться, будет выглядеть как выключенный, так как никакого ответа удаленный компьютер не получит.


http://security.symantec.com - Symantec Security Check - Microsoft Internet Explorer
symantec.
Symantec Security Check
> SECURITY INFORMATION
> FREE SECURITY ALERT
> SECURITY SOLUTIONS
> HELP
Home
Security Scan
Virus Detection
Security Scan Results
Security Scan Statistics
About Security Scan
Security Status: At Risk!
You are vulnerable to at least one form of security threat.
Solution: Install All-In-One Security
= At Risk!
= Possible Risk!
= Safe
Hacker Exposure Check
Show Details
Windows Vulnerability Check
Show Details
Trojan Horse Check
Show Details
Some checks did not complete
Show Details
Norton 360
Norton 360™: Keeps hackers out and personal data in with comprehensive, automated protection with our proven PC Security & PC tuneup technologies PLUS new antiphishing and automated backup.
MORE INFO
SEE A DEMO
Compare Products
Готово
Интернет

Рис. 7.3


http://security.symantec.com - Symantec Security Check - Microsoft Internet Explorer
symantec.
Symantec Security Check
> SECURITY INFORMATION
> FREE SECURITY ALERT
> SECURITY SOLUTIONS
> HELP
Home
Security Scan
Virus Detection
Security Scan Results
Security Scan Statistics
About Security Scan
Security Status: At Risk!
You are vulnerable to at least one form of security threat.
Solution: Install All-In-One Security
= At Risk!
= Possible Risk!
= Safe
Hacker Exposure Check
Hide Details
Description:
Tests your TCP ports for unauthorized Internet connections.
Analysis:
WARNING! Unauthorized persons can make connections to your computer. This means your computer and data are not safe from hackers. To learn more about why you are at risk, view a detailed analysis of your test results.
Recommendation:
Install a personal firewall. If you already have a personal firewall, make sure it is correctly configured for optimum protection.
Windows Vulnerability Check
Show Details
Trojan Horse Check
Show Details
Some checks did not complete
Show Details
Norton 360
Norton 360™: Keeps hackers out and personal data in with comprehensive, automated protection with our proven PC Security & PC tuneup technologies PLUS new antiphishing and automated backup.
MORE INFO
SEE A DEMO
Compare Products
Интернет

Рис. 7.4

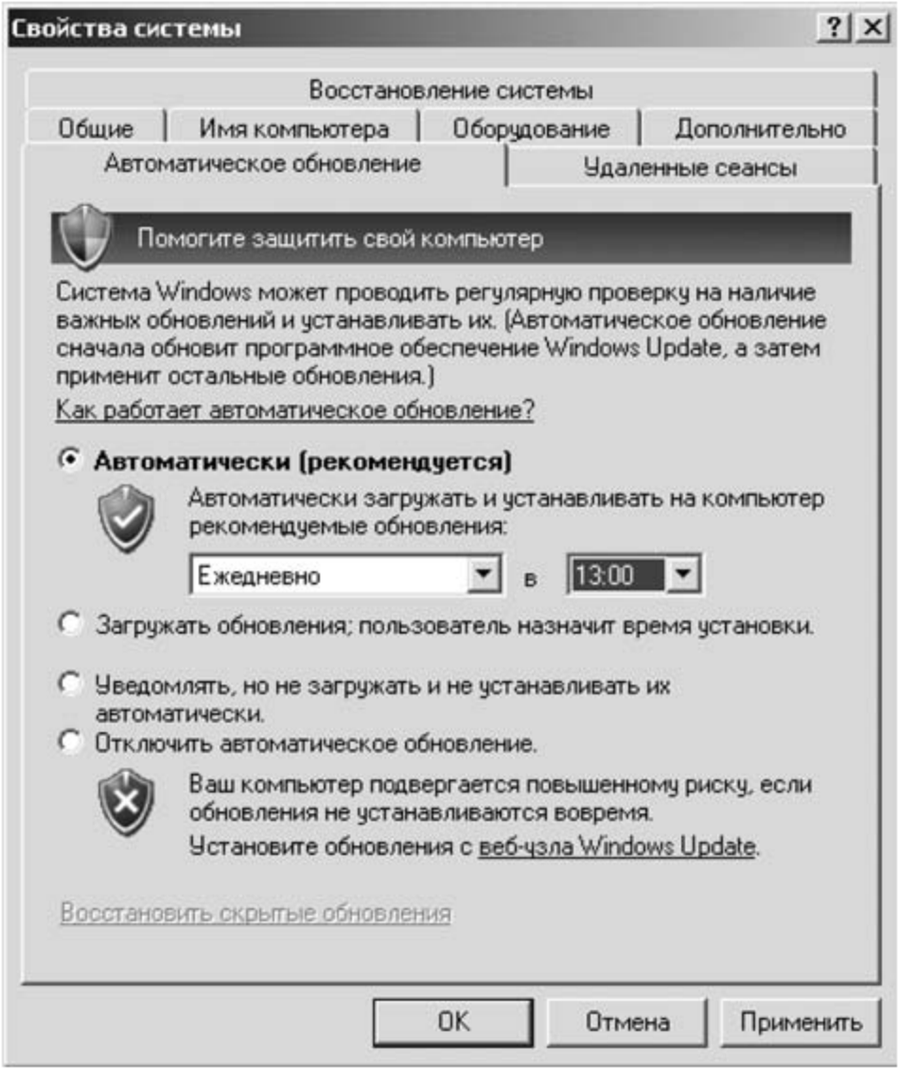


Рис. 7.5

В Windows XP встроен брандмауэр Internet Connection Firewall (ICF). Новая версия брандмауэра, являющаяся частью пакета обновлений Service Pack 2, имеет ряд новых возможностей, упрощающих работу с брандмауэром и обеспечивающих высокий уровень безопасности. Брандмауэр по умолчанию отключен. Для его использования нужно выполнить два действия:

1) в главном меню выбрать команду Выполнить, затем в поле ввода открывшегося окна набрать строку firewall.cpl и щелкнуть по кнопке OK;
2) после открытия диалогового окна (рис. 7.6) установить переключатель Включить и щелкнуть по кнопке OK.

По умолчанию брандмауэр блокирует все подключения, поэтому его нужно настроить так, чтобы трафик определенных приложений мог проходить через брандмауэр. Настройка заключается в указании программ, трафик которых не должен блокироваться брандмауэром.

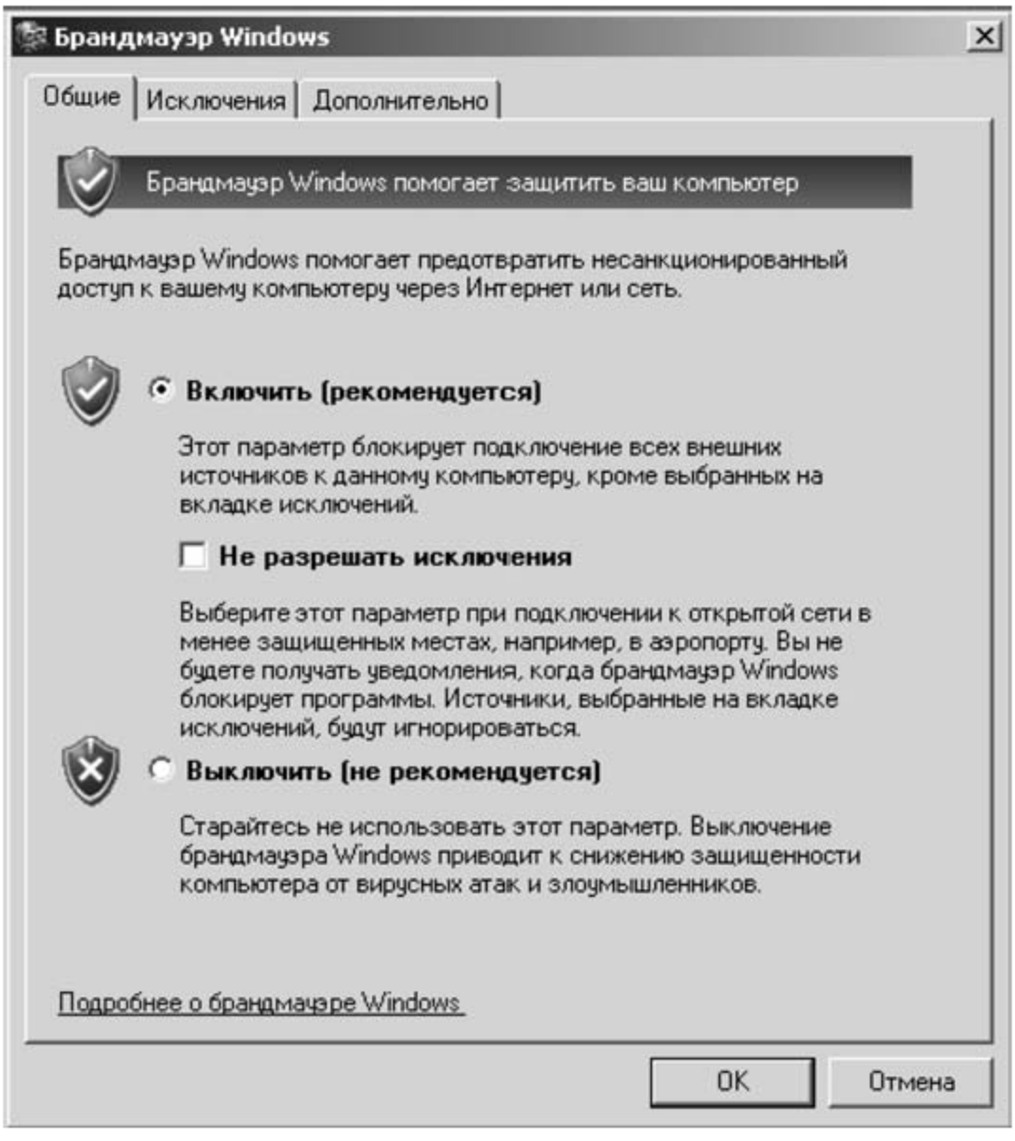


Рис. 7.6

Для открытия брандмауэра для определенного приложения нужно выполнить следующие шаги:

1) перейти на вкладку Исключения (рис. 7.7);
2) просмотреть список всех разрешенных программ (слева от названий таких программ установлен флажок). Целесообразно сбросить флажки для всех программ, которые не предполагается использовать;
3) если нужно добавить в список исключений новое приложение, которое должно обрабатывать подключения и данные из внешнего мира, следует щелкнуть на кнопке Добавить программу;
4) из предложенного списка программ выделить название программы, щелкнуть по кнопке OK, после чего название программы появится в списке;

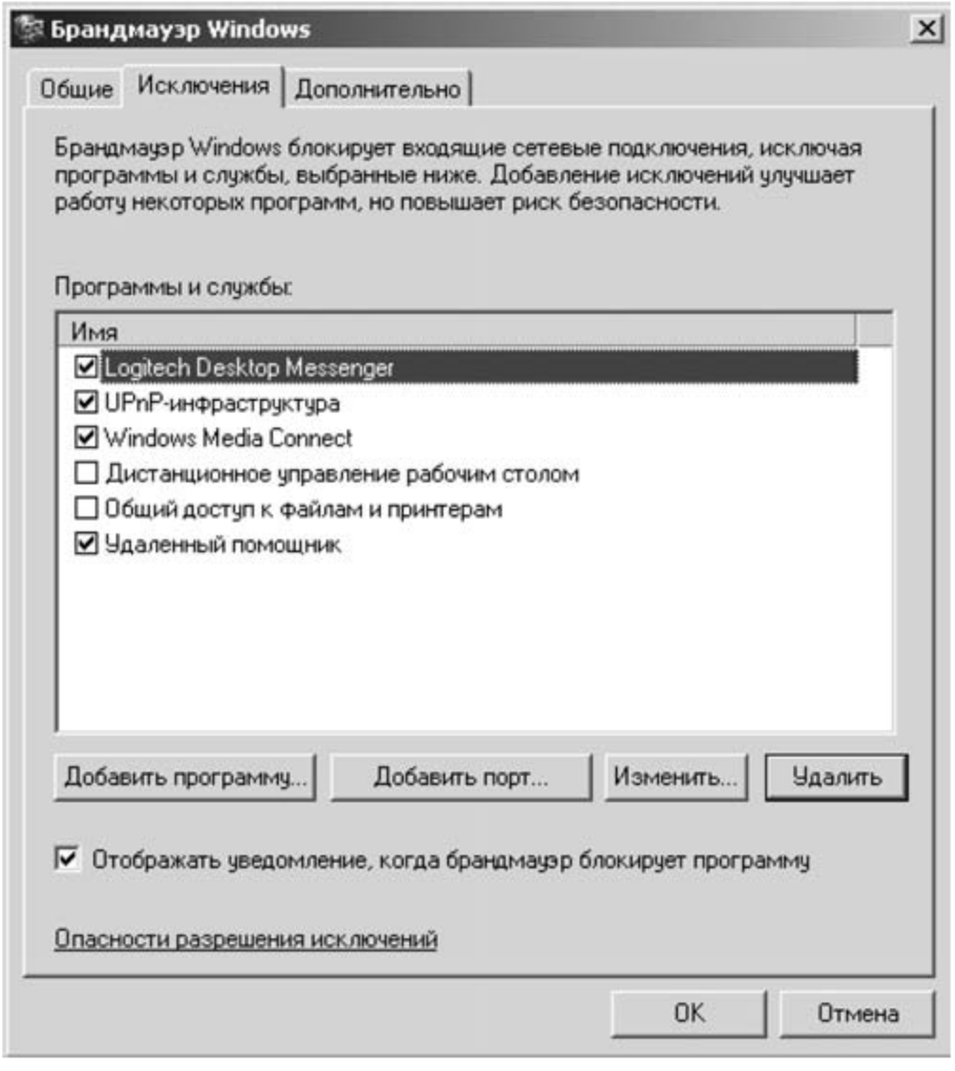


Рис. 7.7

5) установить флажок возле имени добавленного приложения и щелкнуть OK для активизации новых параметров брандмауэра.

Брандмауэр Windows позволяет задать режим ответа компьютера в случае посылки ему некоторых стандартных управляющих интернет-сообщений. Например, можно разрешить или запретить команду ping, которая используется для оценки интервала времени между посылкой данных какому-либо компьютеру и получением от него ответа. Для изменения соответствующего параметра следует перейти на вкладку Дополнительно и щелкнуть по кнопке Параметры в разделе Протокол ICMP. Откроется диалоговое окно Параметры ICMP (рис. 7.8). Если требуется, чтобы компьютер был невидим в Интернете, нужно сбросить все флажки в данном окне.

Брандмауэр Windows XP относится к брандмауэрам одностороннего типа, т.е. может блокировать только входящий трафик. Компа-

ния Zone Labs разработала двухсторонний брандмауэр ZoneAlarm, который поставляется в двух вариантах: профессиональная версия и бесплатная версия (базовый вариант двухстороннего брандмауэра), которую можно загрузить с сайта www.zonealarm.com. Двухсторонний брандмауэр может блокировать не только входящий, но и исходящий трафик, который пытаются отослать приложения с компьютера пользователя.

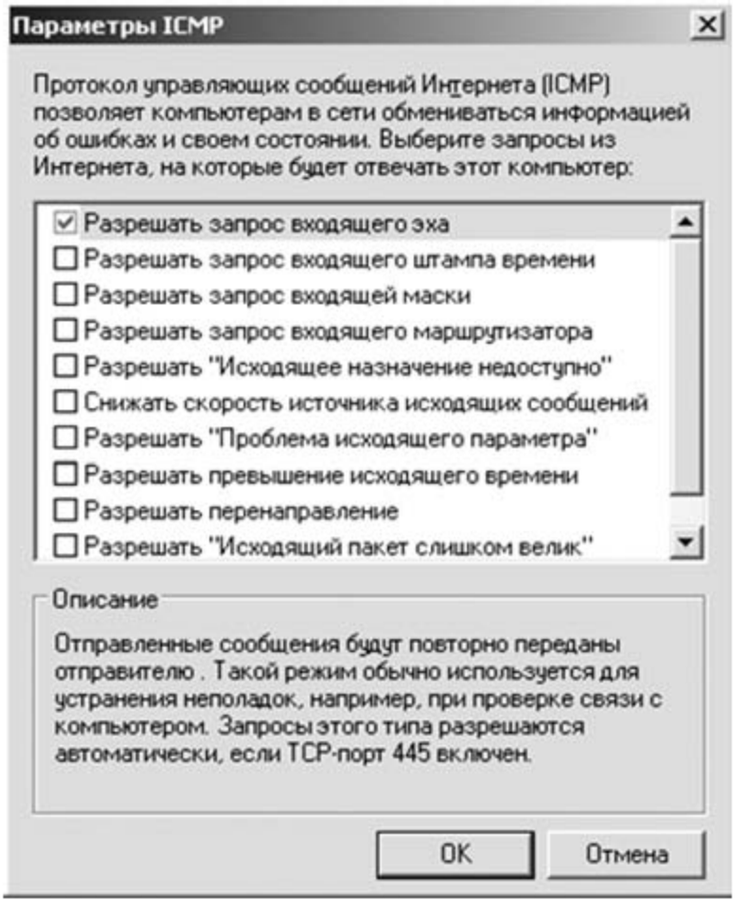


Рис. 7.8

Зачем нужно блокировать исходящий трафик? Например, если пользователь заботится о своей конфиденциальности и не желает, чтобы приложения, установленные на компьютере, связывались с веб-сайтом разработчика для пересылки туда данных, проверки обновлений или лицензий. Кроме того, очень полезной является возможность контролировать, какие приложения получают доступ к Интернету. Особенно эффективен такой брандмауэр в том случае, если пользователь разрешает коллегам иногда работать на своем компьютере. В этом случае недобросовестный коллега, установивший программу Троянский конь, не получит желаемого результата [13]. Двухсторонние брандмауэры типа ZoneAlarm делают подобные при-

ложения бесполезными, так как подобные вредоносные программы оказываются изолированными и не могут связаться с Интернетом.

Для установки, настройки и запуска ZoneAlarm нужно выполнить следующие действия:

1) загрузить копию программы с сайта www.zonealarm.com (рис. 7.9);

Рис. 7.9

2) выполнить инструкции мастера Configuration Wizard для настройки политики компьютера и запустить программу (рис. 7.10);

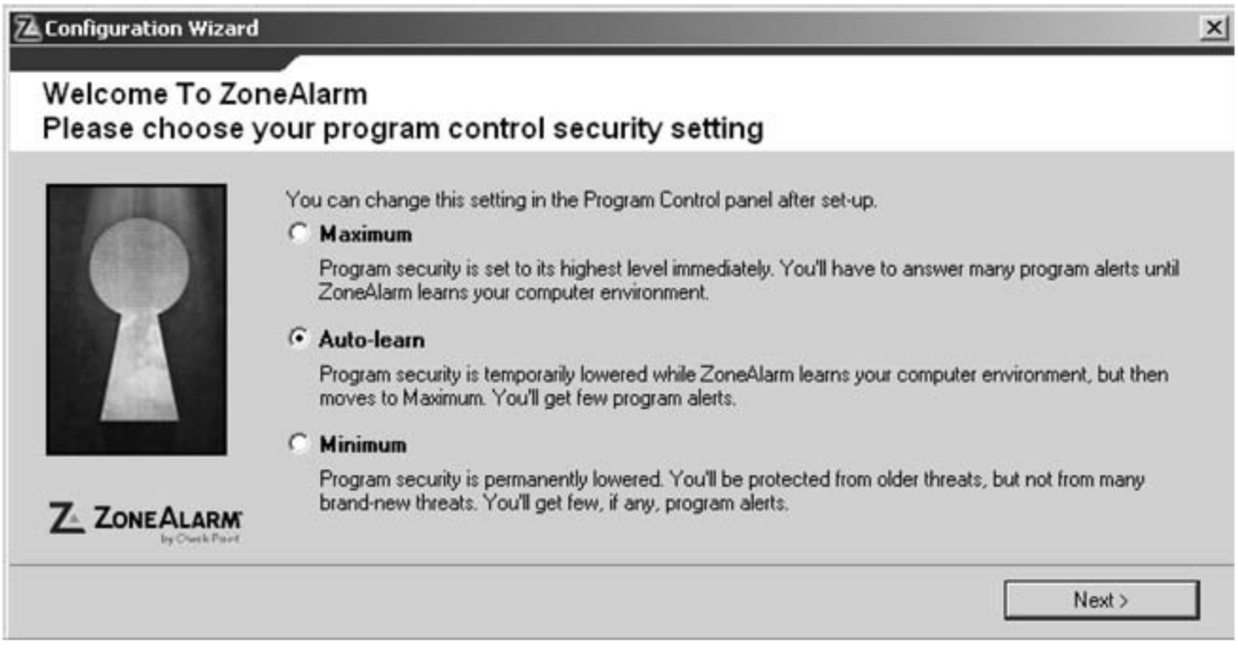


Рис. 7.10

3) установить режим работы брандмауэра (рис. 7.11). Такой режим устанавливается:
   — для зоны Интернета (защита от незнакомых компьютеров). На рисунке 7.11 показан средний уровень защиты, при котором другие компьютеры могут видеть защищаемый компьютер, но не могут использовать его ресурсы. Такой уровень защиты рекомендуется для временной работы в зоне Интернета,
   — для зоны надежных узлов Интернета (зона доверия), в которой предполагается совместная работа с компьютерами. На рисунке 7.11 изображен средний уровень защиты, при котором другие компьютеры могут видеть защищаемый компьютер и могут использовать его ресурсы,
   — для зоны блокированных узлов, через которые соединения запрещены. В эту зону включаются компьютеры, к которым нет доверительного отношения;

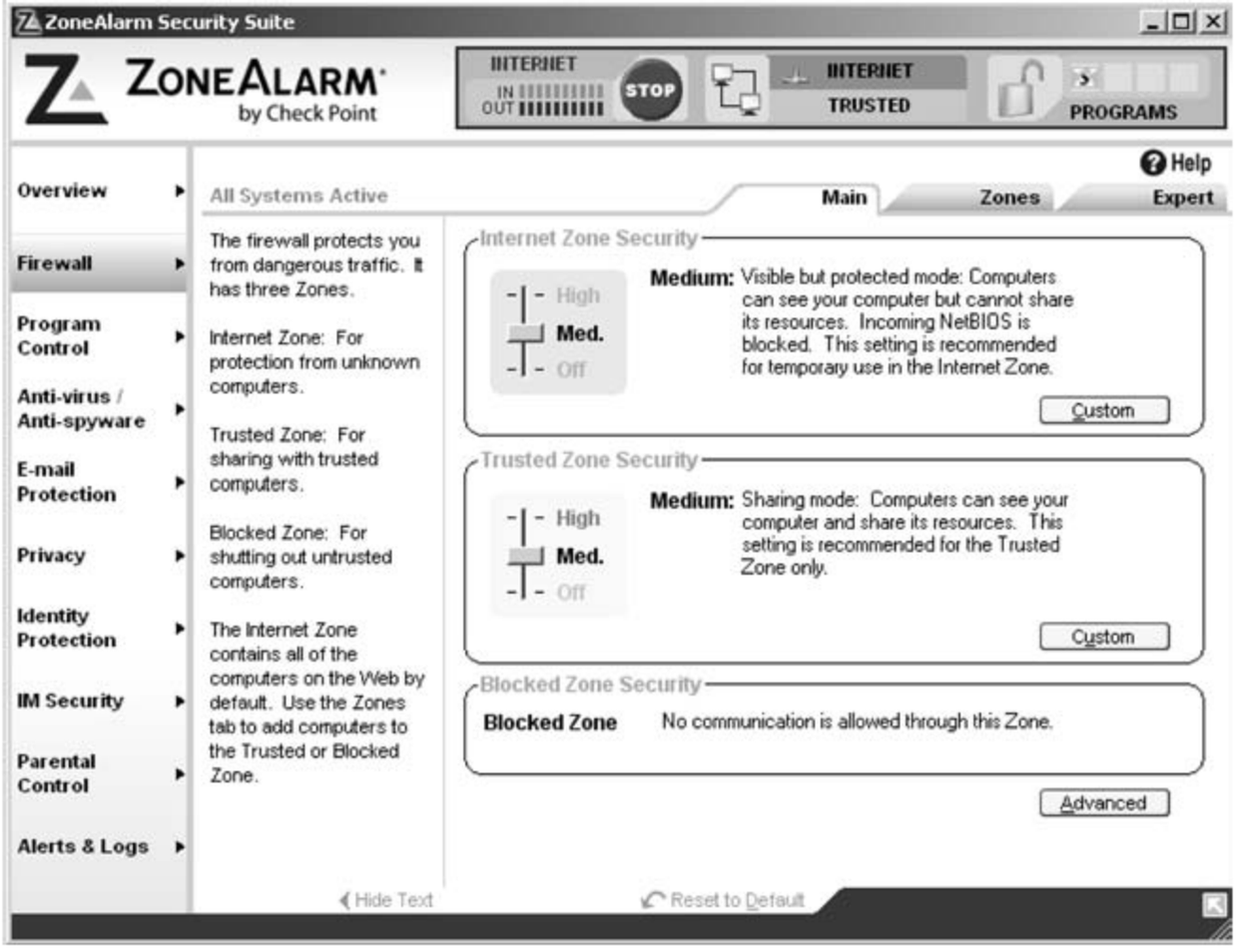


Рис. 7.11

4) если требуется настроить параметры блокировки приложений, следует щелкнуть на ссылке Program Control (рис. 7.12), а затем на вкладке Main задать желаемый уровень контроля. Более де-

тальный уровень контроля по каждому приложению можно задать на вкладке Programs (рис. 7.13). По умолчанию некоторые программы (например, Internet Explorer) всегда имеют доступ в Интернет. Однако при первом запуске программы, которой требуется выход в Интернет (например, Windows Messenger), ZoneAlarm спросит (Ask), действительно ли нужно пропустить трафик этого приложения (рис. 7.14). Нажав кнопку Option, можно получить сведения о выбранной программе (рис. 7.15). Если ничего не известно о программе, запрашивающей доступ в Интернет, следует поискать в Интернете информацию об этой программе. Возможно, что такой информации там нет и будет сделан вывод, что это spyware-программа, которую нужно удалить;

5) список программ, трафик которых пропускается через брандмауэр можно добавить нужные элементы, щелкнув по кнопке Add;
6) после установки всех настроек щелкнуть по кнопке Finish.

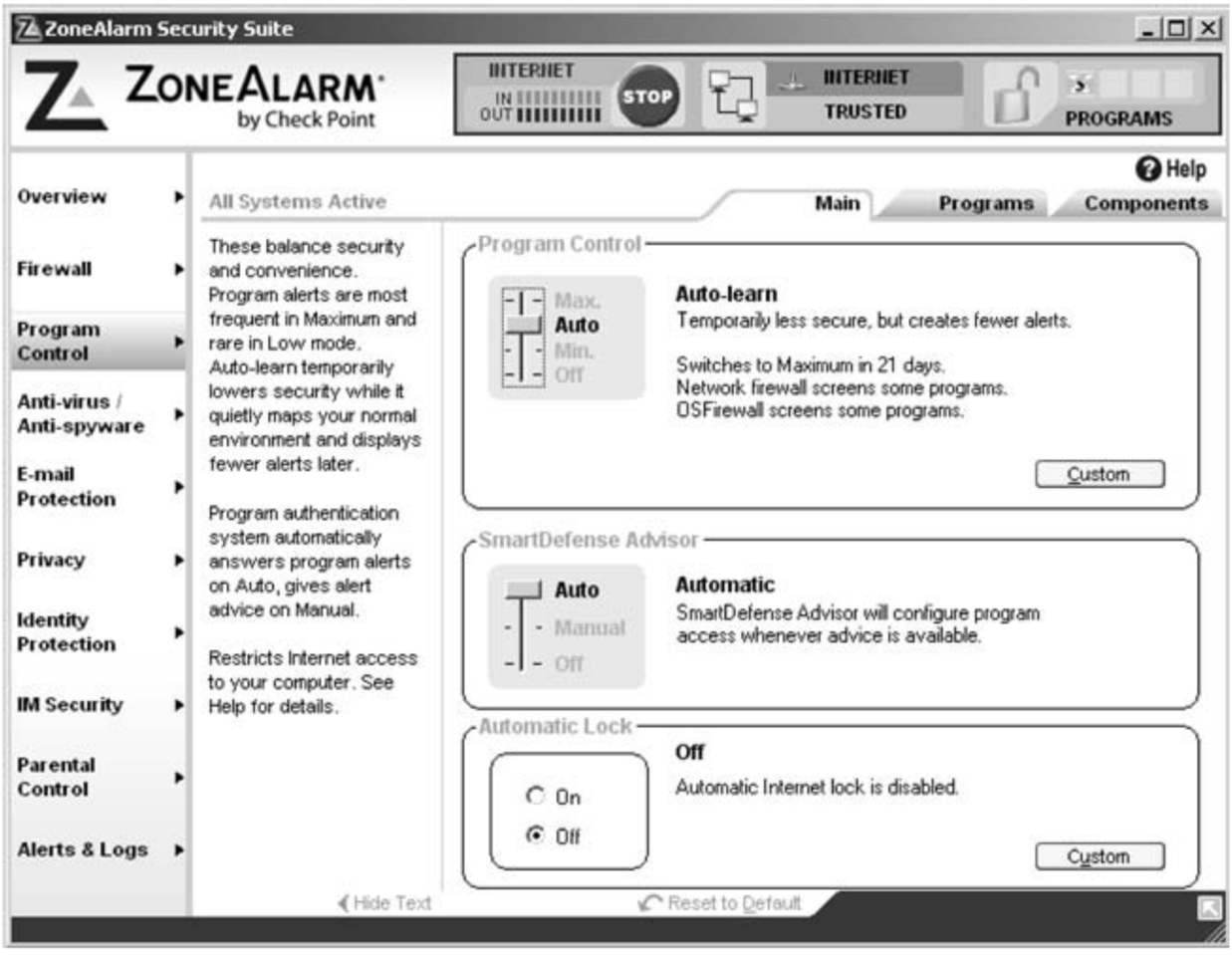


Рис. 7.12

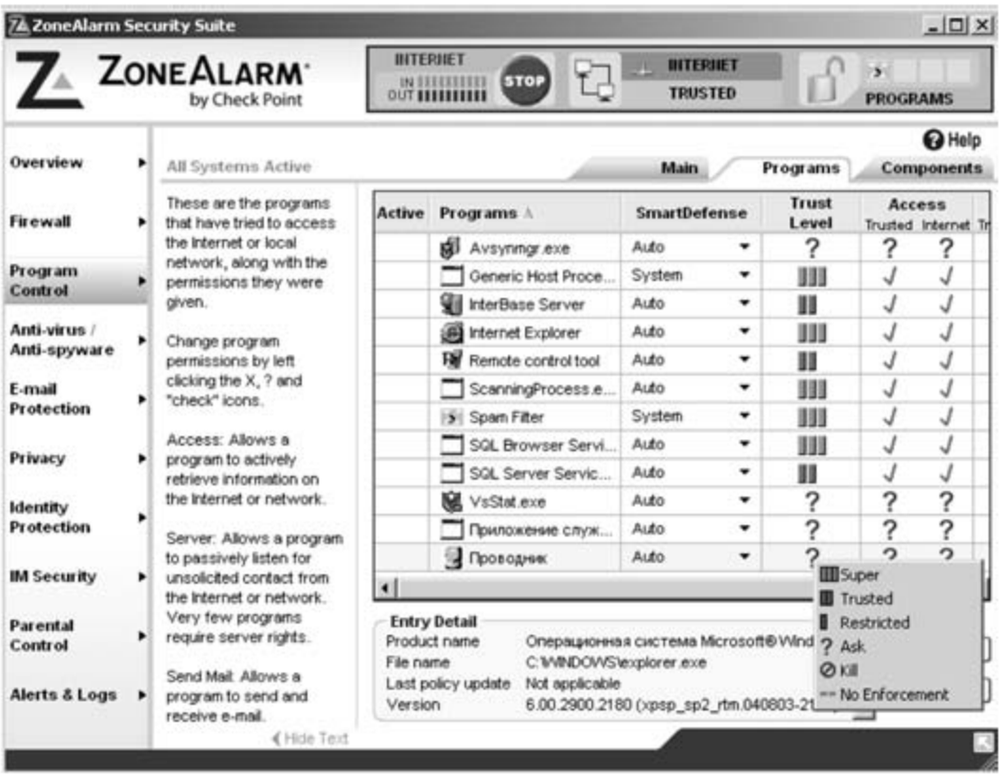


Рис. 7.13

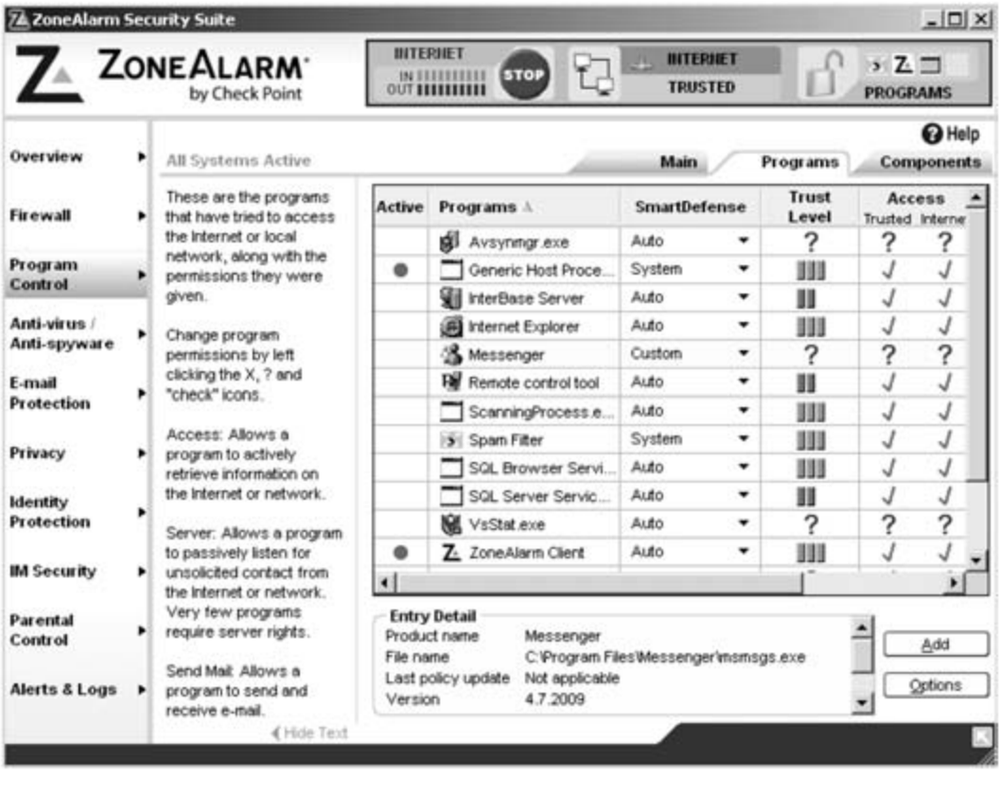


Рис. 7.14

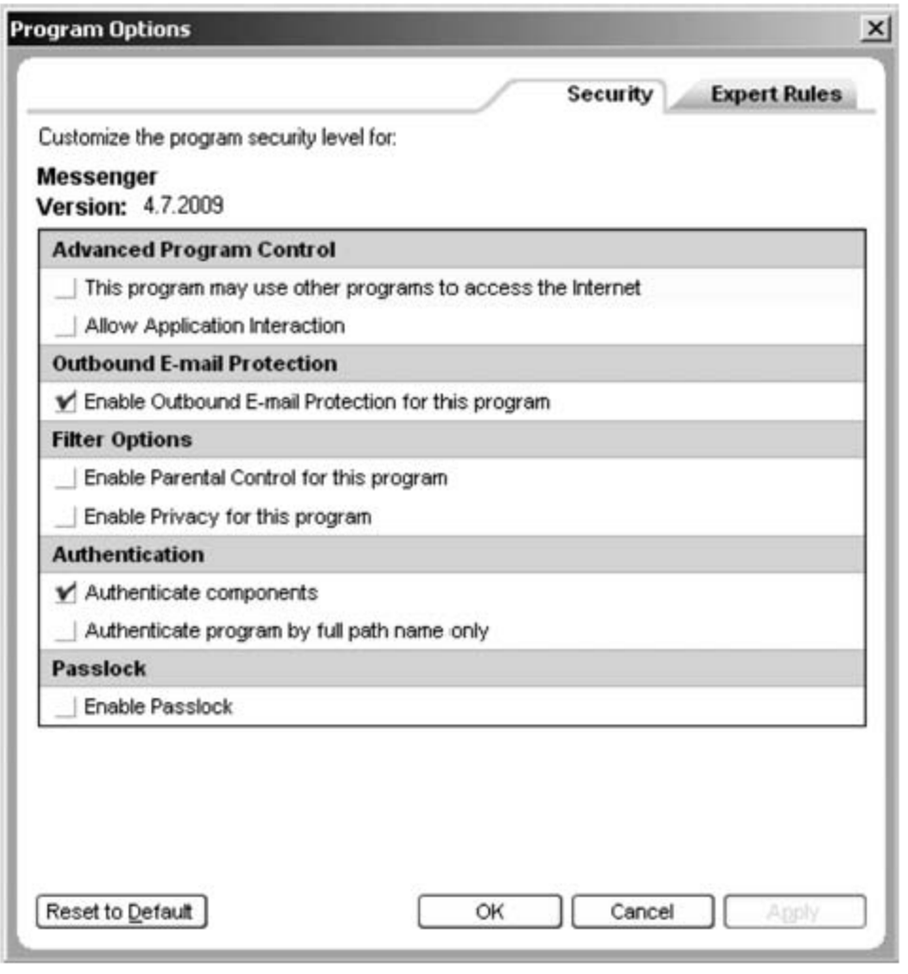


Рис. 7.15

Задания для самостоятельной работы

1. Настроить брандмауэр Windows XP. Определить список программ, которым разрешено обрабатывать данные, поступающие в компьютер из внешнего окружения. Не нужно ли сократить этот список?
2. Установить брандмауэр ZoneAlarm (предварительно отключив брандмауэр Windows XP, чтобы не допустить конфликтов). Определить, какие ваши приложения пытаются посылать данные в Интернет.

7.2. Отключение неиспользуемых служб

В подразделе 6.6 было рассмотрено, какие системные службы можно отключить, чтобы повысить производительность компьютера.

Остановимся на службах, которые можно отключить для повышения защищенности компьютера.

Запрет на подключение удаленного рабочего стола. Удаленный рабочий стол в Windows XP — компонент ОС, позволяющий получить доступ к своему компьютеру в те моменты, когда пользователь находится вдали от своего офиса или дома. Однако, если компьютер недостаточно хорошо защищен, удаленный рабочий стол может стать отличным средством для любого злоумышленника, пытающегося проникнуть на чужой компьютер и установить над ним полный контроль. Вся защита удаленного рабочего стола основывается на пароле, который во многих случаях несложно подобрать. В связи с этим, если удаленный рабочий стол не используется, его лучше отключить. Для этого нужно сделать следующее:

- щелкнуть правой кнопкой мыши на значке Мой компьютер и выбрать в контекстном меню команду Свойства;
- в открывшемся окне перейти на вкладку Удаленные сеансы (рис. 7.16), позволяющую задать параметры удаленного доступа;

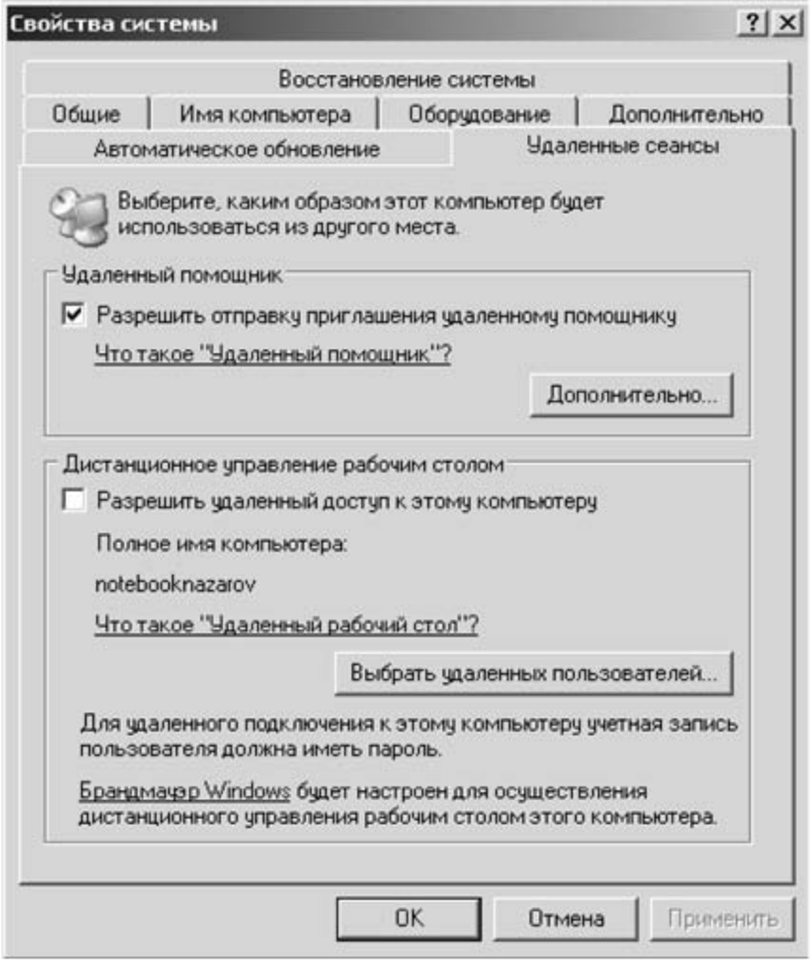


Рис. 7.16

■ сбросить флажки в разделах Удаленный помощник и Дистанционное управление рабочим столом. Щелкнуть по кнопке OK для сохранения изменений.

Отключение службы сообщений. В последних версиях Windows предусмотрена служба, позволяющая системному администратору посылать сообщения всем компьютерам в локальной сети. Это очень продуктивная служба, если ее правильно использовать. Некоторые пользователи, знающие про эту службу, могут злоупотреблять ею, рассылая сообщения и, хуже того, спам всем пользователям сети. Таким образом, пользователи сети будут получать спам не только через свой почтовый ящик, но и в неожиданно всплывающих диалоговых окнах.

Служба сообщений, как и любая другая программа, имеющая доступ во внешнюю сеть, является потенциальной угрозой безопасности компьютера. Поэтому из соображений безопасности службу сообщений лучше отключить. Для этого следует выполнить команды: Пуск — Программы — Администрирование — Службы. В открывшемся окне Службы выбрать из списка служб строку Служба сообщений, щелкнуть на ней правой клавишей мыши и выбрать в контекстном меню команду Свойства (рис. 7.17). Далее в раскрывающемся списке Тип запуска выбрать пункт Отключено и щелкнуть по кнопке OK для сохранения изменений.

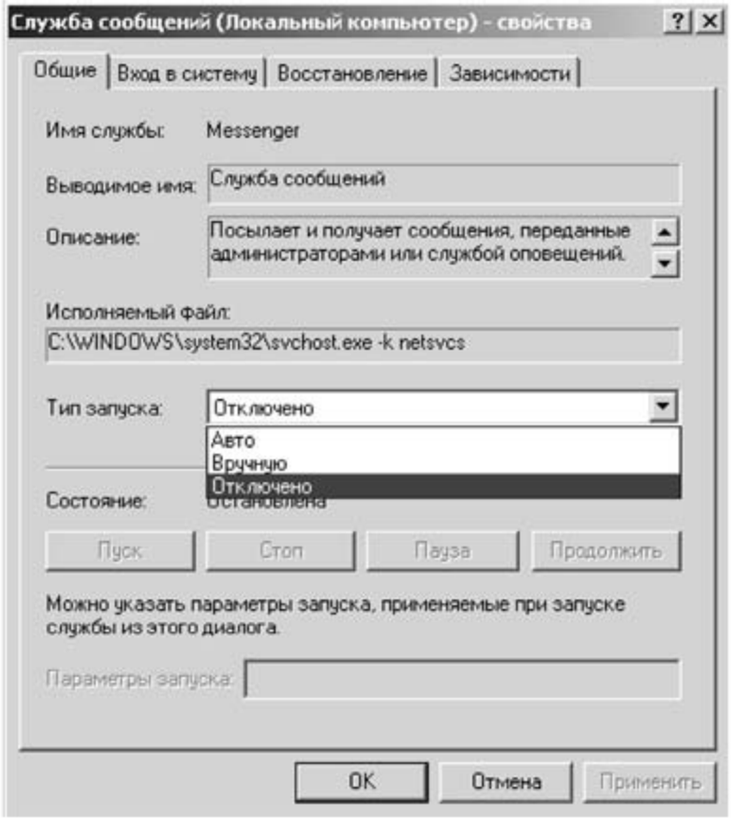


Рис. 7.17

Отключение поддержки универсальной технологии Plug-and-Play. Универсальная технология Universal Plug-and-Play (UPnP) представляет собой развитие технологии Plug-and-Play. Она позволяет быстро и просто добавлять и контролировать самые различные устройства. Учитывая низкую в настоящее время распространенность устройств UPnP и факт снижения уровня безопасности при использовании службы поддержки таких устройств, ее лучше отключить. Для этого нужно поступить так же, как и при отключении службы сообщений, но в перечне служб выбрать Узел универсальных PnP-устройств (рис. 7.18). Далее в раскрывающемся списке Тип запуска выбрать пункт Отключено и щелкнуть по кнопке OK для сохранения изменений.

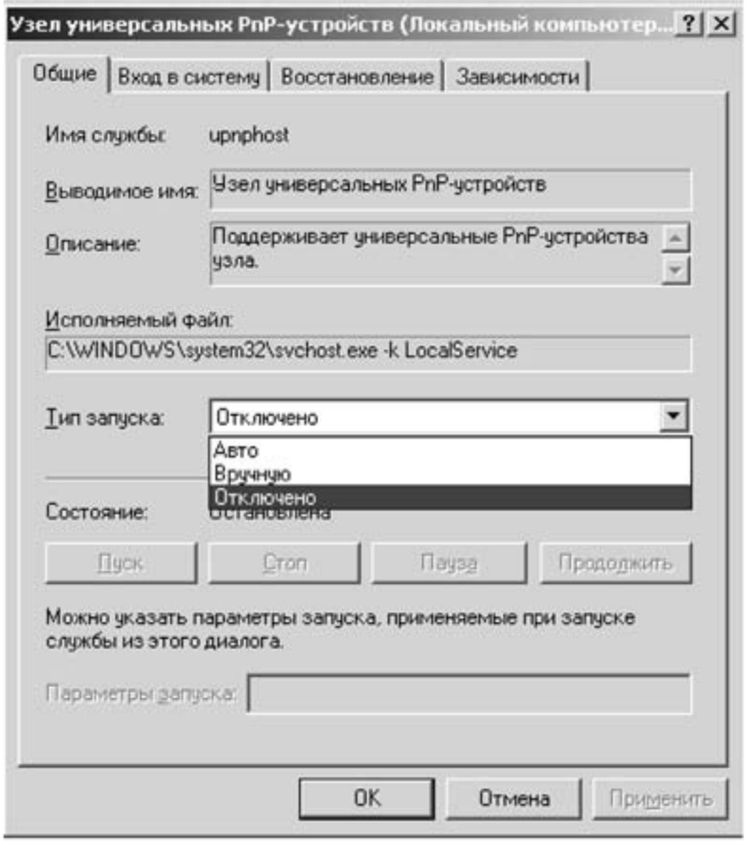


Рис. 7.18

Отключение удаленного доступа к реестру. В состав Windows XP Professional входит служба Удаленный реестр, позволяющая пользователям с правами администратора подключаться к реестру компьютера и редактировать его. Чтобы не дать кому-либо дополнительный шанс проникнуть в один из наиболее важных компонентов ОС, лучше отключить эту службу (рис. 7.19).

Отключение поддержки DCOM. Поддерживаемая Windows технология DCOM (Distributed Component Object Model — распределенная объектная модель программных компонентов) предоставляет удобный

интерфейс программирования для разработчиков сетевых приложений. Эксплойты, построенные на базе уязвимости DCOM, позволили распространиться итернет-червю по сотням тысяч машин с ОС Windows. Большинство пользователей может отключить эту службу (исключение составляют лишь те пользователи, которые пользуются приложениями, реально требующими поддержки DCOM).

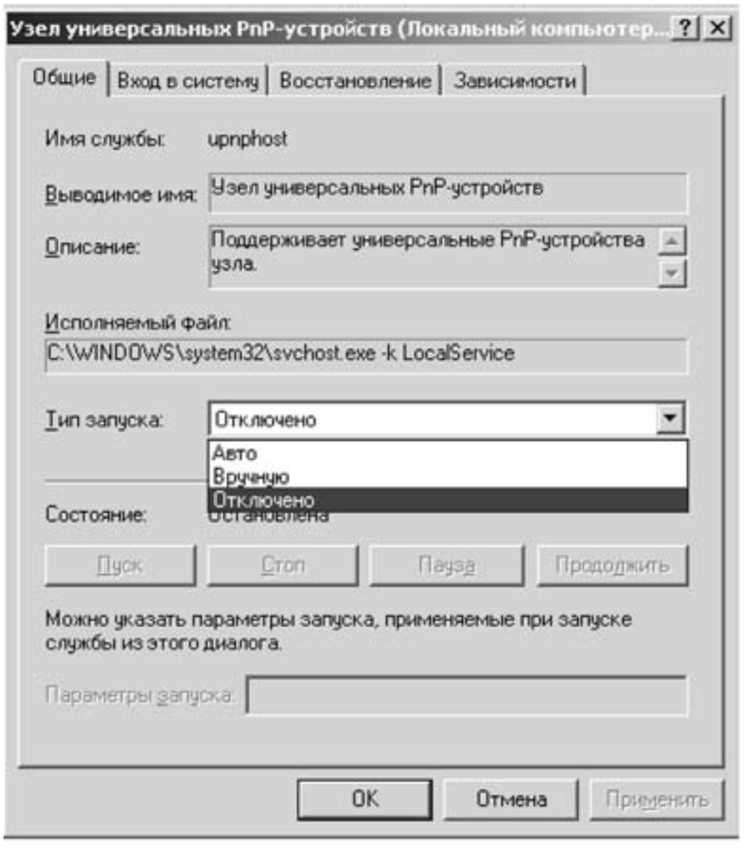


Рис. 7.19

Компания Gibson Research разработала утилиту DCOMbobulator, которая поможет отключить DCOM на компьютере. Утилиту можно загрузить с сайта www.grc.com/dcom/. После ее запуска открывается окно (рис. 7.20), в котором нужно перейти на вкладку DCOMbobulator Me! и щелкнуть по кнопке Disable DCOM, а затем по кнопке Exit.

7.3. Защита от спама

Наиболее распространенной причиной получения спама являются сами пользователи [13]. Они посылают электронные сообщения на веб-сайты или в компании, которые в ответ рассылают им свою рекламу или продают их адреса другим компаниям. Еще одна распространенная причина получения спама — невнимательная подписка на различные новости и информационные рассылки.

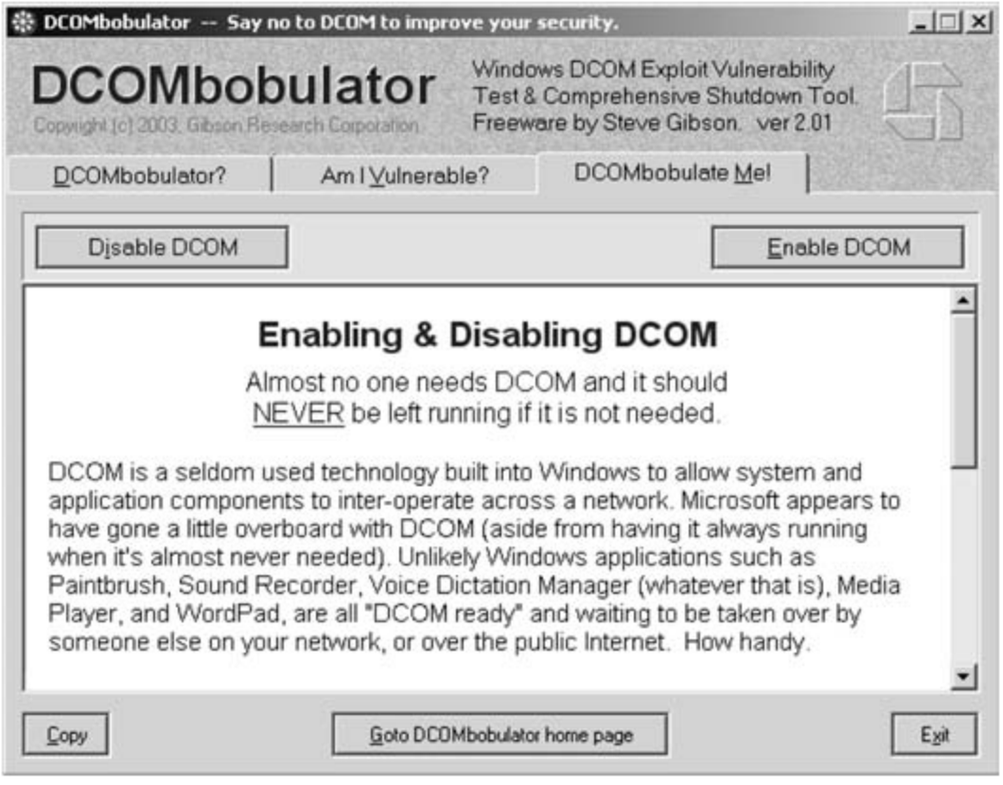


Рис. 7.20

Поскольку программ для рассылки спама создано огромное количество, то непросто выбрать лучшую программу для борьбы с нежелательными рассылками. Не существует программы, которая фильтрует спам со 100%-ной гарантией. Если утилита будет отсекать около 90% нежелательной корреспонденции, это уже хороший результат. Одной из неплохих утилит является McAfee SpamKiller — программа для защиты от спама с ежедневным автообновлением базы по спамерам и легким созданием собственных фильтров. Работает SpamKiller в фоновом режиме, проверяет практически неограниченное число почтовых ящиков, выявляет полученный спам и удаляет его прямо на почтовом сервере — автоматически или в ручном режиме. В случае обнаружения новой почты возможна разнообразная сигнализация об этом, а также автоматический запуск почтовой программы.

Условно-бесплатный вариант программы имеется на множестве сайтов Интернета, например на сайте dl.softportal.com/load/spamkiller2908.exe. После загрузки и запуска программы открывается окно для инсталляции утилиты (рис. 7.21). Установленная утилита после ее запуска предлагает купить лицензионную версию или продолжить работу с 30-дневной демо-версией (рис. 7.22).

Нажав кнопку Continue, пользователь перейдет к мастеру установки параметров программы, который предлагает импортировать адресную книгу, используемую в e-mail-программе, запрашивает тип адрес-

ной книги (например, MS Outlook Express, Netscape Messenger и др.), предлагает ввести информацию о почтовых учетных записях, имя, пароль и т.д. После завершения работы с мастером настройки программы и нажатия кнопки Finish открывается окно программы (рис. 7.23), в котором можно просмотреть и установить параметры утилиты для фильтрации сообщений, в том числе поступающих из разных стран.

Рис. 7.21

Рис. 7.22

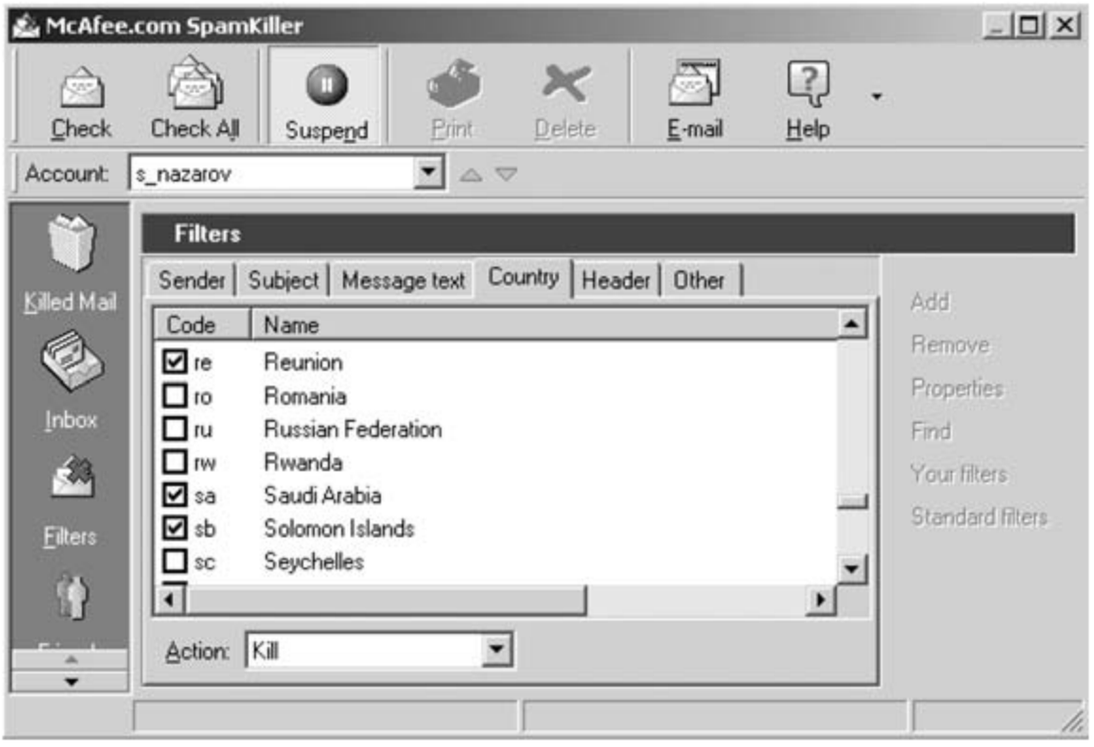


Рис. 7.23

Если почтовый клиент пользователя поддерживает графические сообщения в формате HTML, то при каждом получении почты имеется вероятность того, что отправитель узнает, прочитал ли получатель его письмо. Это делается при помощи скрытых ссылок на изображения, обращающиеся к веб-серверу, на котором запущена специальная программа, отслеживающая подобные обращения. Когда пользователь открывает письмо со спамом, оно может послать сигнал на веб-сервер своего отправителя, сигнализируя ему, что электронный адрес получателя активен и что получатель читает приходящую почту. Если задержать сигналы, отправляемые на серверы распространителей спама, то возможно, что адрес получателя будет удален из баз данных серверов как неактивный.

Последние версии некоторых почтовых программ, например Outlook 2003 и Outlook Express (после установки Windows XP Service Pack 2), автоматически блокируют все внешние ссылки в HTML-сообщениях. Режим блокировки внешних ссылок в Outlook Express можно включить на вкладке Безопасность в окне Параметры, вызываемого из меню Сервис. В Outlook и Outlook Express также имеется список надежных отправителей, с помощью которого можно включать внешнее содержимое только для определенных отправителей. Чтобы разрешить использование рисунков и другого внешнего содержимого для определенного отправителя, нужно щелкнуть правой кнопкой

мыши на сообщении от него и добавить отправителя в список надежных отправителей.

Задания для самостоятельной работы

1. Запустить оснастку Службы. Просмотреть список установленных и работающих служб. Все ли они необходимы для вашей повседневной работы? Удалить ненужные службы.
2. Загрузить с веб-сайта программу SpamKiller. Настроить программу (установить параметры фильтрации сообщений). Проверить, установлен ли режим блокировки внешних ссылок в почтовой программе.

7.4. Защита от вредоносных программ и вирусов

В настоящее время spyware-программы превратились в наиболее опасную угрозу для компьютеров. Скрываясь в свободно распространяемых приложениях, эти программы шпионят за пользователями компьютеров и затем отправляют собранную информацию злоумышленникам. Существует еще один вид вредоносных программ — adware-программы, тесно связанные со spyware-программами. Они также тайно устанавливаются на компьютеры пользователей и наблюдают за ними. Обычно подобные ситуации связаны с установкой программ, загружаемых с веб-сайтов. Пользователи часто перед установкой бесплатных программ не читают соответствующие соглашения о предоставлении услуг и пропускают сообщения, что данные программы будут отображать рекламу.

Шпионское программное обеспечение (Spyware) — относительно новый вид угрозы, который еще недостаточно широко обрабатывается стандартными антивирусами. Шпионское программное обеспечение, не проявляя себя, отслеживает поведение пользователя за компьютером, чтобы создать его «маркетинговый профиль», который также молча передается сборщикам информации, продающим данные пользователя рекламным организациям. Если в браузере появятся новые панели инструментов, которые явно не устанавливались, если браузер постоянно «падает» или стартовая страница неожиданно изменилась, вероятно, что в компьютере завелся «шпион». Но даже если не происходит ничего необычного, то «шпионы» все равно могут появляться, поскольку со временем все больше появляется программного обеспечения такого рода.

В Интернете имеется множество свободно распространяемых утилит, помогающих проверить компьютер на наличие spyware- и adware-программ. Наибольшей популярностью пользуются две такие программы: программа Ad-ware, разработанная компанией Lavasoft (ее базовую версию можно загрузить бесплатно с сайта www.lavasoft.de); программа Spybot Search & Destray, распространяемая также бесплатно (www.spybot.info).

Для загрузки и запуска программы Ad-ware нужно выполнить следующие действия:

1) загрузить копию базовой версии программы Ad-ware с сайта www.lavasoft.de и установить ее на компьютере;
2) запустить программу, в результате чего откроется окно, представленное на рис. 7.24;

Рис. 7.24

3) обновить файлы данных, щелкнув по кнопке Web Update, а затем Update. Если имеются актуальные обновления, будет выдано соответствующее сообщение. В этом случае щелкнуть по кнопке Yes, а затем OK, после чего обновления будут автоматически загружены и установлены;
4) для начала проверки компьютера щелкнуть по кнопке Scan (рис. 7.25). Из показанных режимов сканирования Scan Mode

выбрать необходимый (например, Smart Scan) и нажать кнопку Scan. Начнется процесс сканирования (рис. 7.26);

Рис. 7.25

Рис. 7.26

5) после завершения сканирования будут показаны его результаты (рис. 7.27) с перечислением всех spyware- и adware-программ, обнаруженных на компьютере. Нужно сбросить флажки у тех объектов, которые решено не удалять (например, у объектов типа Tracking cookie). Для удаления объектов нажать кнопку Remove. Программа автоматически сохраняет резервные копии всех удаляемых объектов на случай возникновения проблем в ОС после удаления файлов и параметров реестра;

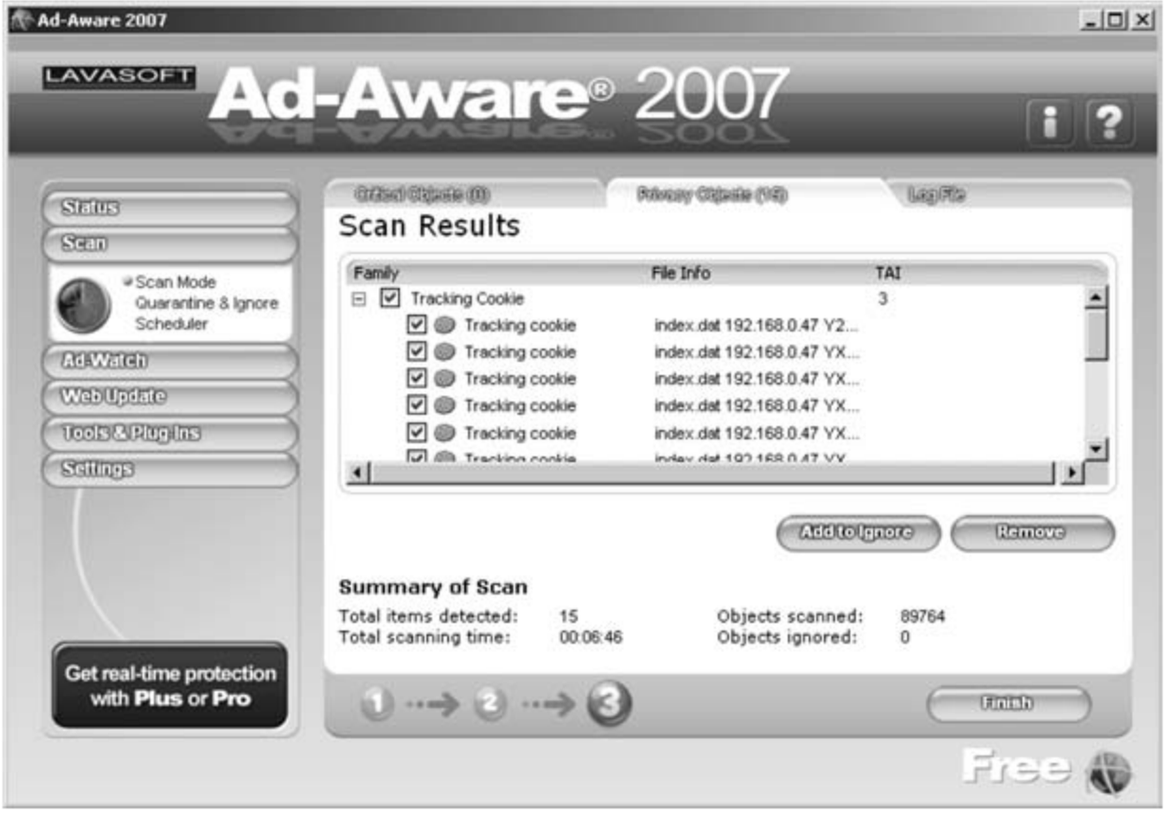


Рис. 7.27

6) для завершения работы с программой нажать кнопку Finish (рис. 7.28).

Программа Spybot Search & Destroy (спайбот — найти и уничтожить) может обнаруживать и удалять с компьютера различного рода шпионское программное обеспечение.

После загрузки, инсталляции и запуска программы открывается ее окно с подменю Spybot Search & Destroy (рис. 7.29). При первом запуске нужно прочитать несколько соглашений об ответственности за использование программы.

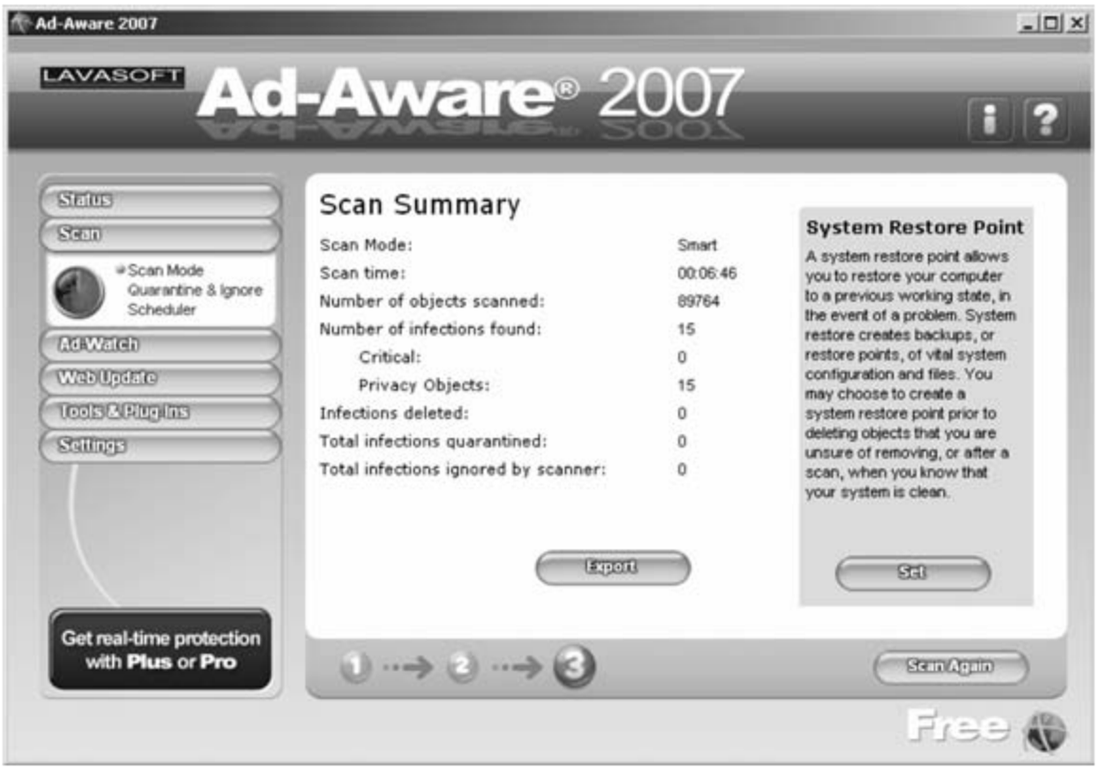


Рис. 7.28

Рис. 7.29

Далее для работы с программой Spybot Search & Destroy нужно выполнить следующие действия:

1) обновить файлы данных, для чего щелкнуть по кнопке Поиск обновлений. Просмотрев список доступных обновлений,

для их загрузки щелкнуть по кнопке Загрузить обновления (рис. 7.30);

Рис. 7.30

2) после загрузки и установки обновлений следует закрыть и заново открыть программу. Для проверки компьютера нажать кнопку Начать проверку. Начнется процесс сканирования;
3) результаты сканирования через некоторое время будут выведены в окне результатов (рис. 7.31). Лучший результат показан на рис. 7.32;
4) для устранения выявленных проблем нужно нажать клавишу Устранить отмеченные проблемы. Устранению подлежат только файлы, помеченные флажками. Программа автоматически сохраняет резервные копии всех удаляемых объектов на случай возникновения проблем в ОС после удаления файлов и параметров реестра;
5) перед устранением выявленных проблем программа создает резервную копию реестра (рис. 7.33). Если возникнут какие-либо трудности с ОС, можно восстановить удаленные файлы и исходное состояние реестра. Для этого нужно щелкнуть по клавише Восстановить. Предварительно программа предложит создать резервную копию реестра.

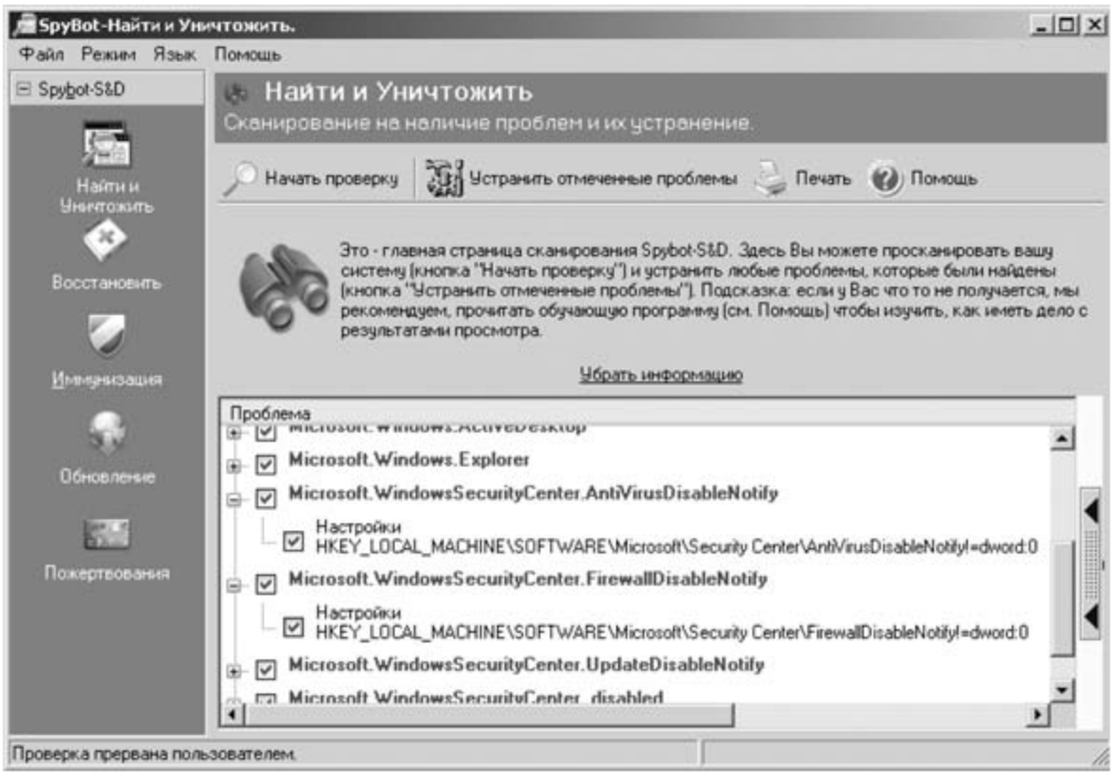


Рис. 7.31

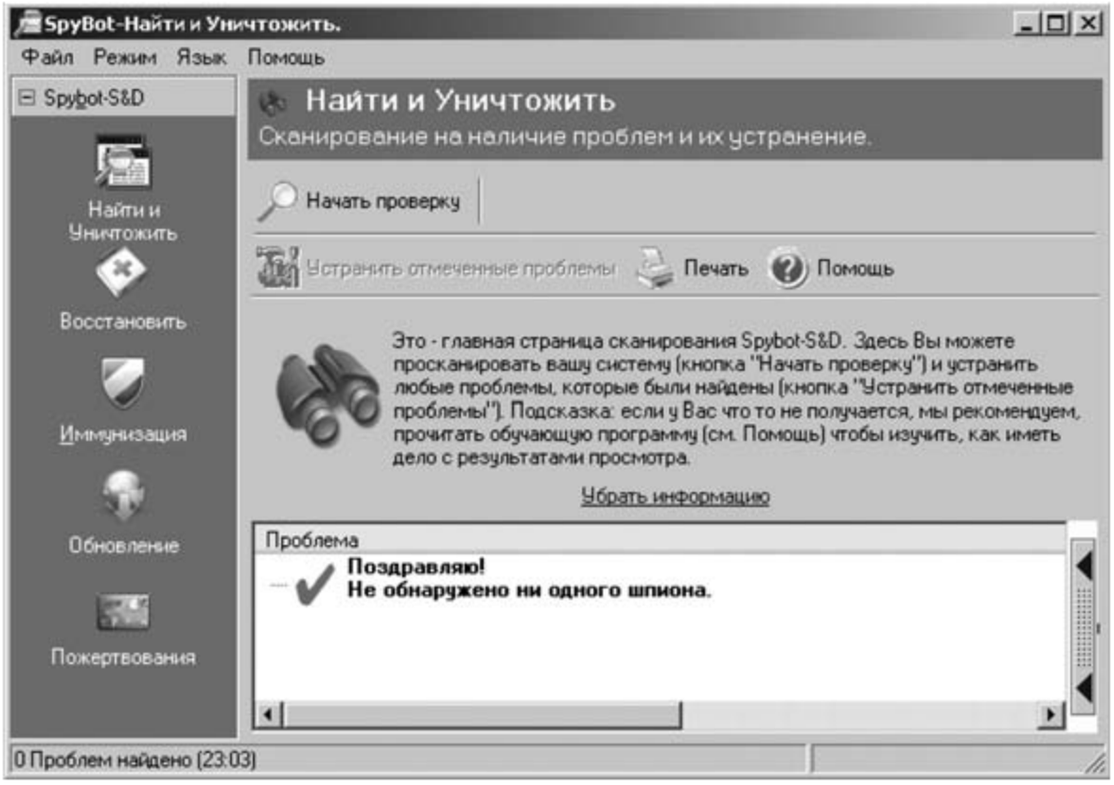


Рис. 7.32

Рис. 7.33

Утилита Spybot Search & Destroy способна не только проверять компьютер и удалять spyware- и adware-программы. С ее помощью можно выполнять вакцинацию, защищающую компьютер от некоторых наиболее распространенных типов вредоносных программ, что значительно повышает защищенность компьютера в борьбе с spyware-программами. Для выполнения вакцинации следует запустить утилиту и щелкнуть по кнопке Иммунизация (рис. 7.34).

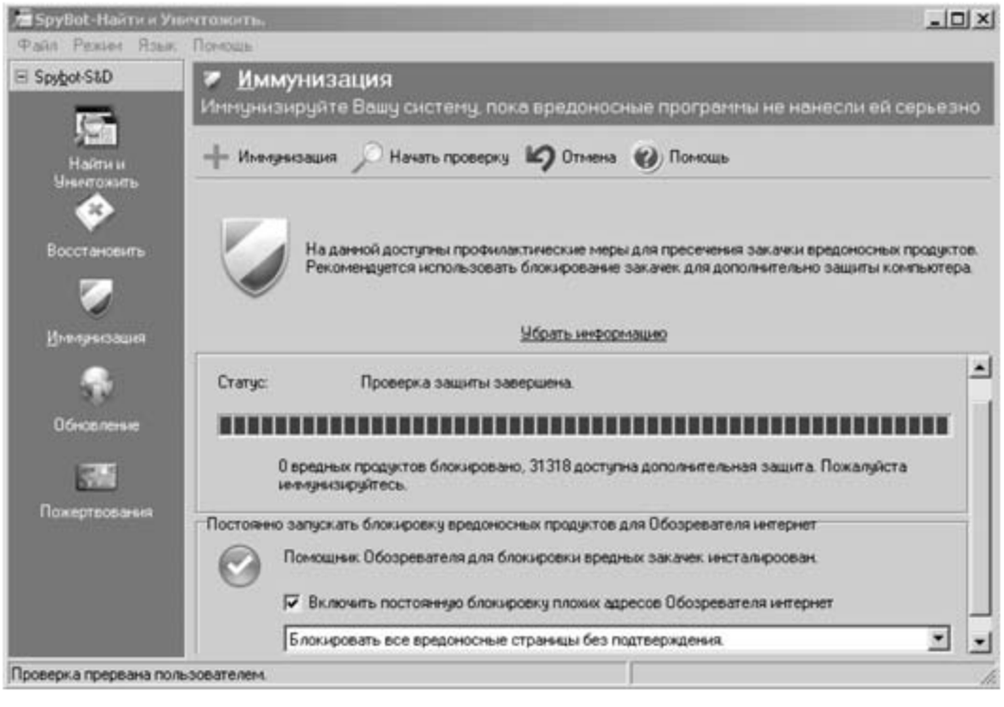


Рис. 7.34

После удаления с компьютера всех spyware- и adware-программ можно отключить некоторые режимы работы браузера Internet Explorer, снизив тем самым риск новой случайной установки spyware-программ.

Целесообразно изменить параметры установки элементов ActiveX, запретив возможность их установки. Для этого необходимо выполнить следующие действия:

1) открыть новое окно Internet Explorer;
2) выбрать команду Свойства обозревателя в меню Сервис;
3) перейти на вкладку Безопасность и щелкнуть по кнопке Другой. Откроется окно Параметры безопасности (рис. 7.35);

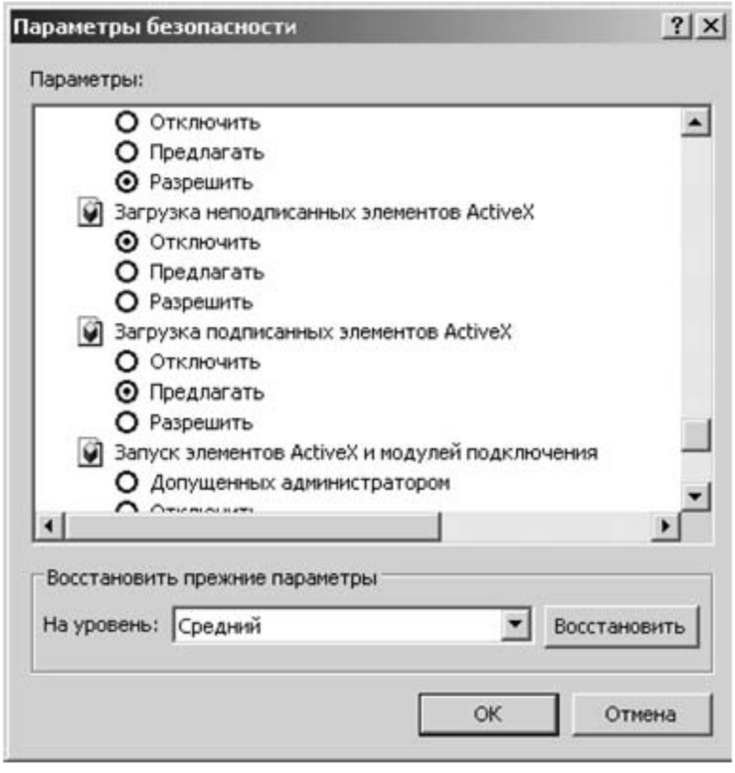


Рис. 7.35

4) найти в списке группу переключателей Загрузка подписанных элементов ActiveX и установить переключатель в состояние Отключить (загрузка неподписанных элементов также должна быть отключена);
5) щелкнуть по кнопке OK, а затем — по кнопке Да;
6) еще раз щелкнуть по кнопке OK для закрытия окна Свойства обозревателя.

Выполненная процедура приведет к запрету установки элементов управления ActiveX с любых веб-сайтов (как хороших, так и плохих). Если при посещении какого-либо сайта возникнут проблемы с загруз-

кой его содержимого, то можно выполнить обратную процедуру, т.е. разрешить загрузку подписанных элементов ActiveX.

Задания для самостоятельной работы

1. Установить и обновить утилиты Ad-ware и Spybot Search & Destroy. Проверить компьютер с помощью этих программ. Удалить обнаруженные spyware- и adware-программы. Провести вакцинацию компьютера.
2. Проверить параметры настройки браузера. Запретить загрузку элементов ActiveX.

7.5. Защита конфиденциальной информации

Современные ОС, в том числе Windows XP, собирают информацию о работе пользователя за компьютером, в том числе адреса веб-сайтов, имена запускаемых приложений и открываемых файлов. Эта информация используется системой для обеспечения комфортной работы пользователя. Однако иногда это нежелательно, например если необходимо соблюдать конфиденциальность своей работы.

Рассмотрим, как удалить с компьютера информацию о действиях пользователя, что особенно актуально, если компьютером пользуется несколько человек.

7.5.1. Очистка Internet Explorer

В целях конфиденциальности приходится очищать четыре части данных браузера: список вводившихся адресов, журнал с историей посещения веб-сайтов, список временных файлов Интернета и список cokie-файлов. Первый список формируется на основе истории ранее вводившихся адресов и позволяет быстро вводить адреса, выбирая их среди возможных вариантов. Отключить функцию автоматического завершения ввода довольно непросто [13]. Файл, в котором хранится это список, является URL-кэшем и имеет имя index.dat. Решение можно получить с помощью утилиты Dr.Delete (www.docsdownloads.com/dr-delete-1.html), для этого нужно выполнить следующие шаги:

1) запустить программу Dr. Delete и щелкнуть по кнопке Browse, чтобы выбрать удаляемый файл (рис. 7.36);
2) открыть папку C:\documents and Settings;
3) открыть папку, название которой совпадает с вашим именем пользователя (Администратор — в нашем случае на рис. 7.36);

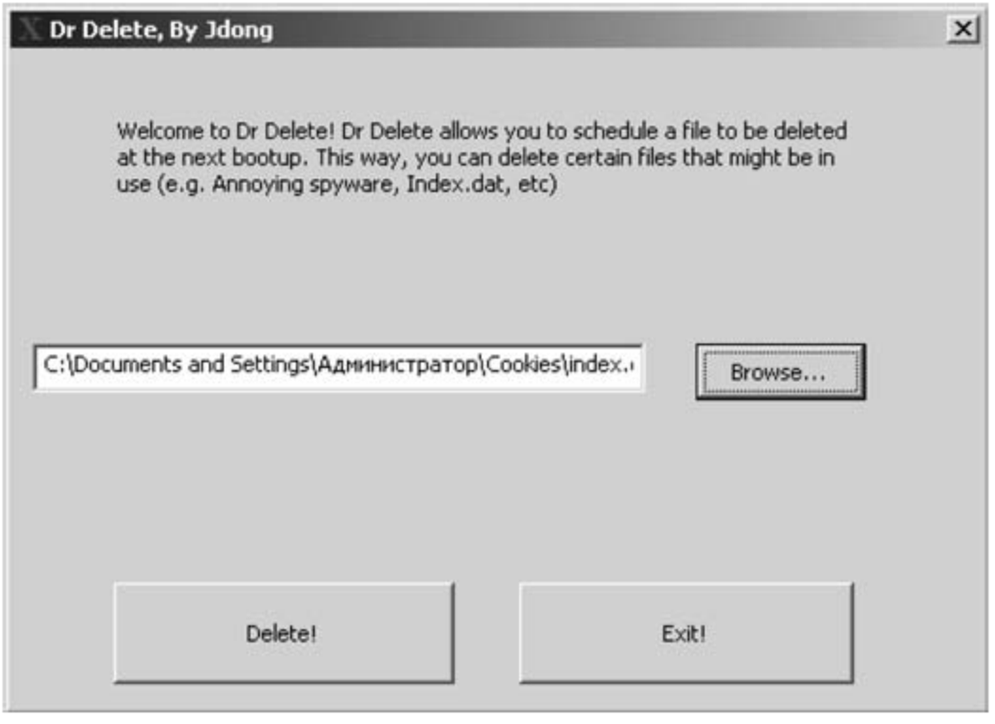


Рис. 7.36

4) открыть папку Cookies, выделить файл index.dat и щелкнуть по кнопке Open;
5) после того как в поле ввода появится путь к файлу, щелкнуть по кнопке Delete!;
6) щелкнуть по кнопке Yes в диалоговом окне подтверждения (рис. 7.37). Появится сообщение о том, что данный файл будет удален при следующей перезагрузке компьютера.

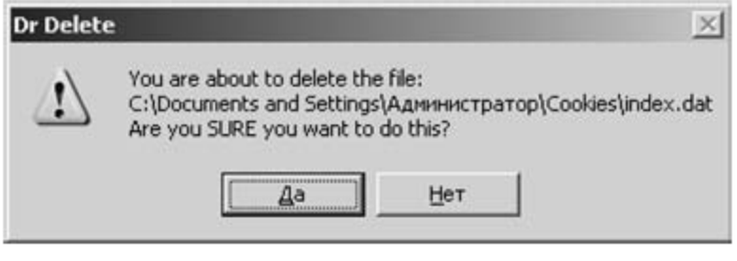


Рис. 7.37

По умолчанию Internet Explorer настроен на запись адресов веб-сайтов, которые посещались в течение 30 дней. Если важно сохранить

конфиденциальность этих посещений, следует периодически очищать журнал посещений. При этом, однако, следует помнить, что после очистки журнала посещений пользователь лишится возможности вернуться на веб-сайты, адреса которых он забыл. Для очистки журнала посещений нужно выполнить следующие действия:

1) открыть окно браузера Internet Explorer и в меню Сервис выбрать команду Свойства. Откроется окно Свойства: Интернет;
2) на вкладке Общие щелкнуть по кнопке Очистить. Появится диалоговое окно Свойства обозревателя (рис. 7.38), в котором нажать кнопку Да;

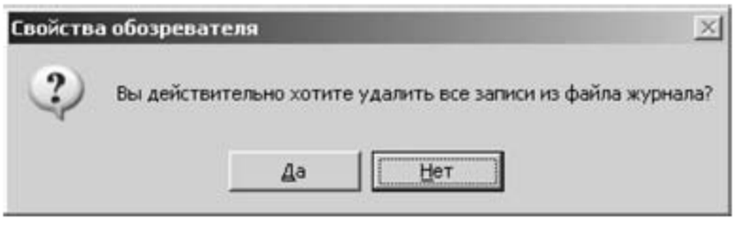


Рис. 7.38

3) указать интервал времени, в течение которого должна храниться информация о посещении сайтов. Щелкнуть по кнопке Применить и OK.

Каждый раз при посещении сайтов Интернета в компьютере в папку Temporary Internet Files записываются соответствующие файлы. Со временем папка увеличивается в объеме и может содержать информацию, которую нежелательно показывать другим пользователям этого компьютера. Однако папка доступна для просмотра каждому пользователю. Если это нежелательно, папку Temporary Internet Files следует очистить.

Еще один вид файлов, которые создаются при посещении Интернета, — Cookie-файлы. С точки зрения конфиденциальности единственный недостаток хранения этих файлов на компьютере связан с тем, что он доступен локальным пользователям этого компьютера. Следовательно, при желании они могут определить, какие сайты кто посещал. Для очистки папки Temporary Internet Files и удаления Cookie-файлов нужно:

1) открыть новое окно браузера Internet Explorer и в меню Сервис выбрать команду Свойства обозревателя;
2) щелкнуть по кнопке Удалить файлы в разделе Временные файлы Интернета. Появится запрос, показанный на рис. 7.39. Нужно установить флажок и щелкнуть по кнопке OK;

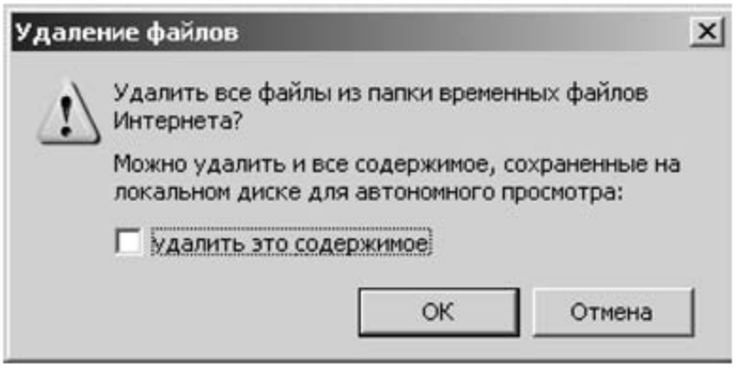


Рис. 7.39

3) после возврата в окно Свойства обозревателя нужно щелкнуть по кнопке Удалить «Cookie»;
4) щелкнуть на кнопке OK в диалоговом окне подтверждения и еще раз OK — для закрытия окна Свойства обозревателя.

В последней версии Internet Explorer появилось много новых функций, в том числе функция настройки вариантов создания cookie-файлов. Можно установить режим блокировки создания cookie-файлов. Следует различать два их типа: основные (first-party) и сторонние (third-party). Основные cookie-файлы размещаются на компьютере тем сайтом, который посетил пользователь. Сторонние cookie-файлы размещаются на компьютере удаленными сайтами (например, рекламными).

Если нежелательно получение и, следовательно, хранение сторонних cookie-файлов, нужно сделать следующее:

1) открыть новое окно Internet Explorer, выбрать в меню Сервис команду Свойства обозревателя и перейти на вкладку Конфиденциальность;
2) оставить ползунок уровня конфиденциальности в положении Умеренно высокий и щелкнуть по кнопке Дополнительно. Откроется окно Дополнительные параметры конфиденциальности (рис. 7.40);
3) установить флажок Перекрыть автоматическую обработку cookie-файлов и установить те параметры приема cookie-файлов, которые показаны на рис. 7.40;
4) 2 раза щелкнуть по кнопке OK для возврата в окно Свойства обозревателя, а затем его закрытия.

При работе пользователя с безопасным веб-подключением, реализуемым с помощью протокола SSL (Security Sockets Layer — слой защищенных сокетов), например при работе со своей учетной записью в виртуальном магазине или банке, происходит шифрование данных,

пересылаемых с веб-сервера на клиентскую машину. После получения данных браузер клиентской машины с помощью специального ключа дешифрует информацию и отображает ее на компьютере. Расшифрованный файл остается в каталоге Temporary Internet Files. Следовательно, он доступен всем, кто имеет возможность локально зарегистрироваться на компьютере.

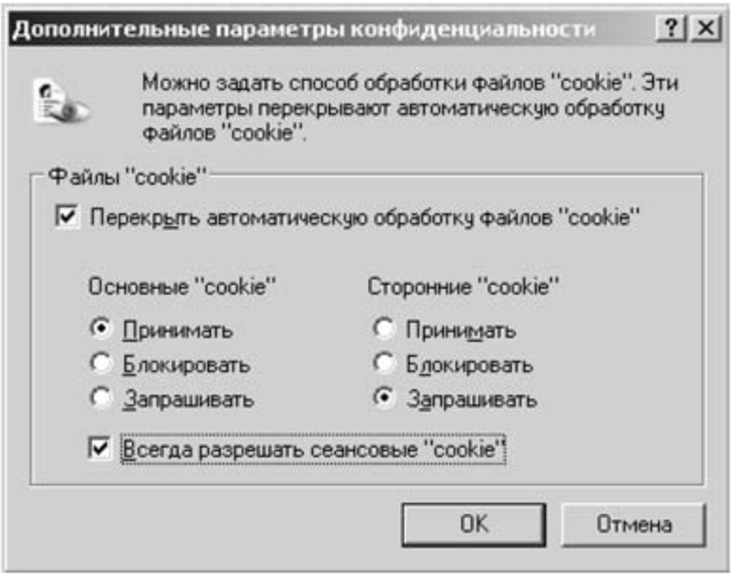


Рис. 7.40

Проблема решается средствами Internet Explorer следующим образом:

1) открыть новое окно Internet Explorer, выбрать в меню Сервис команду Свойства обозревателя и перейти на вкладку Дополнительно;
2) найти в списке группу флажков Безопасность. Установить флажок Не сохранять зашифрованные страницы на диске, как показано на рис. 7.41;
3) щелкнуть по кнопке OK для сохранения и активизации сделанных изменений.

Как отмечалось ранее, функция автоматического заполнения URL-адресов, вводимых в адресной строке браузера, снижает уровень конфиденциальности, поэтому целесообразно очистить файл с историей вводившихся ранее адресов. Однако это не единственная ситуация, в которой срабатывает функция автоматического заполнения.

Система выдает список возможных вариантов и в том случае, когда заполняются поля ввода на веб-страницах. Эта особенность позволяет любому пользователю компьютера видеть, что искали другие пользователи на данном сайте, даже если журнал посещений сайтов в браузере был очищен. Следовательно, функция автоматического заполнения

представляет угрозу конфиденциальности информации. Эту проблему можно решить следующим образом:

1) открыть новое окно Internet Explorer, выбрать в меню Сервис команду Свойства обозревателя и перейти на вкладку Содержание;
2) щелкнуть по кнопке Автозаполнение, как показано на рис. 7.42;
3) после открытия окна Настройки автозаполнения сбросить все флажки в группе Использовать автозаполнение для. Это позволит решить проблему, связанную с функцией автоматического заполнения;
4) в окне Настройка автозаполнения можно для удаления всех данных, хранящихся в журналах автоматического заполнения (рис. 7.43), щелкнуть по двум кнопкам;
5) щелкнуть по кнопке OK для сохранения и активизации сделанных изменений.

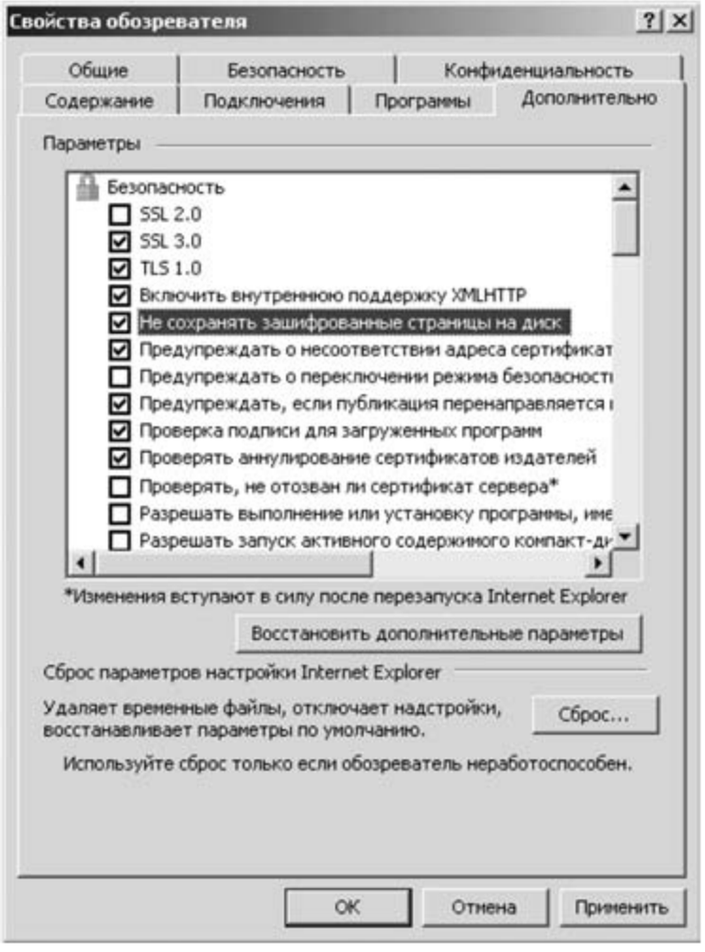


Рис. 7.41

Ранее говорилось об удалении временных файлов Интернета. Удобно, если это делается автоматически при закрытии обозревателя. Для

этого на вкладке Дополнительно окна Свойства обозревателя нужно установить флажок Удалять все файлы из папки временных файлов Интернета при закрытии обозревателя, как показано на рис. 7.44.

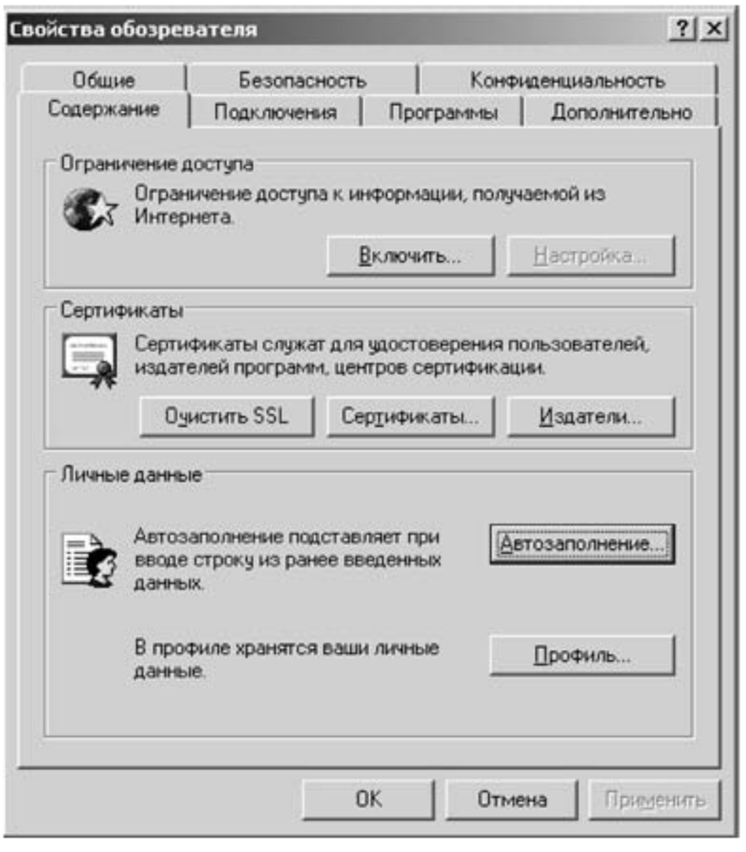


Рис. 7.42

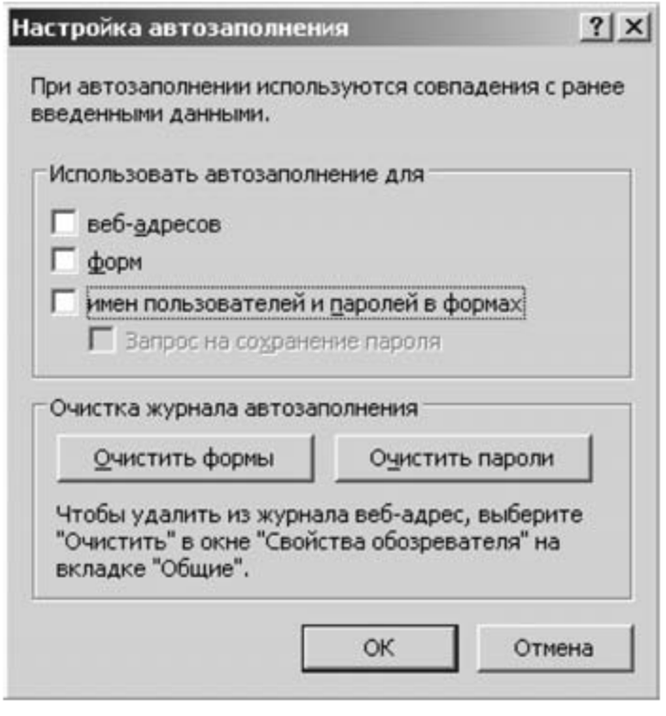


Рис. 7.43

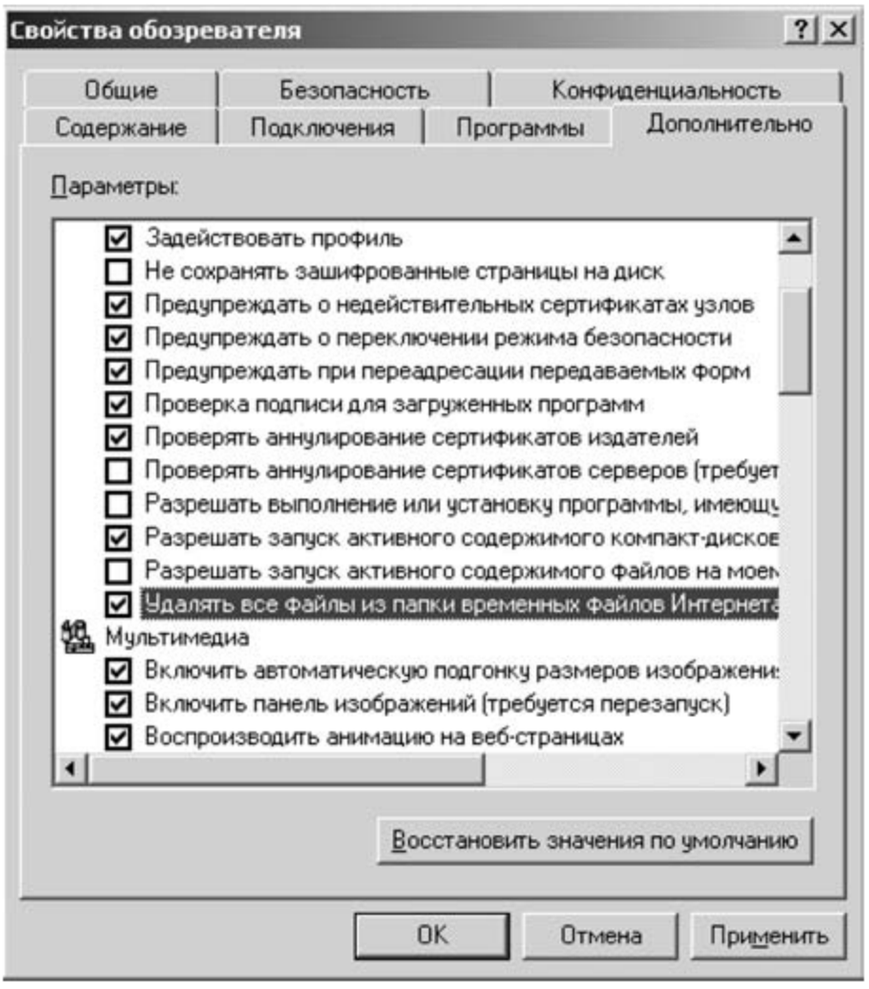


Рис. 7.44

7.5.2. Интерфейс Windows

Проводник Windows сохраняет информацию о запускаемых пользователем приложениях и открываемых файлах. Это сделано для удобства работы пользователя, так как ускоряет его работу, однако и отрицательно сказывается на уровне конфиденциальности, поскольку любой пользователь компьютера может увидеть, с какими программами чаще работает другой пользователь.

Если в целях сохранения конфиденциальности требуется очистить список часто запускаемых приложений, нужно выполнить следующие шаги:

1) щелкнуть правой кнопкой мыши по кнопке Пуск и выбрать в контекстном меню команду Свойства;
2) на вкладке Меню «Пуск» открывающегося окна щелкнуть по кнопке Настроить;
3) щелкнуть по кнопке Очистить список, как показано на рис. 7.45;

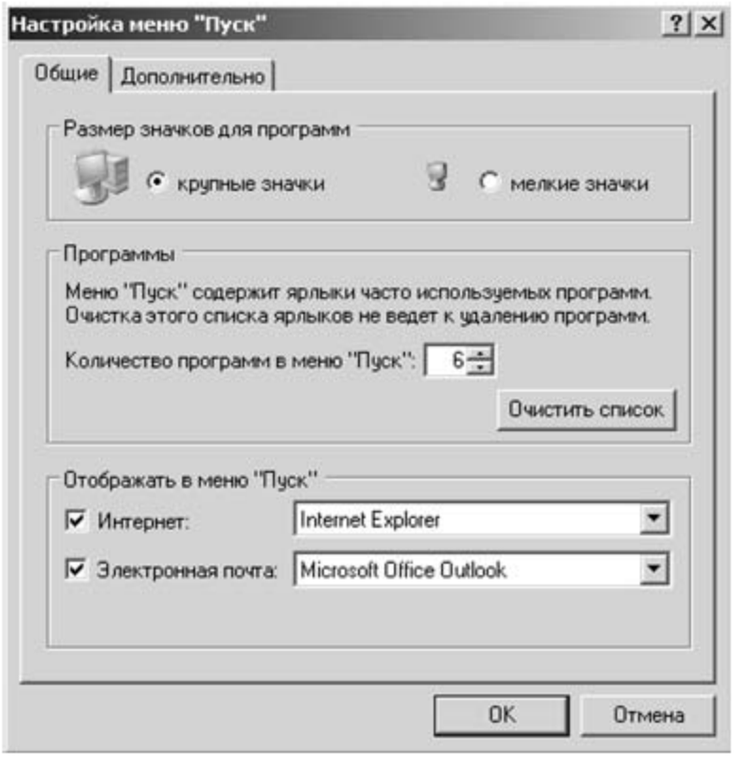


Рис. 7.45

4) щелкнуть по кнопке OK для закрытия окна Настройка меню «Пуск»;
5) еще раз щелкнуть по кнопке OK для закрытия окна Свойства панели задач и меню «Пуск».

Система сохраняет информацию обо всех файлах, которые открывает пользователь. Это позволяет поддерживать список нескольких последних открывающихся файлов любого типа. Для сохранения в секрете перечня документов, с которыми работает пользователь, целесообразно периодически очищать список последних открывавшихся файлов. Сделать это можно следующим образом:

1) щелкнуть правой кнопкой мыши по кнопке Пуск и выбрать в контекстном меню команду Свойства;
2) на вкладке Меню «Пуск» открывающегося окна щелкнуть по кнопке Настроить;
3) после открытия окна Настройка меню «Пуск» перейти на вкладку Дополнительно;
4) щелкнуть по кнопке Очистка списка, как показано на рис. 7.46;
5) щелкнуть по кнопке OK для закрытия окна Настройка меню «Пуск»;
6) еще раз щелкнуть по кнопке OK для закрытия окна Свойства панели задач и меню «Пуск».

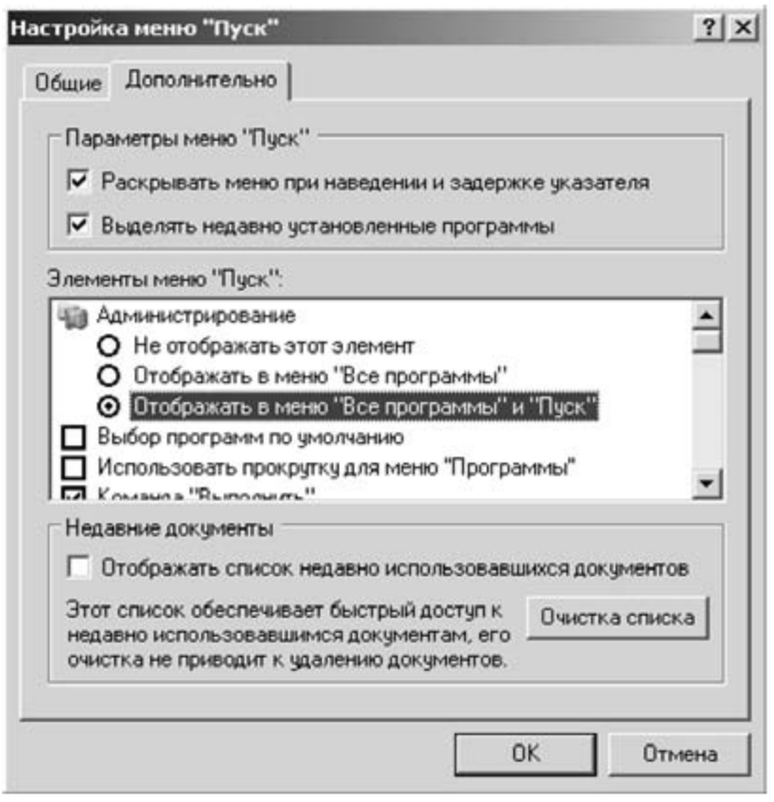


Рис. 7.46

Со временем жесткий диск может заполниться временными файлами, которые создают ОС и некоторые приложения. Эти файлы могут быть использованы для анализа действий пользователя и, кроме того, занимают дисковую память. Поэтому следует периодически очищать диск от таких файлов. В системе имеется служебная программа Очистка диска. Для ее запуска следует выполнить команды Пуск — Программы — Стандартные — Служебные — Очистка диска. После запуска программы откроется окно (рис. 7.47), в котором надо выбрать имя диска, на котором требуется удалить временные файлы, после чего нажать кнопку OK. Через некоторое время открывается окно с перечнем файлов, которые можно удалить (рис. 7.48).

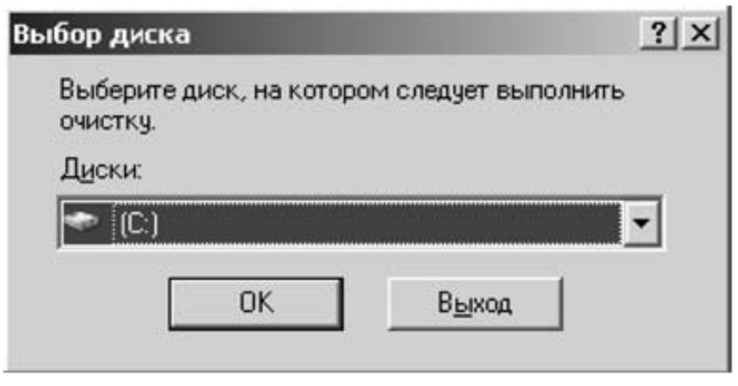


Рис. 7.47

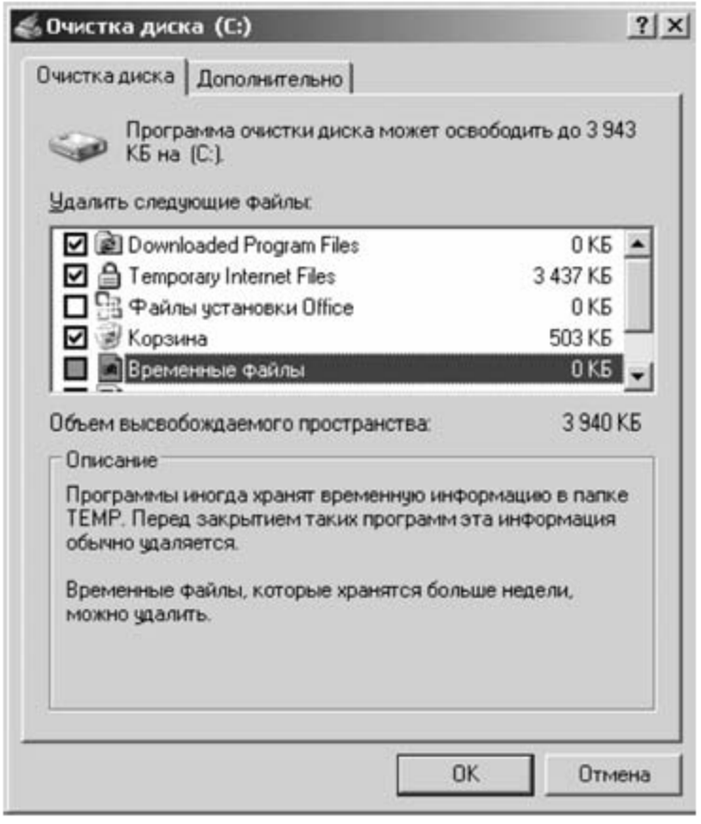


Рис. 7.48

Существуют программы сторонних производителей, которые автоматически находят каталоги с временными файлами и очищают их. Популярную утилиту TempCleaner можно загрузить со многих веб-сайтов (рис. 7.49).

Рис. 7.49

При посещении веб-сайтов, требующих аутентификации, или подключении к удаленным компьютерам пользователю часто предлагается сохранить пароль, чтобы при следующем доступе к сайту не приходилось вводить пароль заново. Такой режим удобен для пользователя, но создает дополнительную уязвимость в системе безопасности, поскольку любой другой пользователь, имеющий доступ к этому же

компьютеру, сможет воспользоваться чужим именем и паролем, даже если он их не знает.

Удаление сохраненных паролей с компьютера позволит защитить свои учетные записи и повысить уровень их конфиденциальности. В работе [13] предлагается следующий способ доступа к списку паролей:

1) в главном меню выбрать команду Выполнить;
2) в поле ввода открывшегося окна набрать строку rundll32.exe keymgr.dll,KRShowKeyMgr и щелкнуть на кнопке OK;
3) откроется окно Сохранение имен пользователей и паролей со списком всех учетных записей, сохраненных на компьютере (рис. 7.50);

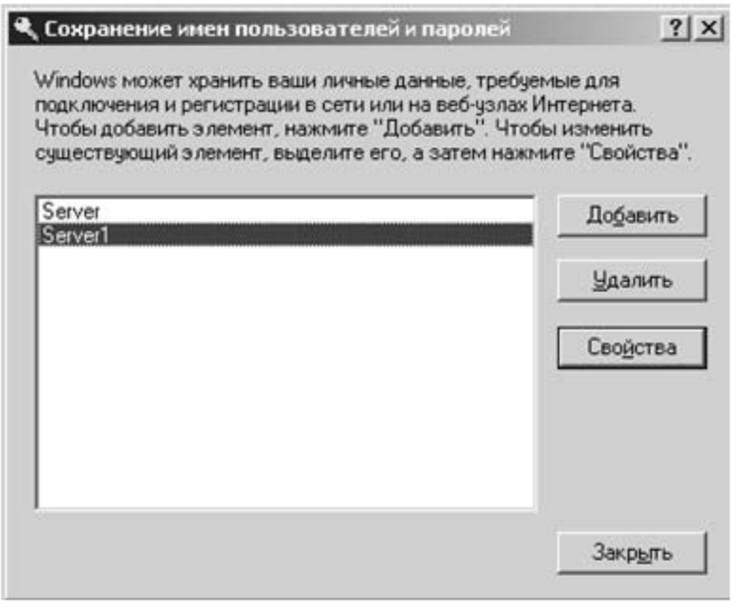


Рис. 7.50

4) для удаления сохраненного пароля выбрать в списке нужную учетную запись и щелкнуть по кнопке Удалить;
5) щелкнуть по кнопке OK в диалоговом окне подтверждения, и учетная запись будет удалена из списка;
6) повторить предыдущие шаги для всех учетных записей, которые нужно удалить;
7) закончив удаление, щелкнуть на кнопке Закрыть.

На компьютерах с ОС Windows XP/2000/2003, использующих файловую систему NTFS, можно устанавливать разрешения на доступ к файлам и папкам. Это может быть очень мощным инструментом в обеспечении конфиденциальности информации.

Чтобы установить полный контроль над разрешениями на доступ, нужно сначала отключить режим общего доступа к файлам. Для этого

следует открыть любую папку, в меню Сервис выбрать команду Свойства папки, перейти на вкладку Вид и сбросить флажок Использовать простой общий доступ к файлам. Далее можно перейти к настройке разрешений на доступ к требуемым папкам и файлам, как это рассмотрено в подразделе 4.4. Кроме того, для защиты данных в файловой системе NTFS можно их зашифровывать (см. подраздел 4.5).

Задания для самостоятельной работы

1. В целях конфиденциальности вашей информации провести очистку четырех частей данных браузера: списка вводившихся адресов, журнала с историей посещения веб-сайтов, списка временных файлов Интернета и списка cokie-файлов.
2. Изменить интерфейс Windows компьютера в целях повышения конфиденциальности вашей работы на компьютере:
   - очистить список часто запускавшихся приложений;
   - очистить список последних открывавшихся документов;
   - удалить временные файлы с жесткого диска;
   - удалить сохраненные пароли;
   - назначить необходимые разрешения к файлам и папкам;
   - зашифровать важную для вас информацию.

ГЛАВА 8

СЕТЕВЫЕ ВОЗМОЖНОСТИ ОПЕРАЦИОННЫХ СИСТЕМ

8.1. Диагностика сетевых подключений в Windows XP

Пользователи Windows XP для диагностики сетевого подключения могут воспользоваться специальным мастером. Эта программа вызывается из меню задачи Сведения о системе (Пуск > Все программы > Стандартные > Служебные > Сведения о системе > меню Сервис > Диагностика сети). Для получения информации об оборудовании, программном обеспечении и сетевых подключениях нужно нажать кнопку Собрать информацию (рис. 8.1).

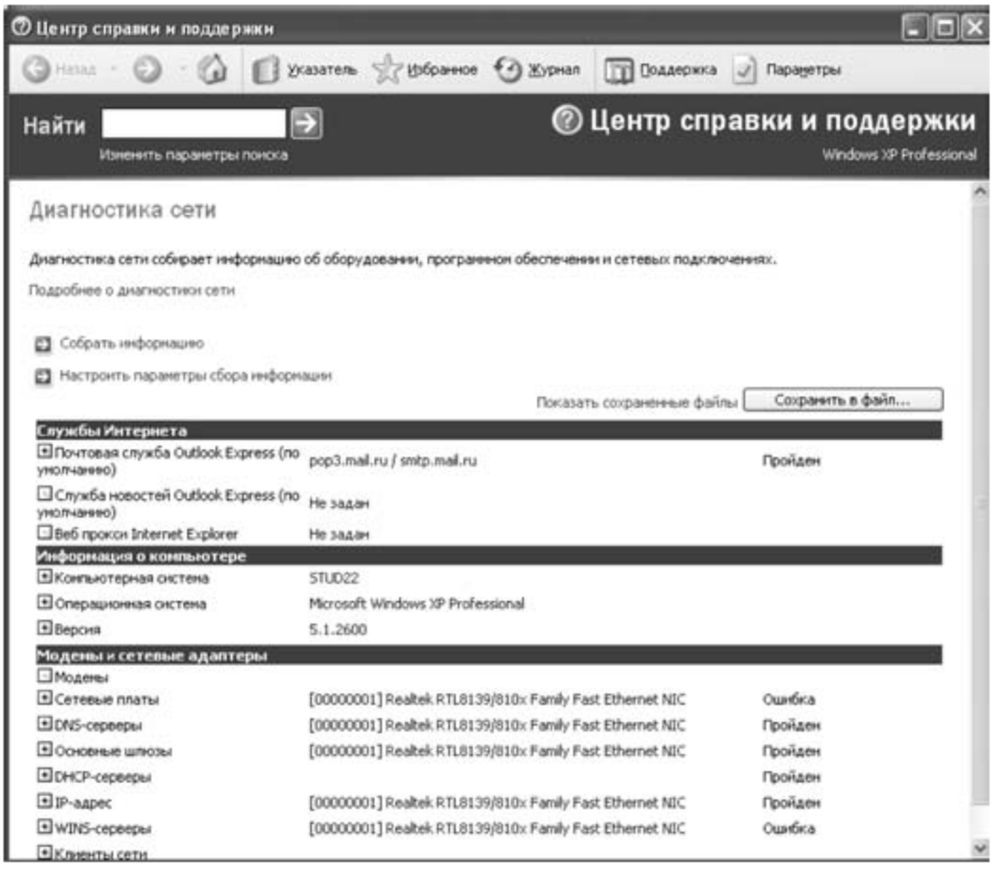


Рис. 8.1

Кроме указанного мастера, ОС в своем составе имеет группу программ для решения проблем при работе с сетью. К ним относятся следующие команды:

Утилита	Назначение
ping	Проверка работоспособности сети
ipconfig	Управление конфигурацией IP и параметрами службы имен доменов (DNS)
tracert	Отслеживание пути, проходимого пакетами данных между локальной станцией и указанным адресом, и временной задержки
telnet	Подключение к удаленному компьютеру
netstat	Информация о маршрутизации, протоколах, портах, активных подключениях локальной станции
nbtstat	Отображение NetBIOS — имен компьютеров в IP-адреса
arp	Отображает и модифицирует IP- и физические адреса по протоколу arp
ftp	Обмен файлами с удаленным компьютером
finger	Получение информации об учетной записи пользователя
hostname	Определение имени локального хоста
nslookup	Диагностика и устранение неисправностей в DNS

Рассмотрим некоторые из них более подробно.

Утилита ping служит для проверки работоспособности сети:

ping -s [-drvRlfnq] [-i время_ожидания] [-p шаблон] хост [размер_данных [кол_пакетов]]

Команда ping запускается из командной строки и использует датаграмму протокола ICMP, чтобы вызвать ответ указанного хоста или сетевого шлюза. Если хост отвечает, ping выдает сообщение, что хост жив (хост is alive), в стандартный выходной поток и завершает работу. В противном случае после таймаута секунд она выдает сообщение, что от хоста ответа нет (no answer from хост). Стандартное значение таймаута — 20 с.

Команда ping воспринимает следующие опции:

-d — режим отладки. Поставщику передается опция SO_DEBUG;

-f — лавинный ping. Выдает пакеты сразу после возвращения или 100 раз в 1 с, в зависимости от того, что быстрее. Для каждого посланного ECHO_REQUEST печатается точка «.», а для каждого полученного ECHO_REPLY печатается забой (backspace). Это позволяет быстро

оценить, сколько пакетов потеряно. Только привилегированный пользователь может использовать эту опцию. Эта команда может существенно увеличить загрузку сети, и ее надо использовать осторожно;

-i — ожидать время_ ожидания секунд между посылками пакетов. По умолчанию интервал между посылками пакетов 1 с. Эта опция несовместима с опцией -f;

-l — исключить маршрут к источнику. Использует эту опцию в заголовке IP для посылки пакета указанному хосту и обратно. Обычно указывается с опцией -R. Опция -l допустима только когда в качестве хоста указан localhost или `uname -n`;

-n — только числовая выдача. Не предпринимать попытки искать символьные имена для адресов хостов;

-p — заполнение посылаемых пакетов. Шаблон задается, как шестнадцатиричная строка байтов, и может иметь длину до 16 байтов. Шаблон повторяется для заполнения раздела данных пакета. Например, -p ff вызывает заполнение пакетов единицами. Эта опция полезна при поиске проблем сети, связанных с передаваемыми данными;

-q — сокращенный вывод. Не выдается ничего, кроме суммарных строк при запуске и завершении работы;

-r — не использовать обычные таблицы маршрутизации и посылать напрямую указанному хосту в подключенной сети. Если хост не находится в непосредственно подключенной сети, возвращается ошибка. Эту опцию можно использовать для обращения к локальному хосту через интерфейс, удаленный демоном маршрутизации (см. routed(1M));

-R — записать маршрут. Устанавливает опцию записи маршрута IP, в результате чего маршрут пакета будет записан в заголовке IP. Содержимое записи маршрута будет выдано, только если указана опция -v, и будет устанавливаться для возвращаемых пакетов, только если целевой хост сохраняет запись маршрута между выдачами или если задана опция -l;

-s — посылать датаграмму каждую секунду и печатать строку вывода для каждого полученного ответа ECHO_RESPONSE (если ответа нет, ничего не выдается);

-v — детальный вывод. Выдает все полученные пакеты ICMP, кроме ECHO_RESPONSE.

Когда указан флаг -s, ping посылает датаграмму каждую секунду и печатает одну строку вывода для каждого полученного ответа. В этом случае ping вычисляет *времена обхода* (round trip times) и статистику потери пакетов; после завершения или по истечении таймаута команда печатает соответствующую итоговую информацию. Если указано необязательное количество пакетов (кол_пакетов), ping посылает только соответствующее количество запросов. Если количество пакетов

не указано, команда будет выполняться бесконечно. Для прекращения продолжающегося вывода используется клавиша прерывания (Del).

Стандартный размер пакета датаграммы равен 64 байтам, но можно задать и другой размер с помощью аргумента командной строки размер_данных. Поскольку ping автоматически добавляет 8-байтовый заголовок к каждой посылаемой датаграмме, размер пакета, показываемый при использовании опции -s с аргументом размер_данных, всегда будет на 8 байтов больше, чем указанное вами значение.

При использовании ping для поиска сбоев в сети необходимо сначала выполнить ping на локальный хост, чтобы убедиться, что работает локальный сетевой интерфейс.

Проверка правильности установки протокола TCP/IP. Откройте командную строку и выполните команду ping 127.0.0.1

Адрес 127.0.0.1 — это личный адрес любого компьютера. Таким образом, эта команда проверяет прохождение сигнала «на самого себя». Она может быть выполнена без наличия какого-либо сетевого подключения. Вы должны увидеть строки, подобные, изображенным на рис. 8.2:

```
E:\>ping 127.0.0.1

Обмен пакетами с 127.0.0.1 по 32 байт:

Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128

Статистика Ping для 127.0.0.1:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 0 мсек, Среднее = 0 мсек

E:\>_
```

Рис. 8.2

По умолчанию команда посылает пакет 32 байта. Размер пакета может быть увеличен до 65 Кбайт. Так можно обнаружить ошибки при пересылке пакетов больших размеров. За размером тестового пакета отображается время отклика удаленной системы (в нашем случае — меньше 1 мс).

Затем следует еще один параметр протокола — значение TTL — «время жизни» пакета. На практике это число маршрутизаторов, через которые может пройти пакет. Каждый маршрутизатор уменьшает значение TTL на единицу. При достижении нулевого значения пакет уничтожается. Такой механизм введен для исключения случаев зацикливания пакетов.

На рисунке 8.3 изображена стандартная справка ОС:

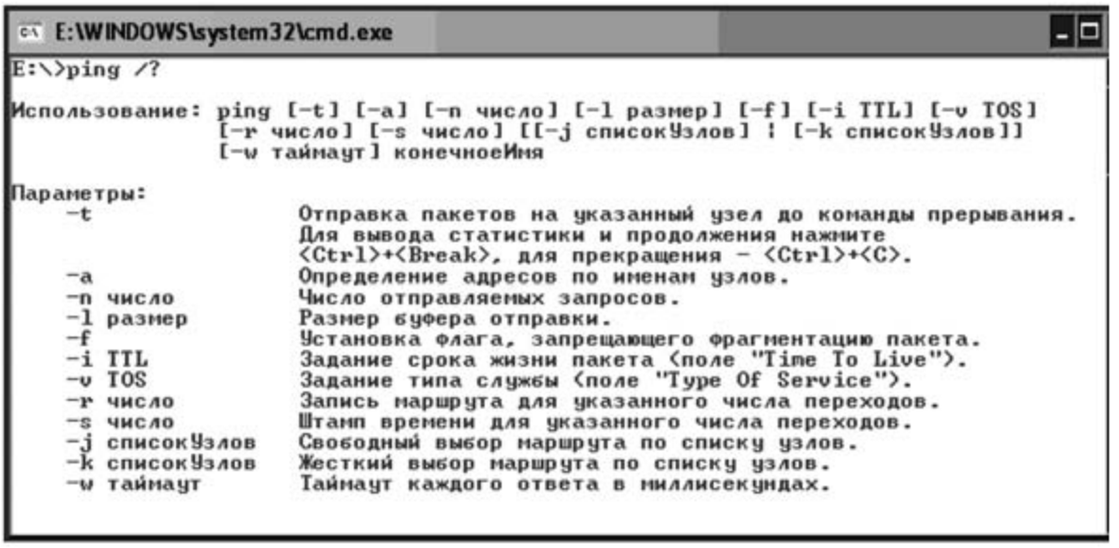


Рис. 8.3

Если будет показано сообщение о недостижимости адресата, то это означает ошибку установки протокола IP. В этом случае целесообразно удалить протокол из системы, перезагрузить компьютер и вновь установить поддержку протокола TCP/IP.

Проверка видимости локального компьютера и ближайшего компьютера сети. Выполните команду ping 192.168.1.3

На экран должны быть выведены строки примерно как на рис. 8.4:

```
E:\>ping 192.168.1.3

Обмен пакетами с 192.168.1.3 по 32 байт:

Ответ от 192.168.1.3: число байт=32 время<1мс TTL=128
Ответ от 192.168.1.3: число байт=32 время=3мс TTL=128
Ответ от 192.168.1.3: число байт=32 время=2мс TTL=128
Ответ от 192.168.1.3: число байт=32 время<1мс TTL=128

Статистика Ping для 192.168.1.3:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 3 мсек, Среднее = 1 мсек

E:\>_
```

Рис. 8.4

Наличие отклика свидетельствует о том, что канал связи установлен и работает.

Проверка работоспособности сервера имен Интернета. Выполните команду ping www.Moscow.ru. Если система сможет различить IP-адрес этого хоста, то система распознавания имен работоспособна. На экран должен быть выведен листинг, аналогичный рис. 8.5:

```
E:\>ping www.Moscow.ru

Обмен пакетами с Moscow.ru [194.87.48.19] по 32 байт:

Ответ от 194.87.48.19: число байт=32 время=9мс TTL=55
Ответ от 194.87.48.19: число байт=32 время=5мс TTL=55
Ответ от 194.87.48.19: число байт=32 время=5мс TTL=55
Ответ от 194.87.48.19: число байт=32 время=4мс TTL=55

Статистика Ping для 194.87.48.19:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
Приблизительное время приема-передачи в мс:
    Минимальное = 4мсек, Максимальное = 9 мсек, Среднее = 5 мсек

E:\>_
```

Рис. 8.5

Если не будет ответа на ввод команды с именем существующего хоста, то это может свидетельствовать либо об ошибке в задании DNS-серверов, либо о их неработоспособности.

Утилита ipconfig предназначена для отображения параметров IP-протокола в Windows NT/2000/XP (в Windows 9x для этого используется *winipcfg*). Эта утилита выводит на экран основные параметры настройки протокола TCP/IP: значения адреса, маски, шлюза.

Нажмите кнопку Пуск, выберите строку меню Выполнить, наберите символы cmd и нажмите клавишу Enter на клавиатуре. В открывшемся окне наберите ipconfig /all. При нормальной работе компьютера на экран должен вывестись листинг, аналогичный рис. 8.6:

```
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

W:\>ipconfig/all

Настройка протокола IP для Windows

        Имя компьютера  . . . . . . . . . : stud22
        Основной DNS-суффикс  . . . . . . : nipp.edu
        Тип узла. . . . . . . . . . . . . : гибридный
        IP-маршрутизация включена . . . . : нет
        WINS-прокси включен . . . . . . . : нет
        Порядок просмотра суффиксов DNS . : nipp.edu
                                             nipp.edu

MIPP - Ethernet адаптер:

        DNS-суффикс этого подключения . . : nipp.edu
        Описание  . . . . . . . . . . . . : Realtek RTL8139/810x Family Fast Eth
ernet NIC
        Физический адрес. . . . . . . . . : 00-17-31-CD-BA-B0
        Dhcp включен. . . . . . . . . . . : да
        Автонастройка включена  . . . . . : да
        IP-адрес  . . . . . . . . . . . . : 192.168.1.22
        Маска подсети . . . . . . . . . . : 255.255.255.0
        Основной шлюз . . . . . . . . . . : 192.168.1.4
        DHCP-сервер . . . . . . . . . . . : 192.168.1.1
        DNS-серверы . . . . . . . . . . . : 192.168.1.1
                                            81.25.51.124
        Основной WINS-сервер  . . . . . . : 192.168.1.1
        Дополнительный WINS-сервер. . . . : 192.168.1.2
        Аренда получена . . . . . . . . . : 4 июня 2008 г. 8:47:15
        Аренда истекает . . . . . . . . . : 12 июня 2008 г. 8:47:15

W:\>
```

Рис. 8.6

Отключите сетевое подключение, повторите команду. При отсутствующем соединении на экран выводится листинг, изображенный на рис. 8.7:

```
E:\>ipconfig/all

Настройка протокола IP для Windows

        Имя компьютера  . . . . . . . . . : stud22
        Основной DNS суффикс  . . . . . . : nipp.edu
        Тип узла. . . . . . . . . . . . . : гибридный
        IP-маршрутизация включена . . . . : нет
        WINS-прокси включен . . . . . . . : нет

NIPP - Ethernet адаптер:

        Состояние сети  . . . . . . . . . : сеть отключена
        Описание  . . . . . . . . . . . . : Realtek RTL8139/810x Family Fast Eth
ernet NIC
        Физический адрес. . . . . . . . . : 00-17-31-CD-BA-B0

E:\>_
```

Рис. 8.7

Обратите внимание, что программа вывела на экран только данные о «физических» параметрах сетевой карты и указала, что отсутствует подключение сетевого кабеля.

Утилита tracert используется для показа пути прохождения сигнала до желаемого хоста. Зачастую это позволяет выяснить причины плохой работоспособности канала. Точка, после которой время отклика резко увеличено, свидетельствует о наличии в этом месте узкого места, не справляющегося с нагрузкой. Стандартная справка этой утилиты содержит расшифровку параметров для ее настройки (рис. 8.8).

```
C:\>tracert/?

Использование: tracert [-d] [-h максЧисло] [-j списокУзлов] [-w таймаут]
                       [-R] [-S адресИсточника] [-4] [-6] конечноеИмя

Параметры:
    -d                  Без разрешения в имена узлов.
    -h максЧисло        Максимальное число прыжков при поиске узла.
    -j списокУзлов      Свободный выбор маршрута по списку узлов (только IPv4).
    -w таймаут          Таймаут каждого ответа в миллисекундах.
    -R                  Трассировка пути (только IPv6).
    -S адресИсточника   Используемый адрес источника (только IPv6).
    -4                  Принудительное использование IPv4.
    -6                  Принудительное использование IPv6.

C:\>_
```

Рис. 8.8

В командной строке введите команду tracert www.lenta.ru и увидите листинг, аналогичный изображенному на рис. 8.9.

```
E:\>tracert www.lenta.ru

Трассировка маршрута к www.lenta.ru [81.19.69.28]
с максимальным числом прыжков 30:

  1    <1 мс    <1 мс    <1 мс  192.168.1.4
  2    <1 мс    <1 мс    <1 мс  mxu.elpag.ru [81.25.51.124]
  3     1 ms    <1 мс    <1 мс  ultra.elpag.ru [192.168.16.1]
  4     1 ms    <1 мс     1 ms  10.7.5.48
  5     1 ms     1 ms     1 ms  border2.ultranet.ru [81.25.53.90]
  6     1 ms     1 ms     1 ms  msk k1 b2 ge0 1 0.fiord.ru [62.140.239.29]
  7     2 ms     2 ms     1 ms  msk-m9-b1-ge0-0-3-vlan2.fiord.ru [62.140.239.180
]
  8     3 ms     3 ms     2 ms  m10-msk-ix.j2.rantel.ru [193.232.246.118]
  9     2 ms     2 ms     2 ms  sl6509-m10.rantel.ru [81.19.64.215]
 10     2 ms     3 ms     3 ms  ge-0-2-cs7206.rantel.ru [81.19.64.70]
 11     3 ms     3 ms     4 ms  lenta12.cust.rantel.ru [81.19.69.28]

Трассировка завершена.

E:\>
```

Рис. 8.9

Утилита Nslookup используется для получения информации от DNS-сервера. Параметры утилиты показаны на рис. 8.10.

```
C:\>nslookup/?
Usage:
   nslookup [-opt ...]             # interactive mode using default server
   nslookup [-opt ...] - server    # interactive mode using 'server'
   nslookup [-opt ...] host        # just look up 'host' using default server
   nslookup [-opt ...] host server # just look up 'host' using 'server'

C:\>
```

Рис. 8.10

По умолчанию (после запуска без указания параметров) осуществляется подключение к указанному в настройках протокола серверу DNS. Набирая необходимые имена в качестве запроса, можно получить информацию о данных DNS по этому имени, найти почтовый сервер, обслуживающий домен, уточнить данные регистрации и т.п.

Для иллюстрации использования утилиты Nslookup необходимо выполнить следующие шаги:

1) запустите команду Nslookup. Возможный результат показан на рис. 8.11.

```
   nslookup [-opt ...]             # interactive mode using default server
   nslookup [-opt ...] - server    # interactive mode using 'server'
   nslookup [-opt ...] host        # just look up 'host' using default server
   nslookup [-opt ...] host server # just look up 'host' using 'server'

C:\>nslookup
*** Default servers are not available
Default Server:  UnKnown
Address:  127.0.0.1

> _
```

Рис. 8.11

При работе в локальной сети учебного класса результат может быть таким, как изображено на рис. 8.12:

```
C:\>nslookup
Default Server:  dc3.staff.corp.local
Address:  192.168.0.13
```

Рис. 8.12

2) наберите server имя_сервера_из__lookup команды и нажмите Enter — этой командой мы указываем, какой DNS-сервер мы хотим использовать для получения интересующих нас данных;
3) наберите set type=all и нажмите Enter — этой командой мы указали, что нас будут интересовать все данные касательно задаваемого нами домена;
4) наберите hse.ru и нажмите Enter — этой командой мы запрашиваем данные по домену hse.ru На экране вы должны получить листинг, аналогичный приведенному на рис. 8.13:

```
Server:  dc3.staff.corp.local
Addresses:  192.168.0.13, 192.168.0.5

*** dc3.staff.corp.local can't find 192.168.0.5: Non-existent domain
> set type=all
> hse.ru
Server:  dc3.staff.corp.local
Addresses:  192.168.0.13, 192.168.0.5

hse.ru  internet address = 192.168.101.94
hse.ru  nameserver = dc12.nnov.corp.local
hse.ru  nameserver = dc5.workgroup
hse.ru  nameserver = dc3.staff.corp.local
hse.ru  nameserver = ds2.corp.local
hse.ru  nameserver = dc4.staff.corp.local
hse.ru  nameserver = dc1.corp.local
hse.ru  nameserver = dc6.study.corp.local
hse.ru  nameserver = dc10.spb.corp.local
hse.ru
        primary name server = dc3.staff.corp.local
        responsible mail addr = hostmaster.corp.local
        serial  = 73489
        refresh = 900 (15 mins)
        retry   = 600 (10 mins)
        expire  = 86400 (1 day)
        default TTL = 3600 (1 hour)
hse.ru  MX preference = 10, mail exchanger = exchange.hse.ru
dc12.nnov.corp.local    internet address = 172.17.2.2
dc3.staff.corp.local    internet address = 192.168.0.5
dc3.staff.corp.local    internet address = 192.168.0.13
ds2.corp.local  internet address = 192.168.0.12
dc4.staff.corp.local    internet address = 192.168.0.9
dc1.corp.local  internet address = 192.168.0.11
dc10.spb.corp.local     internet address = 172.20.20.2
exchange.hse.ru internet address = 192.168.0.61
>
```

Рис. 8.13

8.2. Средства диагностики сетевых протоколов в Unix-подобных ОС

Диагностика TCP/IP-протоколов в Unix-подобных ОС проводится с помощью комплекта программ — утилит, представленных ниже:

Утилита	Назначение
ifconfig	Просмотр и управление параметрами сетевых интерфейсов в UNIX
ping	Проверка связи с IP-хостом
traceroute	Отображает адреса всех маршрутизаторов на пути от клиента до удаленного хоста
route	Позволяет конфигурировать сетевые маршрутные таблицы
netstat	Отображает статистику и текущие соединения по протоколу TCP/IP.
arp(Address Resolution Protocol)	Протокол преобразования адресов. Отображает IP-адреса в адреса Ethernet
nbstat	Выводит статистику и текущие соединения по протоколу NetBIOS поверх TCP/IP
ftp (File Transfer Protocol)	Протокол передачи файлов. Позволяет выводить список файлов, осуществлять передачу файлов, а также управлять каталогами на удаленной системе
telnet	Виртуальный Сетевой Терминал, удаленный доступ к сетевым ресурсам
tcpdump	«сниффер»-программа, предназначенная для перехвата сетевого трафика

Утилита ifconfig является основной утилитой для просмотра и управления параметрами сетевых интерфейсов в UNIX. Утилита ifconfig, вызванная в командной строке без параметров, выводит сведения обо всех настроенных в данный момент сетевых интерфейсах (eth0, eth1, lo, и др.). Чтобы просмотреть информацию о конкретном интерфейсе, необходимо указать его имя в качестве параметра ifconfig.

Пример вывода информации при обращении к утилите без параметров:

```
eth0  Link encap:Ethernet HWaddr 00:04:75:c1:e2:ab
      inet addr:10.2.10.32 Bcast:10.2.10.255 Mask:255.255.255.0
      ...
```

```
eth1  Link encap:Ethernet HWaddr 00:04:75:c1:e2:6b
      inet addr:192.168.140.1 Bcast:192.168.140.255 Mask:255.255.255.0
      ...
      ...

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      ...
      ...
```

Утилита ping (Packet Internet Groper) служит для принудительного вызова ответа конкретной машины. Она позволяет проверять наличие связи с указанным ip-адресом, время прохождения пакета, правильность работы системы распознавания символьных имен. Запросы утилиты ping передаются по протоколу ICMP (Internet Control Message Protocol). Получив такой запрос, программное обеспечение, реализующее протокол IP у адресата, немедленно посылает эхо-ответ. Эхо-запросы посылаются заданное количество раз (ключ -n) или по умолчанию до тех пор, пока пользователь не введет команду прерывания (Ctrl+C или Del), после чего выводятся статистические данные. Например:

```
ubuntu@ubuntu:~$ ping 127.0.0.1

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=39 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=40 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=41 ttl=64 time=0.050 ms
64 bytes from 127.0.0.1: icmp_seq=42 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=43 ttl=64 time=0.050 ms
64 bytes from 127.0.0.1: icmp_seq=44 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=45 ttl=64 time=0.048 ms
^C
--- 127.0.0.1 ping statistics ---
45 packets transmitted, 45 received, 0% packet loss, time 43996ms
rtt min/avg/max/mdev = 0.043/0.049/0.054/0.002 ms
```

Формат команды для обращения к утилите имеет следующий вид:

```
ping [-t][-a][-n][-l][-f][-i TTL][-v TOS] [-r][][имя машины][[-j списокУзлов] [-k списокУзлов]][-w]
```

Параметры утилиты ping:

-t — отправка пакетов на указанный узел до команды прерывания;

-a — определение адресов по именам узлов;

-n — число отправляемых запросов;

-l — размер буфера отправки;

-f — установка флага, запрещающего фрагментацию пакета;

-i TTL — задание времени жизни пакета (поле «Time To Live»);

-v TOS — задание типа службы (поле «Type Of Service»);

-r — запись маршрута для указанного числа переходов;

-s — штамп времени для указанного числа переходов;

-j список узлов — свободный выбор маршрута по списку узлов;

-k список узлов — жесткий выбор маршрута по списку узлов;

-w интервал — интервал ожидания каждого ответа в миллисекундах.

Большинство опций в формате команды обычно не используется. Формат команды в этом случае сокращается: ping имя_узла.

Утилита traceroute. Программа traceroute позволяет выявлять последовательность узлов, через которые проходит IP-пакет на пути к пункту своего назначения.

Формат команды для обращения к утилите:

tracert [-d][-h][-j][-w] IP-address | DNS-имя удаленного хоста.

Параметры утилиты traceroute:

-d — отключение режима определения dns-имен хостов по IP-адресам;

-h — количество ретрансляций;

-j — список_систем;

-w — тайм-аут;

IP-address — цифровой или символьный IP-address удаленного хоста;

DNS-имя — цифровое или символьное DNS-имя удаленного хоста.

Выходная информация представляет собой список пройденных машин с указанием времени прохождения каждого узла. Например:

1 <10 ms 10 ms <10 ms 205.188.247.65
2 <10 ms 20 ms <10 ms 206.23.252.6
3 30 ms 30 ms 20 ms 206.23.252.7
...
12 110 ms 100 ms 90 ms 207.68.145.59

13 100 ms 90 ms 100 ms 207.68.129.34
14 121 ms 130 ms 140 ms 131.107.34.133
Trace complete.

Многие из параметров обычно не используются, в связи с чем формат команды упрощается: traceoute имя_машины

Утилита netstat выводит информацию о сети и средствах TCP/IP. Содержание и форма выходной информации зависят от состава указанных в команде параметров. Обычно выводятся следующие данные:

1) список соединений;
2) статистика сетевых интерфейсов;
3) информация по буферам данных;
4) содержание таблицы маршрутизации;
5) статистика работы протокола.

Формат команды для обращения к утилите:

netstat [-a][-e][-n][-s] [-r] [-i] [-p имя][-r][интервал]

Параметры команды:
-r — вывод таблицы маршрутизации;
-i — вывод статистики сетевых интерфейсов;
-s — вывод статистики передачи данных (по протоколу SNMP);
-n — имена портов в числовом виде;
-a — вывод состояния всех портов;
-e — отображение статистики Ethernet. Этот ключ может применяться вместе с ключом -s;
-p — отображение подключений для протокола «имя»: tcp или udp.

Каждое соединение машины с сетью называется сетевым интерфейсом. Машина, имеющая более одного интерфейса, может принимать данные по одному интерфейсу и передавать их по другому, таким образом осуществляя пересылку данных между сетями. Эта функция называется маршрутизацией, а машина, выполняющая ее, — шлюзом.

Команда netstat -r позволяет отображать таблицу маршрутизации. Пункты назначения и шлюзы могут показываться или именами машин, или их IP-адресами. Флаги дают оценку маршрута.

При использовании ключа -i команды netstat на экран будут выведены статистические данные всех используемых интерфейсов. Исходя из них можно выяснить, исправно ли соединение с сетью.

Утилита arp предназначена для просмотра и модификации ARP-кэша. Адреса можно вводить в ARP-кэш вручную при помощи утилиты ARP.exe.

Формат команды для обращения к утилите:

arp-param IP-адрес MAC-адрес.

Утилита управляется с помощью параметров (param), которые могут принимать следующие значения:

-a — вывод записей, содержащихся в ARP-кэше;

-q — вывод записей, содержащихся в ARP-кэше (этот ключ недоступен в Windows for Workgroups);

-N — вывод записей, содержащихся в ARP-кэше и относящихся к определенному сетевому интерфейсу;

-s — добавление статической записи в ARP-кэш. Синтаксис: ARP -s IP-адрес MAC-адрес;

-d — удаление статической записи из ARP-кэша.

Утилита TCPdump относится к числу так называемых «снифферов» — программ, предназначенных для перехвата сетевого трафика.

Формат команды для обращения к утилите:

```
tcpdump [ -AdDefIKlLnNOpqRStuUvxX ] [ -B buffer_size ] [ -c count ]
[ -C file_size ] [ -G rotate_seconds ] [ -F file ]
[ -i interface ] [ -m module ] [ -M secret ]
[ -r file ] [ -s snaplen ] [ -T type ] [ -w file ]
[ -W filecount ]
[ -E spi@ipaddr algo:secret,... ]
[ -y datalinktype ] [ -z postrotate-command ] [ -Z user ]
[ expression ]
```

Опции **tcpdump** можно подразделить на несколько типов:

1) выбор объекта (какой интерфейс прослушивать, читать ли данные из файла, сохранять ли их в файл);
2) опции форматирования (в каком виде представить дату, выводить ли шестнадцатеричный дамп пакета и т.п.);
3) прочие опции (объем захватываемой информации, привилегии, буферизация, запись в файлы в стиле logrotate и вращение записанных файлов);
4) условия (т.е. какие пакеты перехватывать), например, можно перехватывать только ARP-запросы или только почтовый трафик;

5) опция -i interface определяет, какой интерфейс должен прослушиваться программой;
6) опция -w file записывает всю информацию в файл в бинарном виде, т.е. делает dump того, что происходит на сетевом интерфейсе для последующего анализа, а не выводит всю информацию на экран в режиме реального времени.

Рассмотрим некоторые операции по работе с сетью.

Настройка модемного соединения с Интернетом (dialup). На рабочем столе находим иконку Начать здесь, от нее переходим к Системные параметры, а затем активируем иконку Сеть (рис. 8.14).

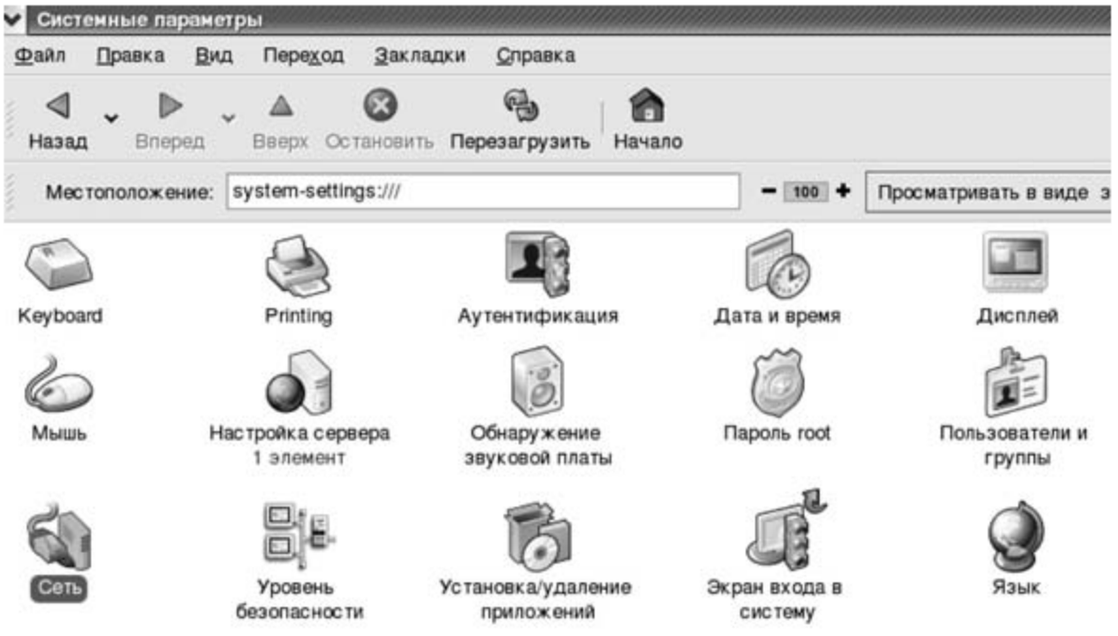


Рис. 8.14

На экране появляется окно (рис. 8.15) настройки сети (в этом окне вначале нет второй строки — модемного соединения).

В этом окне выбираем опцию Создать. Появляется окно Новое соединение, в котором выбирается опция Модемное соединение (рис. 8.16).

Начинается настройка удаленного доступа через модем. Для выполнения этой настройки на экране появляется окно, показанное на рис. 8.17, содержащее шесть режимов (вкладок) настройки.

На вкладке Общие:

- задается псевдоним, под которым это соединение будет фигурировать в окне Настройка сети;
- определяется:
  — нужно ли активировать устройство при загрузке ОС ЭВМ;

— все ли пользователи могут активировать и дезактивировать данное соединение.

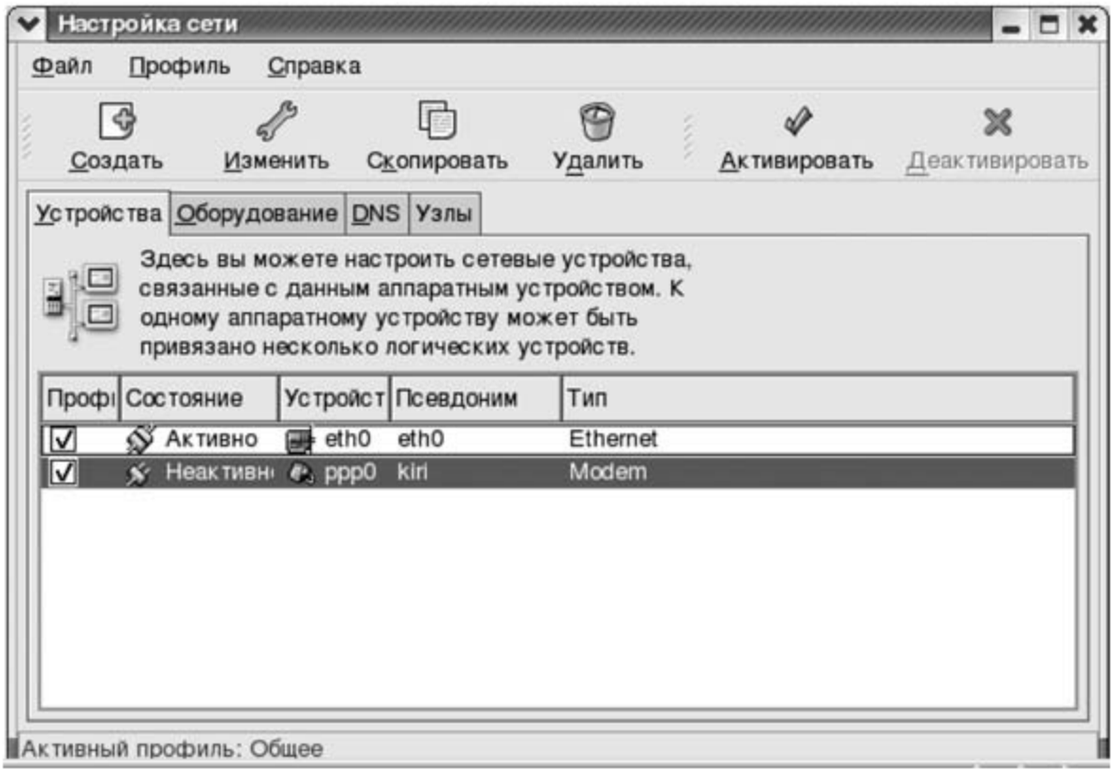


Рис. 8.15

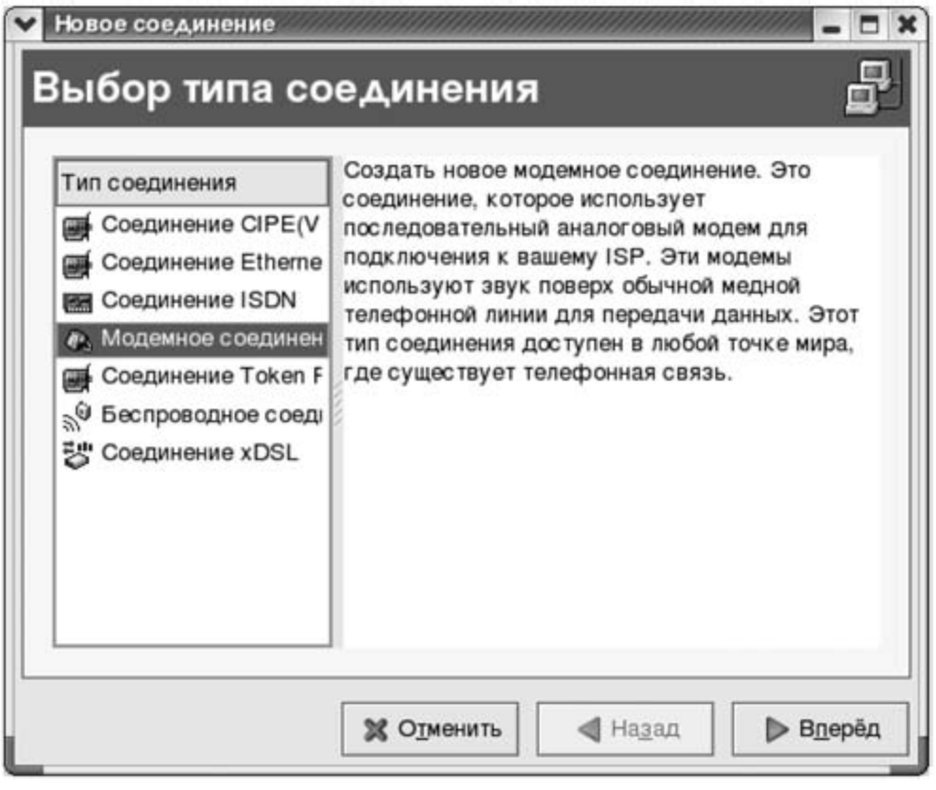


Рис. 8.16

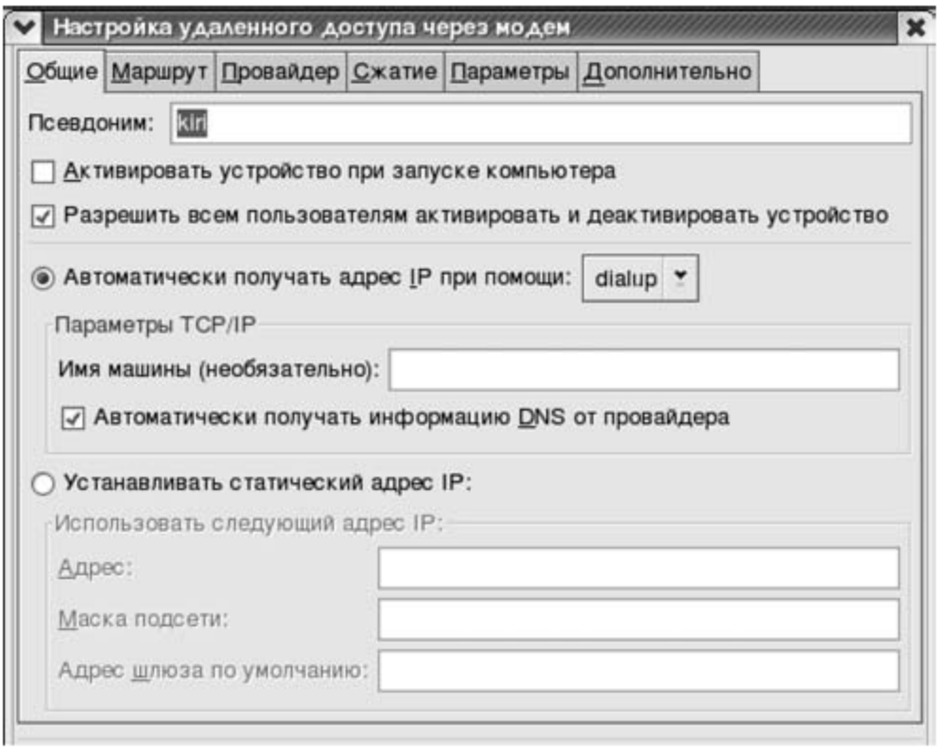


Рис. 8.17

При соединении типа dialup ip-адрес на время сеанса обычно выделяет провайдер, что отмечается меткой в строке Автоматически получать адрес IP при помощи dialup. Информацию о DNS так же чаще получают от провайдера автоматически. Некоторые пользователи покупают себе постоянный ip-адрес. В этом случае вместо получения временного адреса от провайдера на данной вкладке заполняются строки, относящиеся к установлению статического адреса IP (в нижней части окна).

После установки общих настроек начинается настройка окна провайдера (рис. 8.18).

В этом окне определяется номер телефона (а при необходимости — префикс и код города), по которому будет осуществляться дозвон до провайдера. Указывается его имя (обычно в виде 2-уровневого доменного адреса) и учетные параметры (так называемая учетная запись T-Online): учетное имя и пароль — под этими параметрами происходила регистрация пользователя у провайдера. Остальные настройки не обязательные.

После окончания настройки переходим к окну Настройка сети и проверяем вновь созданное соединение, заставляя его работать, для чего запускаем режим Активировать. Модем дозванивается до провайдера, и после выяснения регистрационных данных устанавливается связь. Кнопка Активировать в окне Настройка сети становится бледной и не реагирует на мышь, но активизируется кнопка Деакти-

вировать. Для начала работы в Интернете можно запускать требуемый сервис: WWW, telnet, ftp и др.

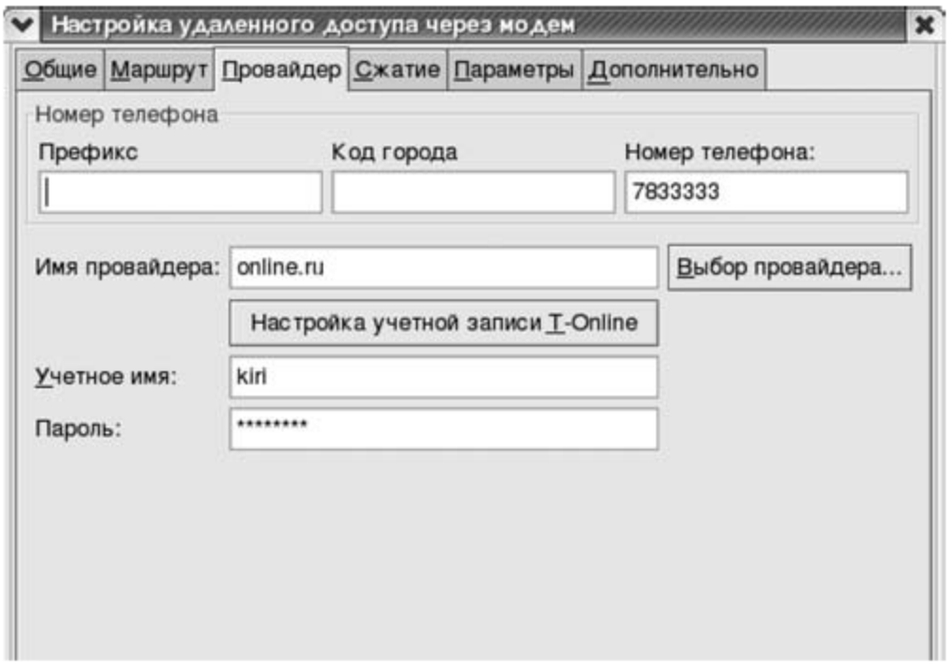


Рис. 8.18

Настройка «Стрим» — скоростного широкополосного соединения с Интернетом типа ADSL. Настройка соединения под названием «Стрим» начинается с режима Создать окна Настройка сети. В окне Создать выбирается тип соединения Соединение xDSL. ADSL — одна из нескольких технологий скоростной широкополосной связи.

Технология ADSL (Asymmetric Digital Subscriber Line — асимметричная цифровая абонентская линия) позволяет передавать информацию с высокой скоростью по телефонным сетям типа «витая пара». Однако эта технология асимметрична, обеспечивает большую скорость при передаче информации из Интернета пользователю (так как обычно пользователь выдает в Интернет короткие запросы, а получать может мегабайтные ответы). Еще одной особенностью этой технологии является то, что передача информации ведется в цифровом виде и не занимает аналоговую телефонную линию (т.е. можно одновременно и работать в Интернете, и вести переговоры по телефону).

Широкополосное оборудование обычно предоставляет услугу, называемую Dynamic Host Configuration Protocol (DHCP). Служба DHCP назначает адрес сети Интернет и предоставляет другую необходимую клиенту для подключения к сети информацию.

Свойства соединения ADSL, необходимые для настройки ОС, немногочисленны, они приводятся в окне Новое соединение (рис. 8.19).

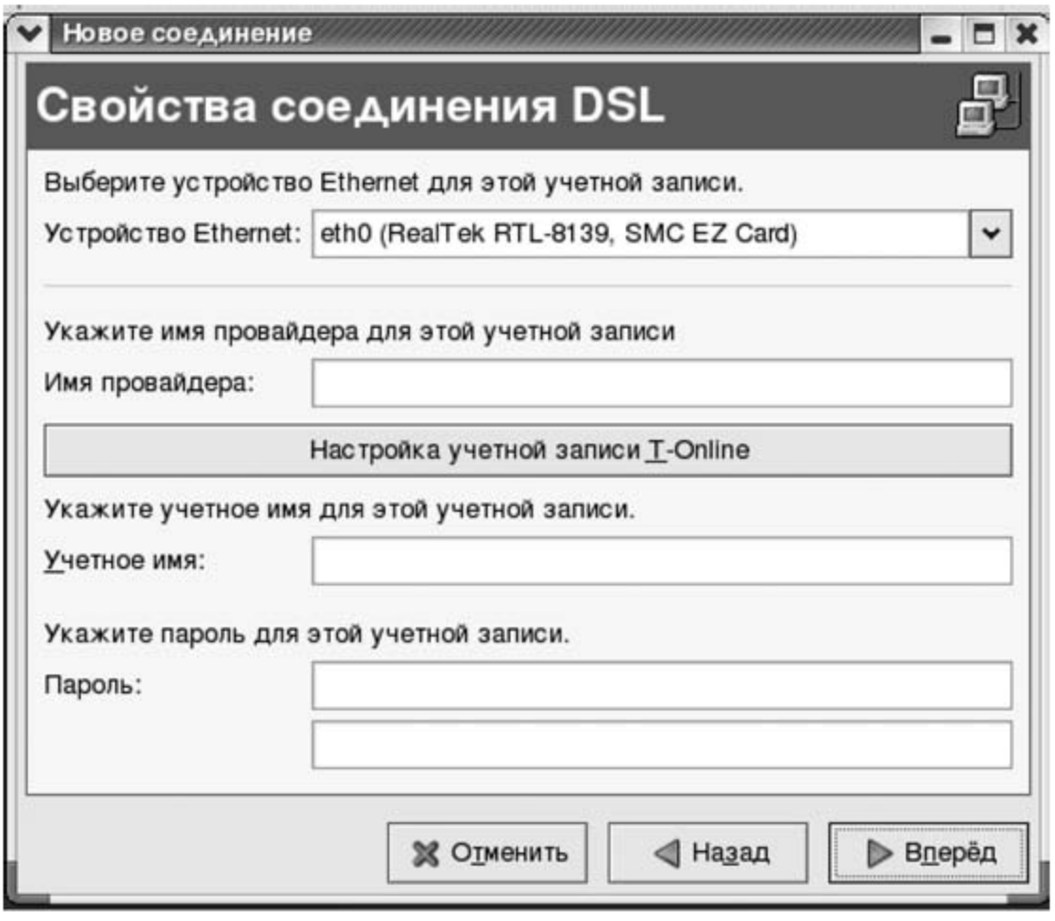


Рис. 8.19

В этом окне указывается имя провайдера (для Москвы — mtu), учетное имя и пароль, выданные при заключении договора на получение услуги «Стрим». Проверить правильность подключения можно активизацией нового соединения.

Параметры нового соединения можно увидеть после подключения, набрав в терминале /sbin/ifconfig -a. Сразу после загрузки ОС, до соединения с провайдером эта команда выведет следующую информацию:

```
[root@localhost root]# /sbin/ifconfig -a
eth0    Link encap:Ethernet HWaddr 00:A1:B0:08:60:E5
        inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:29 errors:0 dropped:0 overruns:0 frame:0
        TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
        RX bytes:3120 (3.0 Kb) TX bytes:2540 (2.4 Kb)
        Interrupt:5 Base address:0xf000
```

```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:13402 errors:0 dropped:0 overruns:0 frame:0
        TX packets:13402 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:914464 (893.0 Kb) TX bytes:914464 (893.0 Kb)
```

Здесь содержится информация о создании двух интерфейсов: локального (lo) и сетевого (eth0, типа Ethernet) — и результатах их проверки при передаче пакетов. После подключения к Интернету несколько изменяется информация о проверке локального и сетевого интерфейсов и появляется информация о создании третьего интерфейса (point-to-point, ppp0):

```
[root@localhost root]# /sbin/ifconfig -a
eth0  Link encap:Ethernet HWaddr 00:A1:B0:08:60:E5
      inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500
Metric:1
      RX packets:225 errors:0 dropped:0 overruns:0 frame:0
      TX packets:239 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:120737 (117.9 Kb) TX bytes:27794 (27.1 Kb)
      Interrupt:5 Base address:0xf000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:14922 errors:0 dropped:0 overruns:0 frame:0
      TX packets:14922 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:1018264 (994.3 Kb) TX bytes:1018264 (994.3 Kb)

ppp0  Link encap:Point-to-Point Protocol
      inet addr:85.140.70.20 P-t-P:85.140.68.1 Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST MTU:
      1492 Metric:1
      RX packets:187 errors:0 dropped:0 overruns:0 frame:0
      TX packets:202 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:3
      RX bytes:112849 (110.2 Kb) TX bytes:20210 (19.7 Kb)
```

Третий интерфейс характеризует связь с провайдером Интернета и сообщает временный ip-адрес подключенного к Интернету локального компьютера.

Проверка правильности подключения адаптера Ethernet. При наличии в компьютере адаптера Ethernet его подключение должно быть обнаружено шиной PCI при выполнении начальной загрузки ОС. Убедиться в том, что адаптер подключен, можно, воспользовавшись конвейером dmesg | grep eth:

```
[root@localhost root]# dmesg | grep eth
eth0: Setting 100mbps full-duplex based on auto-negotiated partner ability 45e1.
divert: not allocating divert_blk for non-ethernet device ppp0
```

Команда dmesg выводит все сообщения ядра Linux в процессе загрузки ОС. Команда grep eth пропускает только строки, содержащие слово eth. Наличие сообщения eth0 говорит о том, что ОС нашла сетевой адаптер и создала для него интерфейс.

Если интерфейс eth0 отсутствует, то узнать, обнаружено ли подключение адаптера Ethernet шиной PCI, можно с помощью конвейера lspci –vv | grep –i eth.

При правильном подключении сетевого адаптера получим сообщение:

```
[root@localhost root]# lspci –vv | grep –i eth
grep: eth: No such file or directory
```

Проверка списка настроенных сетевых интерфейсов. Для просмотра списка настроенных сетевых интерфейсов следует ввести команду ifconfig:

```
[root@localhost root]# ifconfig
eth0  Link encap:Ethernet HWaddr 00:A1:B0:08:60:E5
      inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:318 errors:0 dropped:0 overruns:0 frame:0
      TX packets:332 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:127173 (124.1 Kb) TX bytes:35076 (34.2 Kb)
      Interrupt:5 Base address:0xf000
```

```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:20280 errors:0 dropped:0 overruns:0 frame:0
        TX packets:20280 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:1384159 (1.3 Mb) TX bytes:1384159 (1.3 Mb)

ppp0 Link encap:Point-to-Point Protocol
        inet addr:85.140.70.20 P-t-P:85.140.68.1 Mask:255.255.255.255
        UP POINTOPOINT RUNNING NOARP MULTICAST
        MTU:1492 Metric:1
        RX packets:244 errors:0 dropped:0 overruns:0 frame:0
        TX packets:259 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:3
        RX bytes:115871 (113.1 Kb) TX bytes:24078 (23.5 Kb)
```

В выводе показан локальный интерфейс обратной связи (lo), сетевой адаптер Ethernet (eth0) и адаптер по протоколу Point-to-Point (ppp0). Интерфейсу ppp0 назначен ip-адрес 85.140.70.20.

Проверка работоспособности сетевых настроек компьютера. Для комплексной проверки настроек компьютера на работу с сетью необходимо воспользоваться программой ping, указав ей локальный цифровой ip-адрес: 127.0.0.1. Если сеть настроена верно, то ping отработает, передаст и примет вернувшиеся пакеты и зафиксирует затраченное время:

```
[root@localhost root]# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.076 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.060 ms
...
64 bytes from 127.0.0.1: icmp_seq=23 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=24 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=25 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=26 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=27 ttl=64 time=0.060 ms
--- 127.0.0.1 ping statistics ---
27 packets transmitted, 27 received, 0% packet loss, time
25992ms
rtt min/avg/max/mdev = 0.058/0.062/0.076/0.009 ms
```

Программа ведет счет посылаемым и принимаемым пакетам. Ее выполнение можно остановить нажатием клавиш Ctrl+C. После этого выводится резюме: сколько пакетов отправлено, сколько принято, затраченное время на отправку-получение одного пакета и статистические характеристики: минимальное (min), среднее (ave) и максимальное (max) время пересылки пакета и максимальная девиация (т.е. отклонение) времени пересылки пакета от среднего значения. Время измеряется в миллисекундах.

Проверка выхода в сеть. Комплексная проверка работоспособности в Сети производится следующим образом: дважды запускается программа ping, один раз с указанием ей цифрового ip-адреса какого-либо сайта, другой раз — с указанием его символьного (доменного) адреса (выполнение программы прерываем после отправки пятого пакета):

```
[root@localhost root]# ping www.narod.ru
PING www.narod.ru (213.180.204.46) 56(84) bytes of data.
64 bytes from narod.yandex.ru (213.180.204.46): icmp_seq=1 ttl=60 time=46.2 ms
64 bytes from narod.yandex.ru (213.180.204.46): icmp_seq=2 ttl=60 time=27.9 ms
64 bytes from narod.yandex.ru (213.180.204.46): icmp_seq=3 ttl=60 time=30.9 ms
64 bytes from narod.yandex.ru (213.180.204.46): icmp_seq=4 ttl=60 time=29.2 ms
64 bytes from narod.yandex.ru (213.180.204.46): icmp_seq=5 ttl=60 time=28.4 ms
--- www.narod.ru ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4038ms
rtt min/avg/max/mdev = 27.936/32.549/46.262/6.930 ms
```

Получив доменный адрес, программа ping запрашивает у службы DNS цифровой адрес и дальше работу ведет только с ним. Полученный цифровой адрес выводится в листинге, генерируемом программой. Так, доменному адресу www.narod.ru соответствует цифровой адрес 213.180.204.46.

После проверки доменного адреса проверяем цифровой (выполнение программы прерываем после отправки четвертого пакета):

```
[root@localhost root]# ping 213.180.204.46
PING 213.180.204.46 (213.180.204.46) 56(84) bytes of data.64 bytes from 213.180.204.46:
```

```
icmp_seq=1 ttl=60 time=33.1ms
64 bytes from 213.180.204.46: icmp_seq=2 ttl=60 time=27.0ms
64 bytes from 213.180.204.46: icmp_seq=3 ttl=60 time=27.0ms
64 bytes from 213.180.204.46: icmp_seq=4 ttl=60 time=28.1ms
64 bytes from 213.180.204.46: icmp_seq=5 ttl=60 time=29.0ms
--- 213.180.204.46 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4033ms
rtt min/avg/max/mdev = 27.007/28.893/33.162/2.271 ms
```

Если удается связаться с провайдером по цифровому адресу и не удается по доменному, скорее всего проблема в недоступности сервера DNS, которая может иметь место и при неправильной настройке локальной системы на работу с Сетью.

Идентификация других компьютеров Сети. При работе в Сети при каждом обращении к компьютеру по имени осуществляется преобразование его имени в цифровой ip-адрес. Это преобразование выполняется последовательно, с помощью различных баз данных. Порядок опроса баз данных определяется тремя файлами в каталоге: /etc: resolv.conf, nsswitch.conf и host.conf.

По умолчанию сначала рассматриваются имена узлов, добавленные администратором (из файла /etc/hosts), потом рассматриваются имена узлов в базе данных NIS, после чего опрашивается база данных службы DNS.

Содержимое файла /etc/resolv.conf:

```
[root@localhost root]# more /etc/resolv.conf
; generated by /sbin/dhclient-script
search home
nameserver 212.188.4.10
nameserver 195.34.32.116
```

Здесь содержатся адреса основного и вспомогательного DNS-серверов. Именно с их помощью резолвер (программа, определяющая цифровой ip-адрес по доменному, и наоборот) осуществляет преобразование (разрешение) адресов. Обращение к резолверу производят разные программы (и при этом они сообщают результат разрешения адресов). К ним относится такая программа, как ping.

Файл /etc/nsswitch.conf является конфигурационным файлом службы имен NSS.

Файл /etc/host.conf является вспомогательным конфигурационным файлом службы имен, обычно используемым для задания имен компьютерам своей локальной сети.

Проверка связи с сервером DHCP. Компьютер клиента постоянного адреса Интернета не имеет. Временный (иначе — арендованный) адрес ему присваивается провайдером (с сервера DHCP) на время сеанса. После окончания сеанса этот адрес возвращается в распоряжение провайдера и может быть присвоен очередному присоединившемуся к DHCP клиенту.

Информация об арендованном адресе DHCP может быть получена по команде /sbin/ifconfig ppp0:

```
[root@localhost root]# /sbin/ifconfig ppp0
ppp0 Link encap:Point-to-Point Protocol
     inet addr:85.140.70.20 P-t-P:85.140.68.1 Mask:255.255.255.255
     UP POINTOPOINT RUNNING NOARP MULTICAST
     MTU:1492 Metric:1
     RX packets:319 errors:0 dropped:0 overruns:0 frame:0
     TX packets:334 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:3
     RX bytes:122878 (119.9 Kb) TX bytes:31048 (30.3 Kb)
```

Из приведенного листинга видно, что вышедшему на связь с провайдером клиенту присвоен временный адрес 85.140.70.20.

ГЛАВА 9

ВИРТУАЛИЗАЦИЯ. МНОЖЕСТВЕННЫЕ ПРИКЛАДНЫЕ СРЕДЫ

9.1. Варианты организации множественных прикладных сред

Множественные прикладные среды, т.е. системы, позволяющие на одном компьютере работать с программным обеспечением, разработанным для различых ОС, можно реализовать по-разному. Такую среду можно получить:

1) сделав компьютер мультизагрузочным — установить несколько различных операционных систем (Unix, Windows, MS DOS, Linux и др.);
2) загружая различные операционные системы с компакт-диска (CD) без инсталляции на жесткий диск;
3) воспользовавшись эмуляторами операционных систем;
4) воспользовавшись программным обеспечением системы виртуальных машин.

Чтобы сделать компьютер мультизагрузочным, нужно уметь выполнять различные довольно сложные операции: форматирование жестких дисков, разделение их на разделы, инсталляцию различных ОС, в процессе которой часто производится их подстройка. Устанавливать различные ОС надо в определенной последовательности, такие тонкости в литературе могут быть и не описаны, и постигать их приходится экспериментально, допуская достаточно много ошибок. Гораздо проще загружать различные ОС с компакт-диска или с DVD. Для этого нужно лишь иметь устройство чтения CD-DVD и в BIOS разрешить загрузку с компакт-диска.

В любом случае на компьютер может быть загружена только одна ОС, и поэтому одновременно воспользоваться программами, написанными для разных ОС, не представляется возможным. Поэтому наиболее доступными, хотя и менее распространенными, являются последние два способа — эмуляторы и средства создания систем виртуальных машин.

9.2. Эмуляторы

Эмуляторы для различных ОС можно найти в Интернете. Один из небольших эмуляторов Unix для работы с ним под Windows (эмулятор Степанищева) можно получить с сайта http://junix.kzn.ru/. Этот эмулятор выполнен на языках HTML и Java Script, запускается с помощью Internet Explorer. Его экран выглядит, как представлено на рис. 9.1:

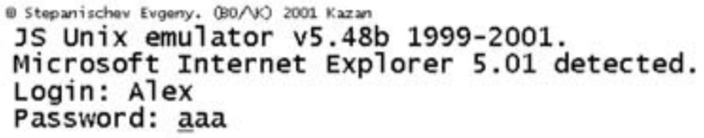


Рис. 9.1

Эмулятор Степанищева предназначен для изучения системы команд Unix, в него включен довольно полный справочник (man). Например, подсказка по одной из команд в нем выглядит, как изображено на рис. 9.2:

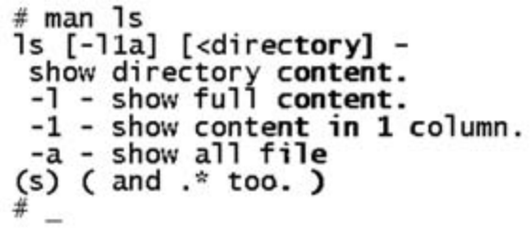


Рис. 9.2

К достоинствам этого эмулятора можно отнести и то, что он не нуждается в инсталляции и готов к использованию с любого носителя, даже из Интернета. Более функциональный эмулятор Cygwin (Цигвин) представляет собой набор программных инструментов, разработанных фирмой Cygnus Solutions. Изначально Cygwin задумывался как среда для переноса программ из POSIX-совместимых ОС (таких как GNU/Linux, BSD и UNIX) в Windows.

Эмулятор Cygwin представляет собой библиотеку, которая реализует интерфейс прикладного программирования POSIX на основе системных вызовов Win32. Кроме того, он содержит инструменты разработки GNU для выполнения основных задач программирования, а также и некоторые прикладные программы, эквивалентные базовым программам UNIX. Подобные функциональные возможности предлагает также и Microsoft в пакете Services for UNIX, включающем в себя подсистему Interix.

В настоящее время проект Cygwin разрабатывается сотрудниками Red Hat и другими программистами, которые реализуют библиотеку Cygwin под лицензией GNU GPL, с оговоркой, что разрешается ее свободное использование с любым свободным программным обеспечением с открытым кодом. Для распространения коммерческих программ, использующих библиотеку Cygwin, необходимо приобретение лицензии у Red Hat.

Работа над проектом Cygwin была начата в 1995 г. Стивом Чемберленом, программистом Cygnus. С 1998 г. Cygnus предлагает пакет Cygwin в качестве самостоятельного продукта. В 2001 г. в Cygwin был включен пакет X Window System.

Для инсталляции эмулятора Cygwin нужно запустить программу Cygwin.exe с домашней страницы Cygwin Home, которая расположена по адресу http://www.cygwin.com. После запуска программы инсталляции предлагается выбор вариантов загрузки эмулятора. Лучше всего инсталляцию проводить из Интернета. Для этого следует выполнять все указания, стараясь соглашаться с предлагаемыми решениями.

После инсталляции в директорию C:\cygwin следует поместить папку unixtools, которую отдельно можно получить из Интернета (например, из http://cygwin.com/ported.html). Кроме указанной папки можно поместить и другие программы Unix. В итоге можно получить набор средств, показанных на рис. 9.3, а на экране появится иконка cygwin.ico, щелчок по которой вызовет окно, приведенное на рис. 9.4. Курсор во второй строке является приглашением к вводу Unix-команд. Эмулятор Cygwin также, как и эмулятор Степанищева, имеет свой справочник команд (рис. 9.3).

Рис. 9.3

Рис. 9.4

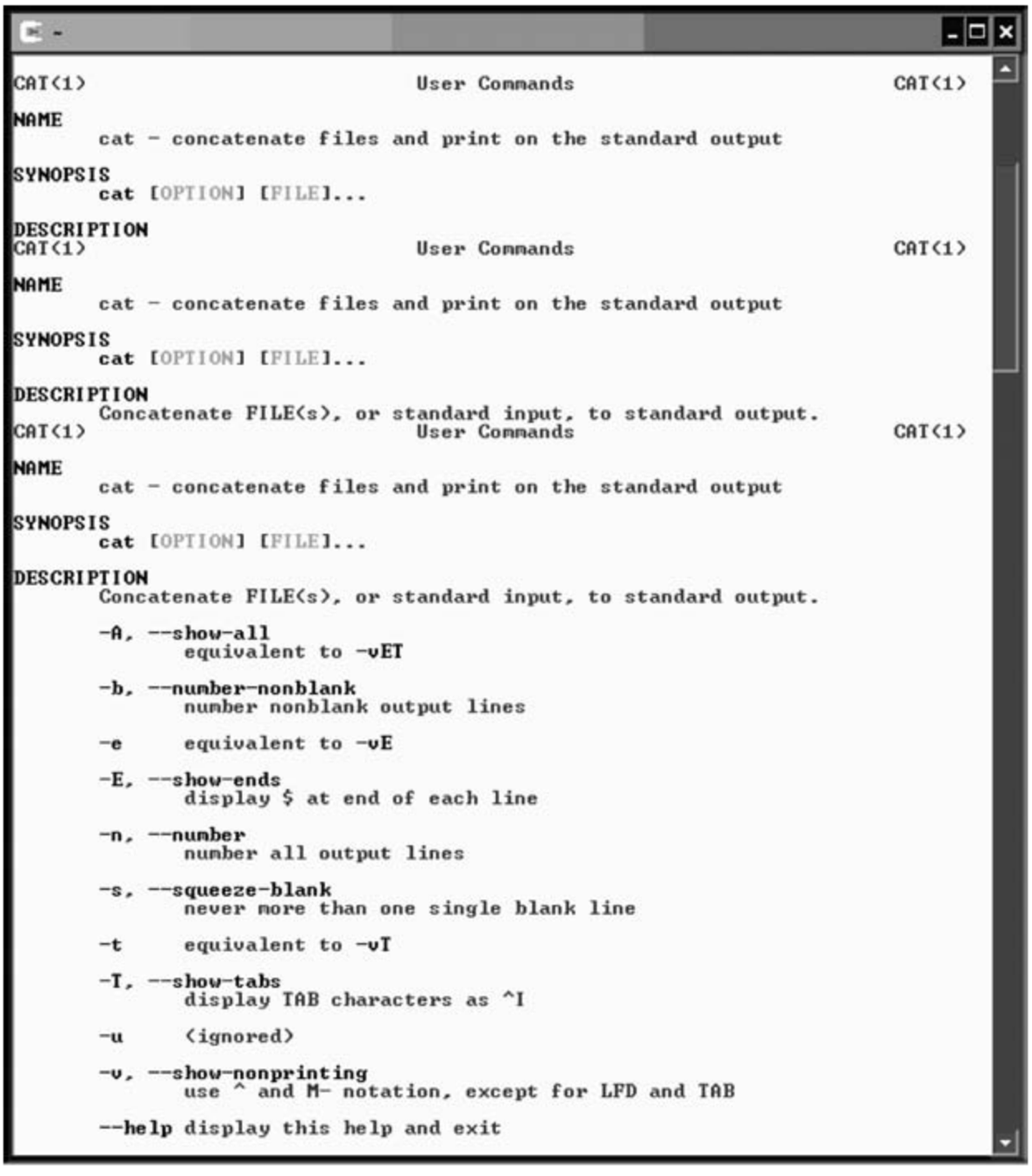


Рис. 9.5

Недостатком эмулятора является игнорирование кириллицы — вместо нее появляются вопросительные знаки.

Команда df позволяет увидеть связь файловых систем Windows и эмулятора (рис. 9.6).

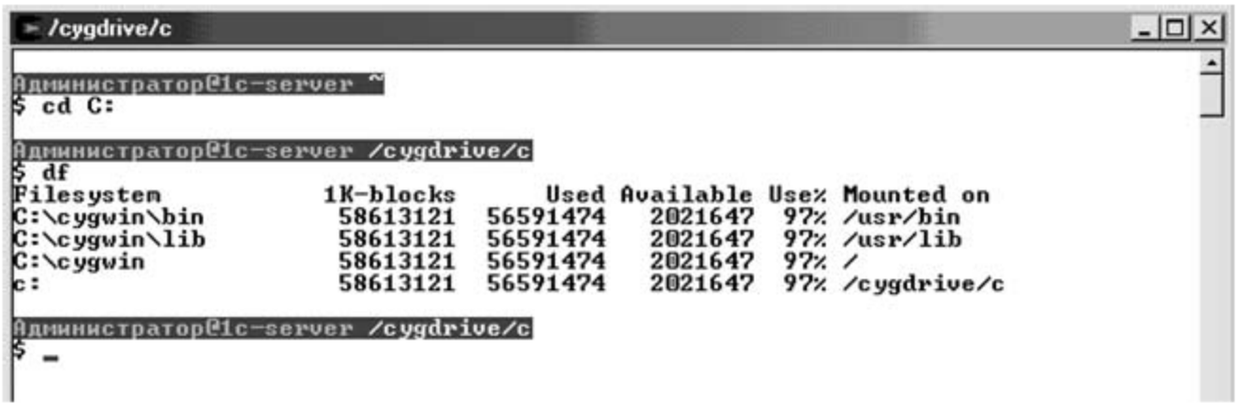


Рис. 9.6

Эмулятор Cygwin монтирует локальные диски в директории /cygdrive. Сюда входят разделы жесткого диска, дисководы, устройства для чтения компакт-дисков, устройства для чтения ZIP. При выполнении команды df файлы, сохраненные в директории C:\cygwin\, используются эмулятором Cygwin в качестве корневой точки монтирования файловой системы. Эмулятор смонтировал C:\cygwin\bin как /usr/bin и C:\cygwin\lib как /usr/lib. В директориях /usr/bin, /bin, и /usr/local/bin были добавлены переменные окружения Cygwin (но не Windows). Директории из среды окружения Windows были импортированы в среду Cygwin, чтобы иметь к ним доступ.

Эмулятор Cygwin позволяет не только выполнять Unix/Cygwin-приложения, но и запускать Windows-приложения из командной строки. Для этого необходимо воспользоваться Windows-командой command.com (рис. 9.7).

```
Администратор@ic-server /cygdrive/c/Program Files
$ command.com
    5405 [main] bash 1332! _pinfo::dup_proc_pipe: DuplicateHandle failed, pid 133
2, hProcess 0x2D9, wr_proc_pipe 0x338, Win32 error 6
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-1999.

C:\PROGRA~1>cd\

C:\>_
```

Рис. 9.7

С помощью команды exit (рис. 9.8) можно вернуться к работе с Unix (Linux).

Эмулятор Cygwin позволяет не только работать с файлами и запускать программы, но и составлять программы на языках C, C++, Perl. Можно использовать наиболее интересные утилиты Unix, приложения для Unix с открытым программным кодом. Внутри Cygwin можно запустить X Window (для этого следует обратиться к http://

cygwin.com/xfree/ и выбрать зеркало для скачивания двоичных файлов X Window).

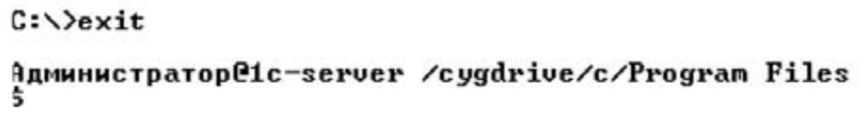


Рис. 9.8

Можно также загрузить один из оконных менеджеров: KDE, Gnome, или Window Maker, каждый из которых доступен для Cygwin.

Доступные ссылки из Интернета:

- http://www.cygwin.com;
- http://freshmeat.net/projects/cygwin/;
- http://cygnome/sourceforge.net;
- http://kde-cygwin.sourceforge.net;
- http://unxutils.sourceforge.net.

9.3. Организация системы виртуальных машин под управлением ОС Windows

9.3.1. Система виртуальных машин VMware

Виртуальная машина — компьютер, не существующий в действительности. Его структура создается программным путем на реально существующем компьютере, который считается основной машиной, или host-компьютером. Для создания системы виртуальных машин используется специальное программное обеспечение, установленное на host-компьютере, — программное обеспечение системы виртуальных машин (СВМ). Это программное обеспечение работает под управлением основной ОС и создает один или несколько виртуальных компьютеров, каждый из которых имеет все компоненты реального компьютера: центральный процессор, оперативную память, жесткий диск, клавиатуру, мышь и т.д.

Естественно, что физически существует только один компьютер, а на нем моделируется работа нескольких компьютеров, каждый из которых может иметь свою собственную ОС и свои ресурсы. После запуска всех виртуальных компьютеров создается система, в которой на одном компьютере могут одновременно работать несколько разных ЭВМ со своими ОС, и переход с одного компьютера на другой

(а следовательно, замена ОС) может осуществляться путем перехода из одного окна в другое без перезапуска компьютера.

Чаще всего система виртуальных машин и создается для того, чтобы иметь возможность на одном компьютере работать под разными ОС. Виртуальные машины могут быть независимыми, а могут быть соединены в сети разной конфигурации.

По существу, СВМ представляет собой слой программного обеспечения, расположенный между основной ОС и ОС виртуальной машины, в которой в свою очередь запускаются прикладные программы. Для создания СВМ специальное программное обеспечение предлагают различные фирмы. Наибольшую популярность приобрели программы фирмы VMware.

Система виртуальных машин от VMware (VMware Virtual Platform) обладает следующими возможностями:

- позволяет запускать одновременно несколько различных ОС на одном стандартном персональном компьютере;
- позволяет запускать виртуальные машины в окнах рабочего стола или в полноэкранном режиме (другие виртуальные машины в это время будут продолжать работать в фоновом режиме, а для переключения между виртуальными машинами используются «горячие» клавиши);
- обеспечивает запуск ОС, ранее установленных на компьютере, с многовариантной загрузкой ОС, без их переустановки или переконфигурирования;
- дает возможность устанавливать виртуальные машины и новые ОС без переразбиения дисков;
- позволяет запускать приложения Microsoft Windows на компьютере с ОС Linux, и наоборот, приложения Linux — на компьютере с ОС Microsoft Windows;
- обеспечивает совместное использование файлов и приложений разными виртуальными машинами за счет использования виртуальной сети (даже в пределах одного компьютера);
- позволяет запускать клиент-серверные или Web-приложения на одном ПК, запуская серверную часть на одной виртуальной машине, а клиентскую — на другой.

При использовании программного обеспечения этой фирмы организация под ОС Windows СВМ предусматривает выполнение следующих действий:

1) инсталляции VMware на host-машину (т.е. на основной компьютер системы виртуальных машин); после инсталляции host-машина продолжает работать под основной ОС;

2) загрузки VMware;
3) создания с помощью VMware виртуальной машины (ВМ);
4) инсталляции ОС на созданную ВМ.

9.3.2. Инсталляция VMware

Архив VMware содержит программы, показанные на рис. 9.9.

Рис. 9.9

Выделенное серым фоном название программы является инсталлятором. После ее запуска открывается окно, приведенное на рис. 9.10.

Как обычно, при инсталляции необходимо ознакомиться и принять условия лицензии. После принятия условий VMware Workstation предложит папку для размещения пакета (рис. 9.11) и сообщит, что готов инсталлировать пакет.

При нажатии кнопки Install начинается установка пакета на компьютер (рис. 9.12).

В процессе инсталляции у системы возникает вопрос, использовались ли на данном компьютере более ранние версии VMware Workstation (рис. 9.13).

Инсталляция СВМ заканчивается, как показано на рис. 9.14.

Рис. 9.10

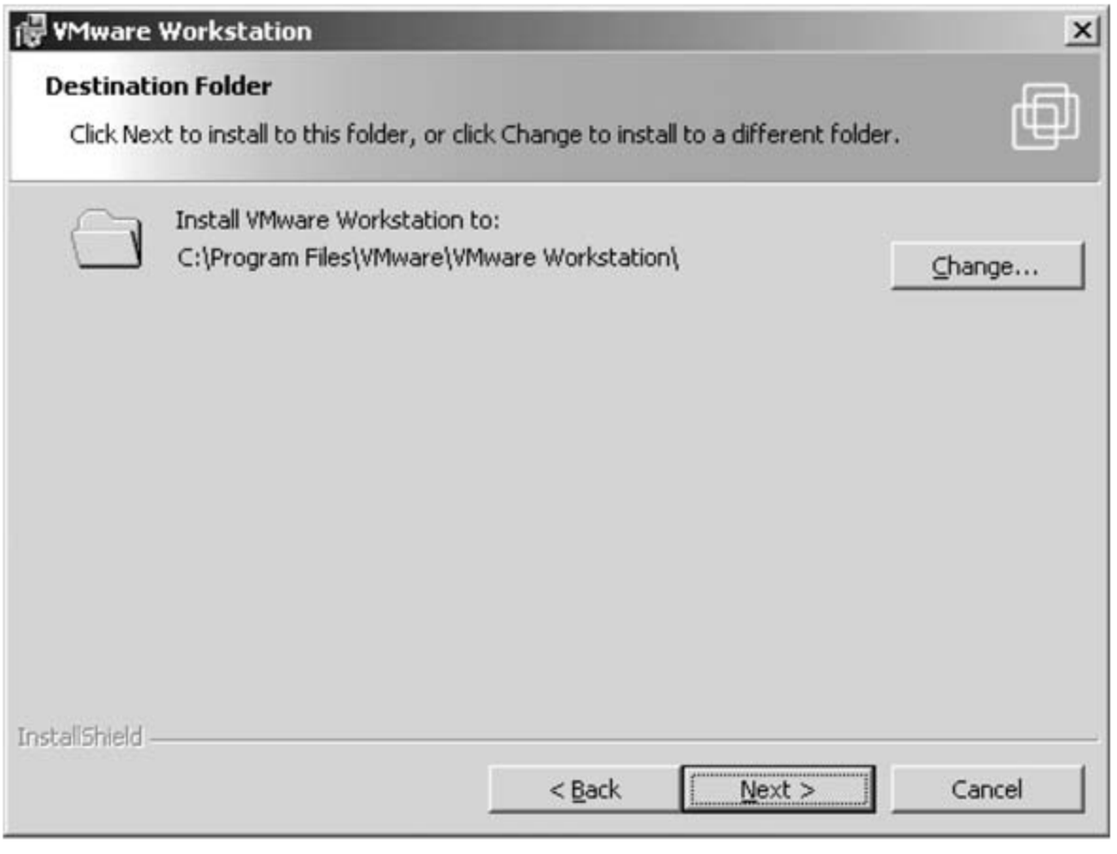


Рис. 9.11

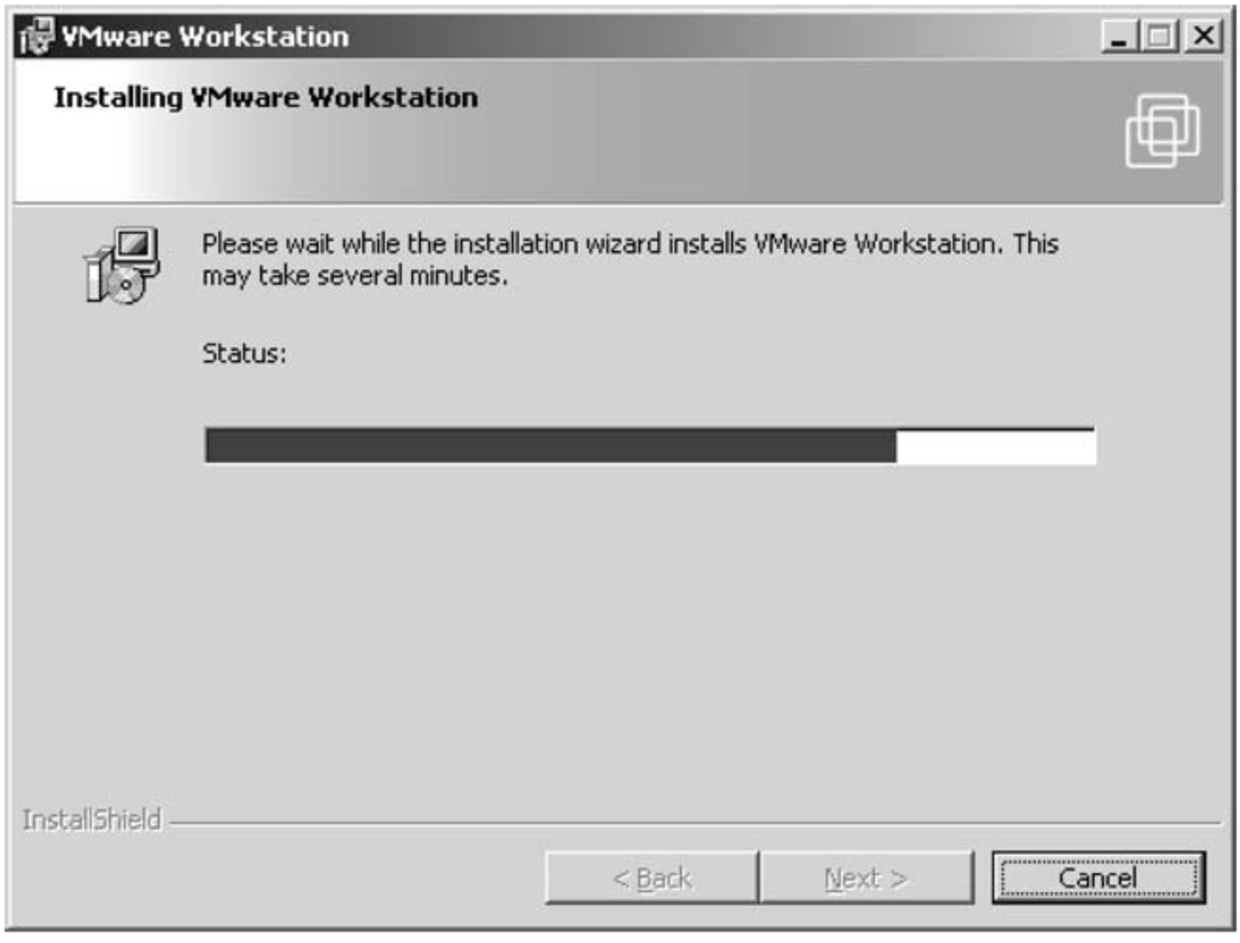


Рис. 9.12

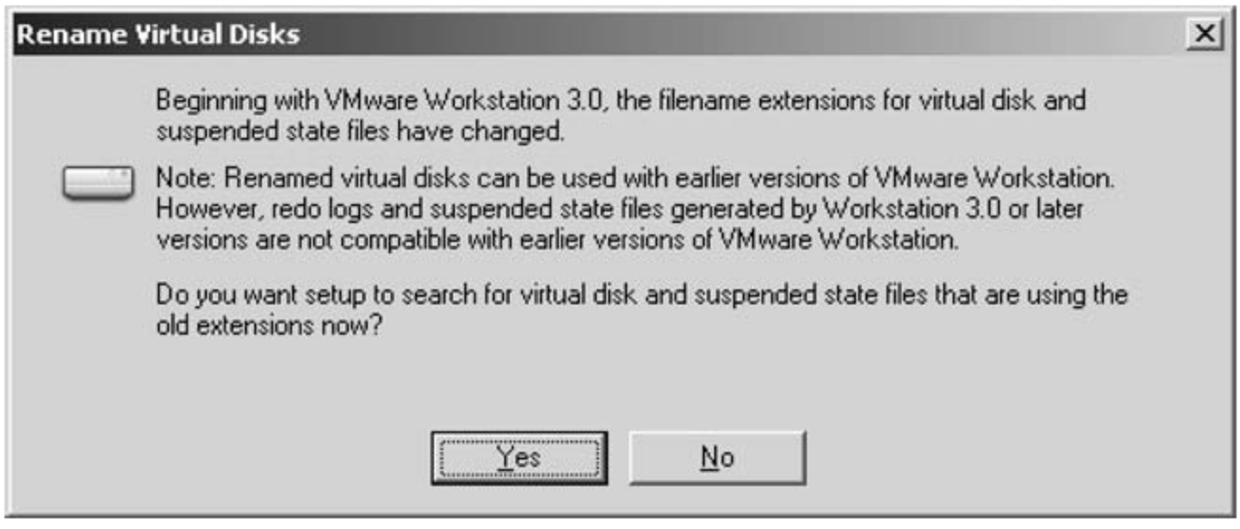


Рис. 9.13

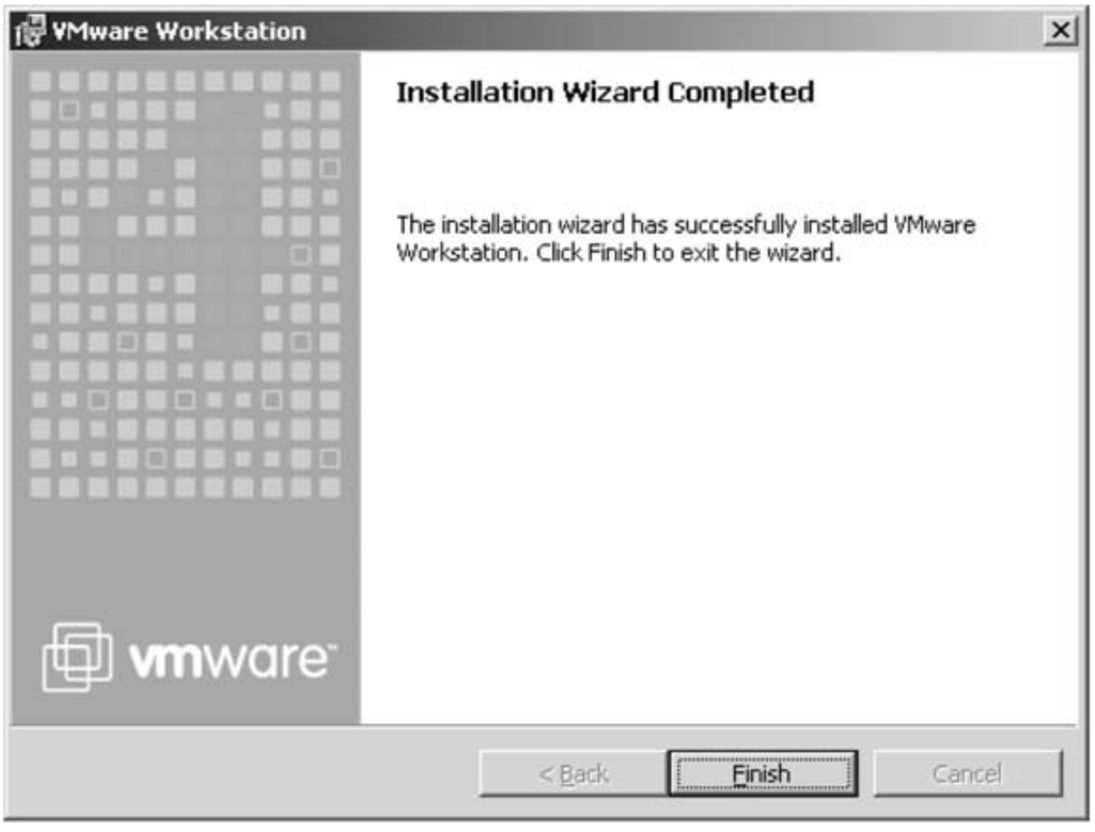


Рис. 9.14

Теперь можно запустить VMware, выполнив команды: Пуск –> Все программы. На экране появляется основное окно СВМ (рис. 9.15).

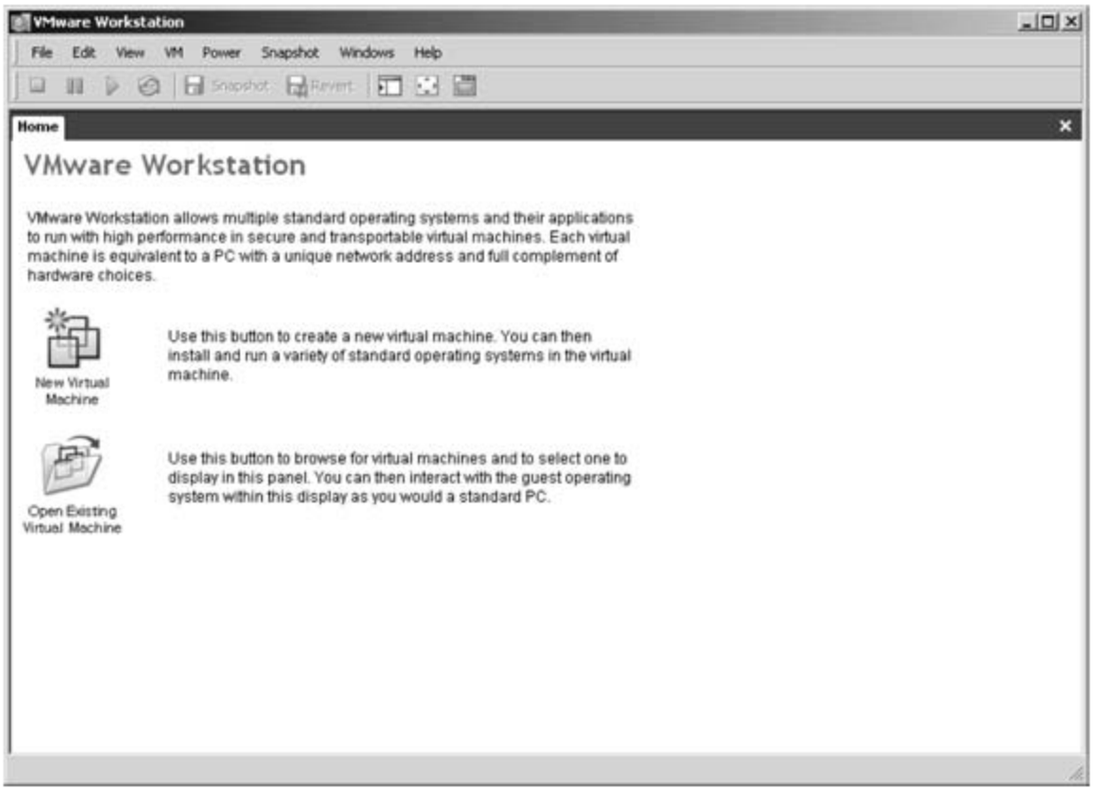


Рис. 9.15

Рассмотрим возможности, которые предоставляет меню пакета VMware. Пункт меню File (рис. 9.16) позволяет приступить к созданию новой виртуальной машины или нового окна, открыть или закрыть существующую виртуальную машину, захватить экран. Пункт меню Edit (см. рис. 9.16) имеет стандартные для Edit возможности (вырезать, копировать, вставить) и дополнительные: настройку виртуальной сети, и преференции (размещение в памяти системы виртуальных машин, особенности использования и настройка клавиатуры, выделяемый виртуальным машинам размер памяти и т.д.).

Рис. 9.16

В пункте меню View можно изменять режим отображения виртуальной машины: вывод информации на полный экран (кроме информации выбранной виртуальной машины на экран не выводится ничего); режим «быстрое переключение», когда при нажатии клавиши F11 полный экран переключается на промежуточный режим, в котором можно взаимодействовать с окном VMware, и наоборот; включить режим перехода в окно Home по клавише Tab; отображать на экране окно Favorites (с перечнем установленных виртуальных машин); определять особенности использования режима вставки; изменять структуру основного окна VMware и т.п.

Меню VM (рис. 9.17) позволяет подстраивать используемые устройства, обновлять структуру и состав виртуальных машин, устанавливать дополнительное программное обеспечение VMware и т.д.

С помощью меню Power (рис. 9.18) можно включать, выключать питание виртуальных машин, приостанавливать и восстанавливать их работу.

Меню Snapshot (рис. 9.19) позволяет делать моментальный снимок состояния системы и восстанавливать состояние системы, возвращаясь к предыдущему.

С помощью меню Windows (рис. 9.20) можно переключать окна запущенных виртуальных машин.

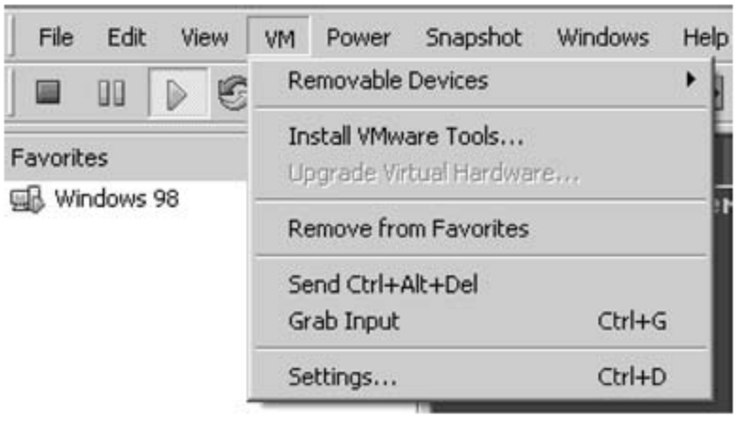


Рис. 9.17

Рис. 9.18

Рис. 9.19

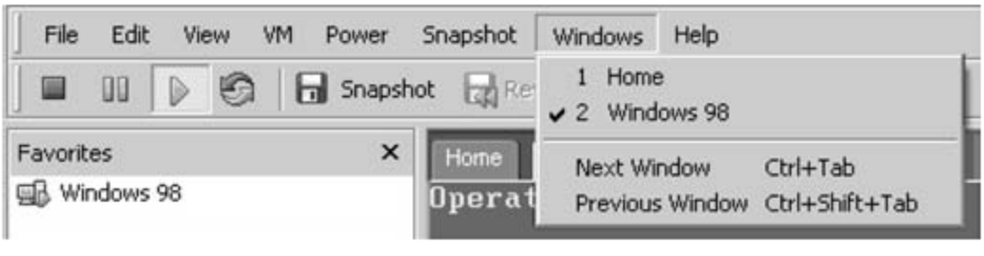


Рис. 9.20

Меню Help (рис. 9.21) позволяет использовать различные системы помощи СВМ.

Рис. 9.21

9.3.3. Загрузка VMware

В результате инсталляции программа VMware версии 4.5.1-7568 размещается в папке C:\Program Files\VMware Workstation (рис. 9.22).

Рис. 9.22

При запуске через меню Пуск –> Все программы –> VMware открывается меню для выбора одной из двух функций:

1) Manage Virtual Networks;

2) VMware Workstation.

При выборе второй функции появляется окно, изображенное на рис. 9.23.

Система VMware установлена, и теперь можно перейти к созданию отдельных виртуальных машин для каждой из ОС, с которыми предстоит работать.

Рис. 9.23

9.3.4. Создание виртуальной машины

При создании виртуальной машины нужно определиться сначала, где будет находиться загружаемая ОС. Она может находиться:

- на CD-ROM или DVD;
- в файле с расширением.iso на жестком диске host-машины;
- в виде ранее установленной на мультизагрузочной host-машине ОС.

При размещении ОС на CD-ROMe или компакт-диске DVD с ОС придется при каждой загрузке держать в устройстве чтения CD-DVD.

При размещении ОС в файле с расширением.iso на жестком диске host-машины указанный файл, представляющий собой образ компакт-диска, перед созданием виртуальной машины нужно разместить на реальном диске ЭВМ и указать это в конфигурации виртуальной машины. При размещении ОС в виде ранее установленной на мультизагрузочной host-машине операционной системы под эту ОС должен был быть выделен специальный раздел реального жесткого диска, после чего произведена инсталляция этой ОС. Первый из этих способов самый простой, но и достаточно неудобный, так как диск с ОС должен постоянно находиться на устройстве чтения CD-DVD.

Наиболее удобным является второй способ, когда все ОС для всех виртуальных машин находятся в соответствующих iso-файлах на реальном жестком диске host-машины. Для создания новой виртуальной машины в окне VMware выбирается функция New Virtual Mashine.

При нажатии кнопки New Virtual Mashine запускается Wizard — мастер создания новой виртуальной машины (рис. 9.24).

Рис. 9.24

Мастер прежде всего предлагает определить, создавать ли типовую конфигурацию виртуальной машины или пользователь будет ее изменять (рис. 9.25). Проще выбрать типовую конфигурацию, так как впоследствии ее можно будет легко изменить. Затем возникает вопрос: к какому типу относится гостевая ОС создаваемой виртуальной машины (рис. 9.26)?

После выбора типа ОС возникает необходимость уточнить версию ОС, а затем имя виртуальной машины и место ее размещения. Например, при указании типа ОС Microsoft Windows выберем версию Windows 2000 Professional и имя Windows 98 (рис. 9.27).

После этого задается вариант подключения виртуальной машины к локальной сети (рис. 9.28). Вариант Bridged Networking означает, что виртуальная машина будет непосредственно подключаться к локальной сети, используя реальную Ethernet-плату host-компьютера. Вариант Host-Only Networking подразумевает включение в виртуальную сеть, состоящую из базового компьютера с его ОС, и всех виртуальных машин, запущенных из VMware. При этом реальная сеть будет доступна через

базовый компьютер. Он будет выполнять роль моста между виртуальной сетью и реальной ЛВС, к которой подключен базовый компьютер.

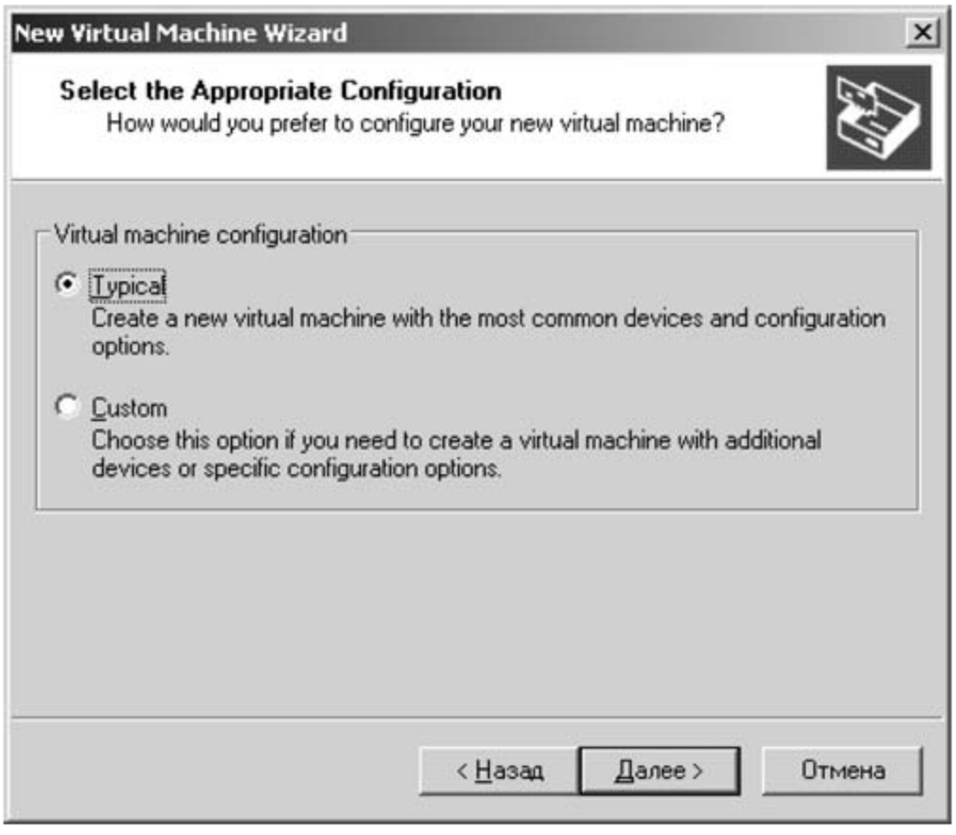


Рис. 9.25

Рис. 9.26

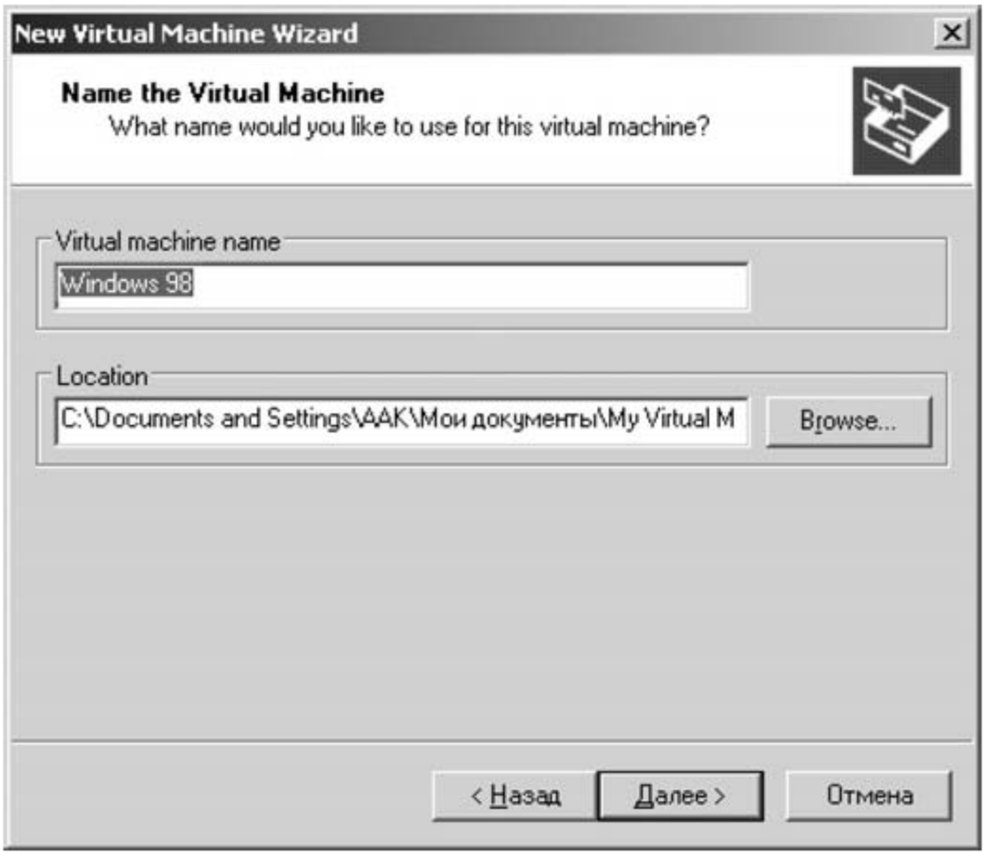


Рис. 9.27

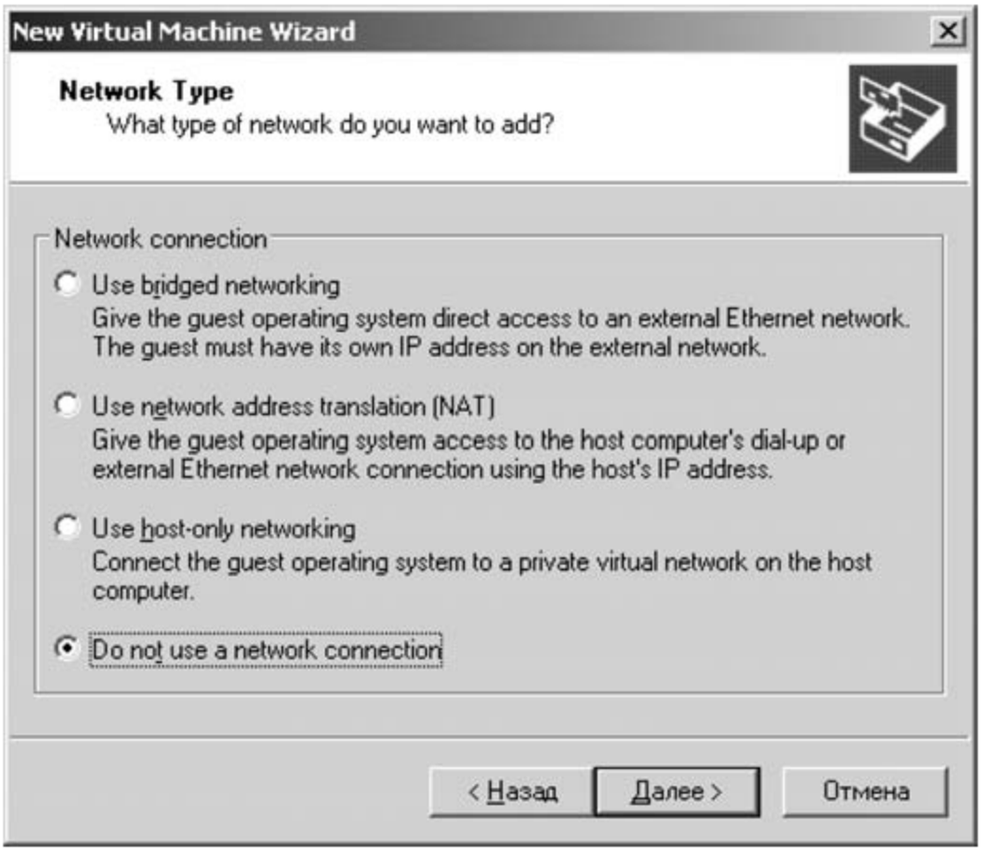


Рис. 9.28

При выборе опции Bridged and Host-Only Networking виртуальный компьютер сможет использовать как реально существующее Ethernet-соединение основного компьютера, так и виртуальную сеть. Самый простой вариант — отказаться от использования сети (впоследствии можно будет все переопределить).

Затем решается вопрос, какой объем памяти должен быть выделен создаваемой виртуальной машине и как (рис. 9.29):

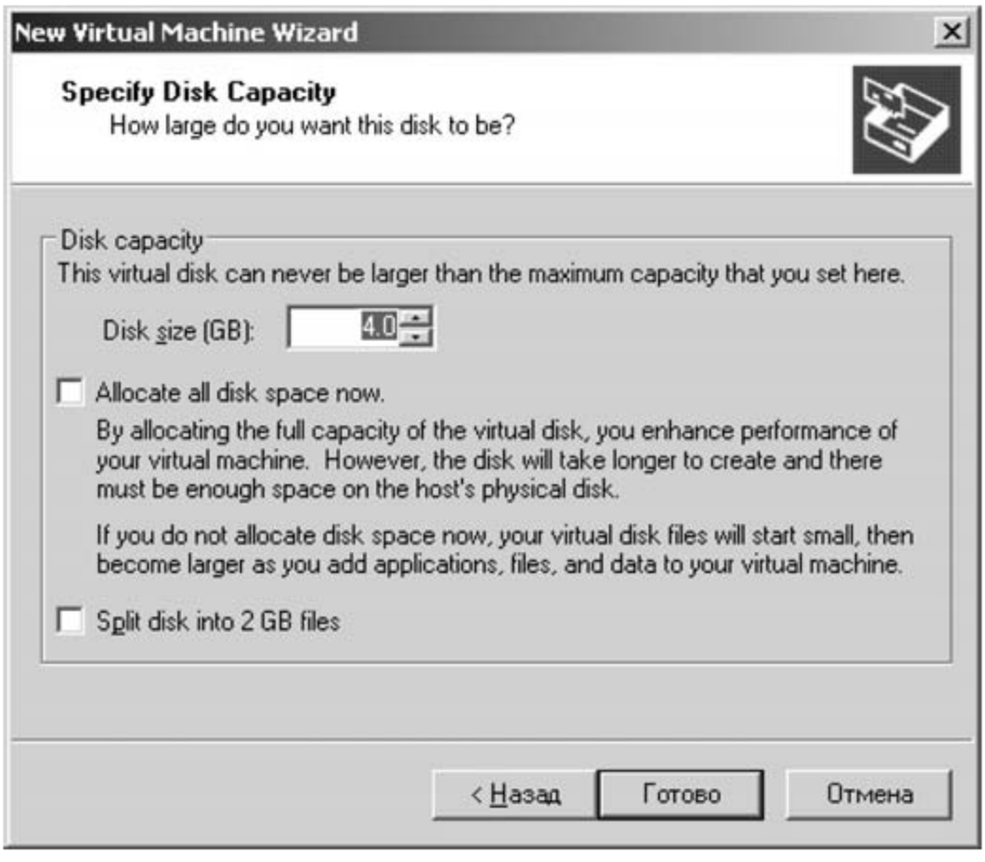


Рис. 9.29

■ выделить всю память сразу или добавлять ее по мере необходимости;

■ выделять ее целым массивом или дробить ее на файлы по 2 Гбайт.

На этом создание виртуальной машины завершено.

9.3.5. Инсталляция гостевых ОС

Инсталляция ОС на вновь созданную виртуальную машину осуществляется точно так же, как инсталляция ОС на физический компьютер:

1) установить в дисковод CD-ROM или гибкий диск с установочными файлами ОС или указать, где находится iso-файл;

2) включить виртуальную машину — щелкнуть мышкой по клавише Power On на корпусе виртуальной машины;
3) следовать инструкциям, которые выдает программа инсталляции.

На сайте фирмы VMware можно найти более подробные рекомендации по установке на ВМ конкретных ОС.

Рассмотрим, как выполняется инсталляция операционной системы, загрузка которой каждый раз будет происходить с загрузочного CD:

1) создаем виртуальную машину RedHat7-1;
2) на CD-ROM устанавливаем диск с ОС RedHat 7.1;
3) сразу после нажатия кнопки Power-On программа VMware начинает инсталляцию этой виртуальной машины.

После завершения инсталляции на экран компьютера выводится окно, приведенное на рис. 9.30.

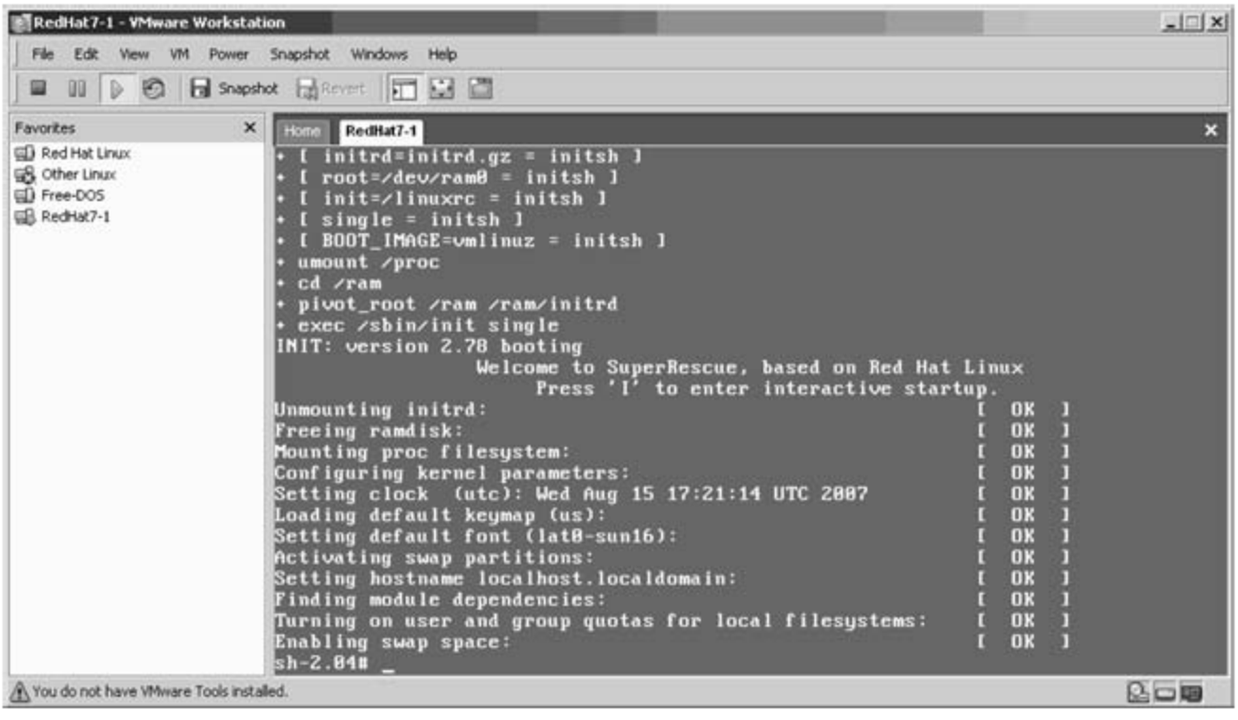


Рис. 9.30

При инсталляции загрузчиком был задан вопрос: загружать ли именно данную систему (рис. 9.31)? Ответ на этот вопрос нажатием клавиши Enter система не принимала: пока на экране есть курсор мыши, клавиатура в черном поле не работает. Надо щелкнуть мышью в черном поле, после чего курсор мыши исчезнет и начнет действовать клавиатура, затем следует нажать клавишу Enter. В остальном вся загрузка прошла по умолчанию.

После загрузки курсор устанавливается возле названия ОС sh-2.04. Определимся, где мы находимся. Для этого выполним команду ls –l. В окне высветится система каталогов из корня новой ОС (рис. 9.32).

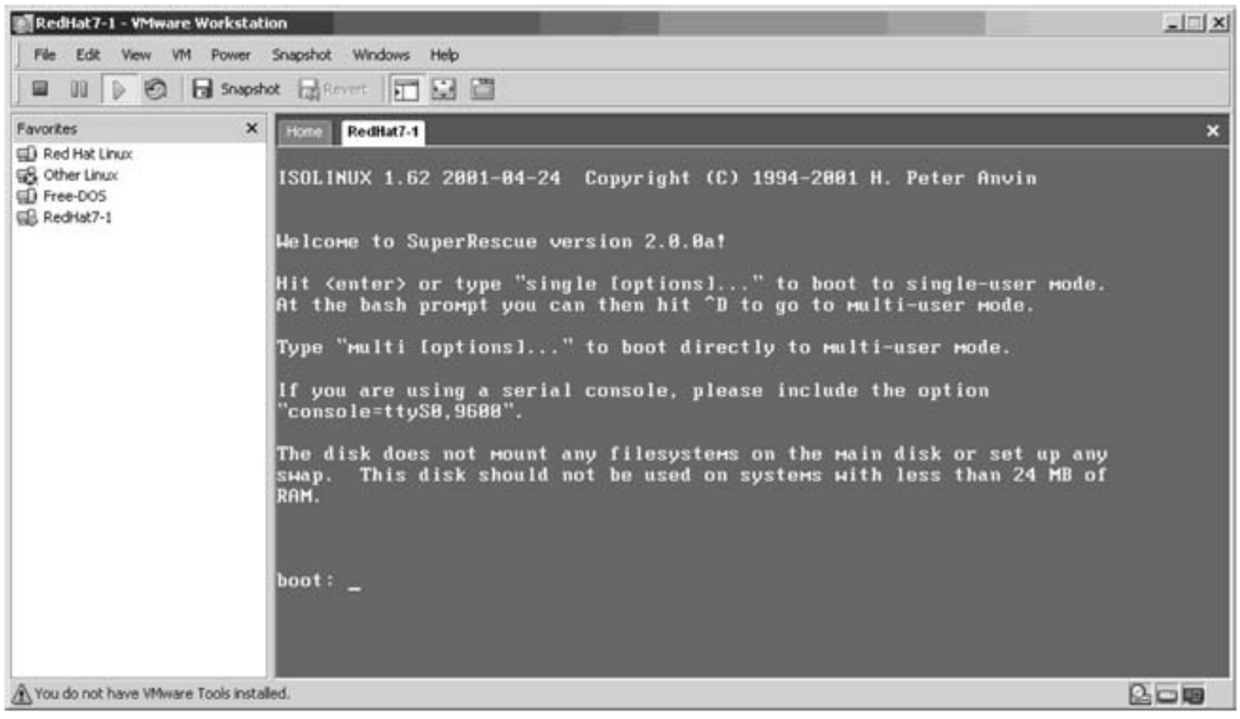


Рис. 9.31

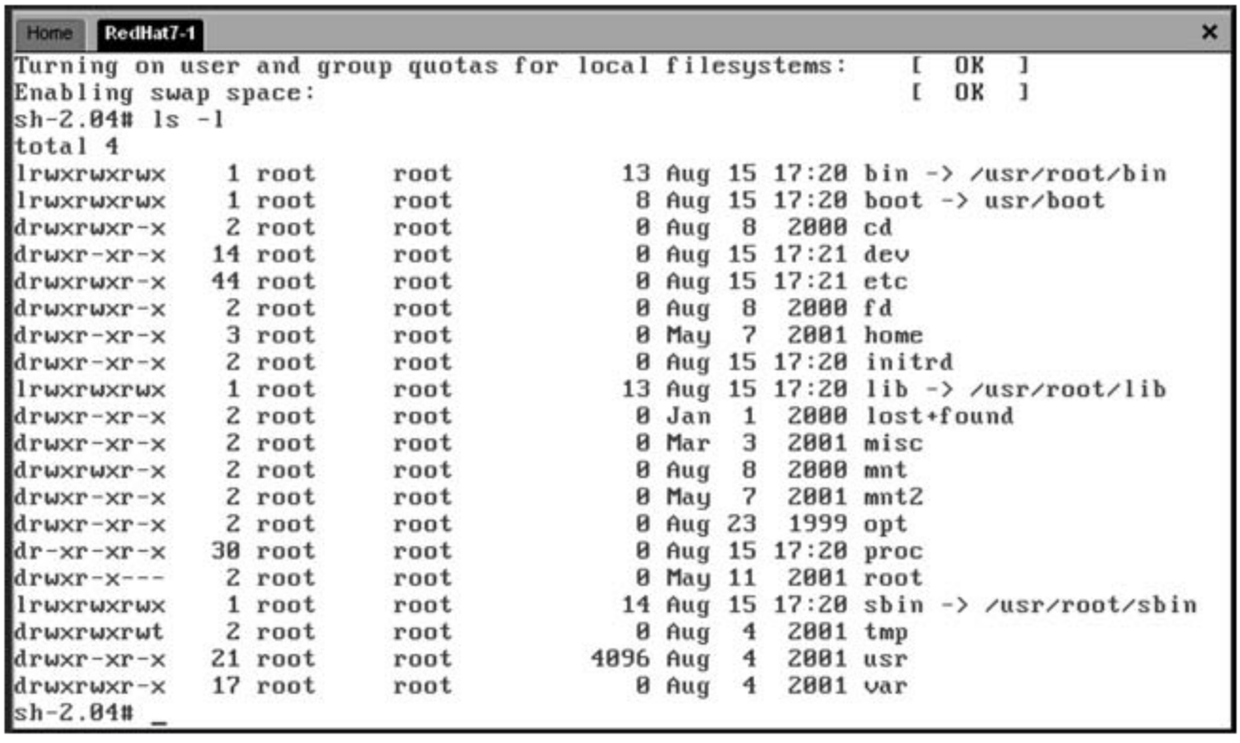


Рис. 9.32

При создании этой виртуальной машины использовались настройки, с которыми можно подробно ознакомиться при выборе функции Edit virtual machine setting в основном окне VMware. При выборе данной функции открывается окно с панелью управления виртуальной машиной (рис. 9.33).

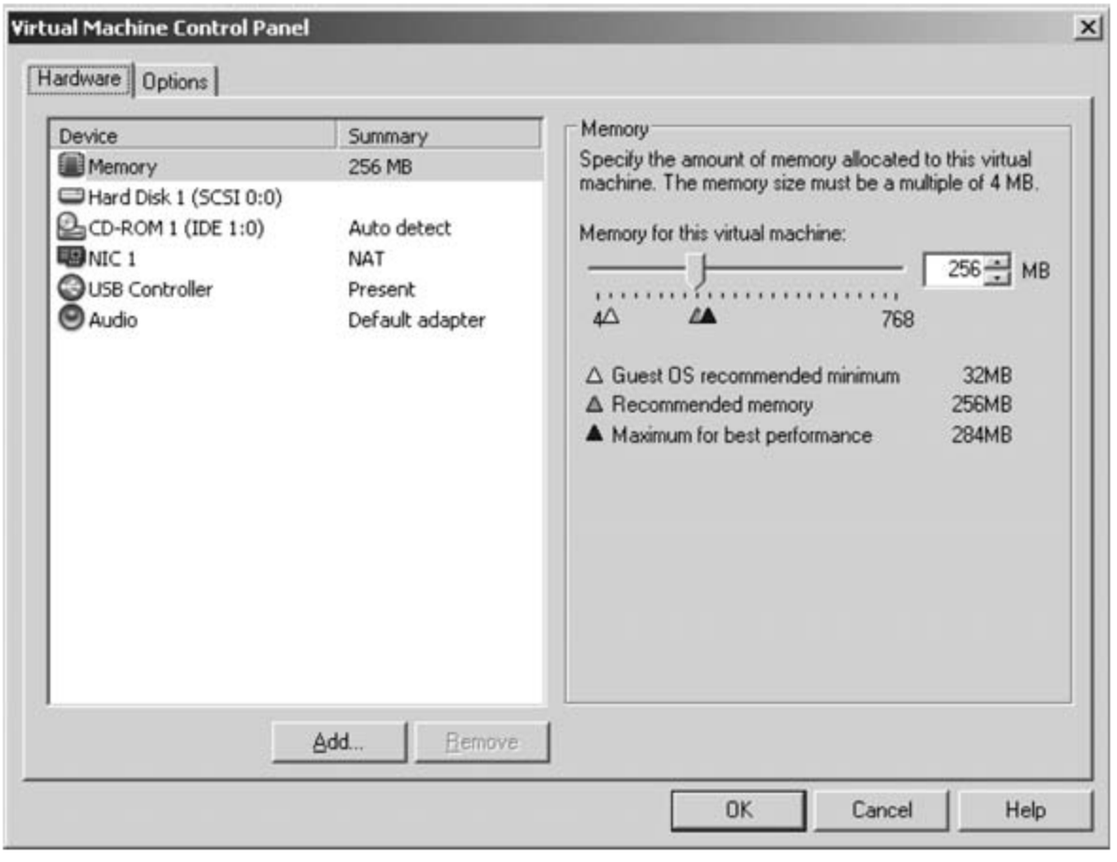


Рис. 9.33

Панель управления имеет две вкладки: Оборудование (Hardware) и Опции (Options). Выбрав вкладку Hardware, можно установить объем памяти виртуальной машины. Виртуальный жесткий диск определен как SCSI (рис. 9.34). Выделенный для данной виртуальной машины CD-ROM определен как физический диск, тип которого определяется автоматически. Поэтому ОС для загрузки этой виртуальной машины должна находиться на устройстве чтения CD-ROMa, иначе появится сообщение об отсутствии загрузочного диска.

Настройка сетевой системы определена как виртуальная сеть. USB-контроллер присутствует и может быть подключен к созданной виртуальной машине (см. рис. 9.35). Аудиосистема настроена на использование адаптера host-компьютера (рис. 9.36).

Вкладка Опции имеет несколько иной вид (рис. 9.36). В ней разрешается подстройка общих параметров, связанных с типом виртуальной машины, с возможностью сохранения свойств виртуальной машины для облегчения ее восстановления, для защиты ВМ и т.д.

Различные особенности включения и выключения ВМ представлены на рис. 9.37.

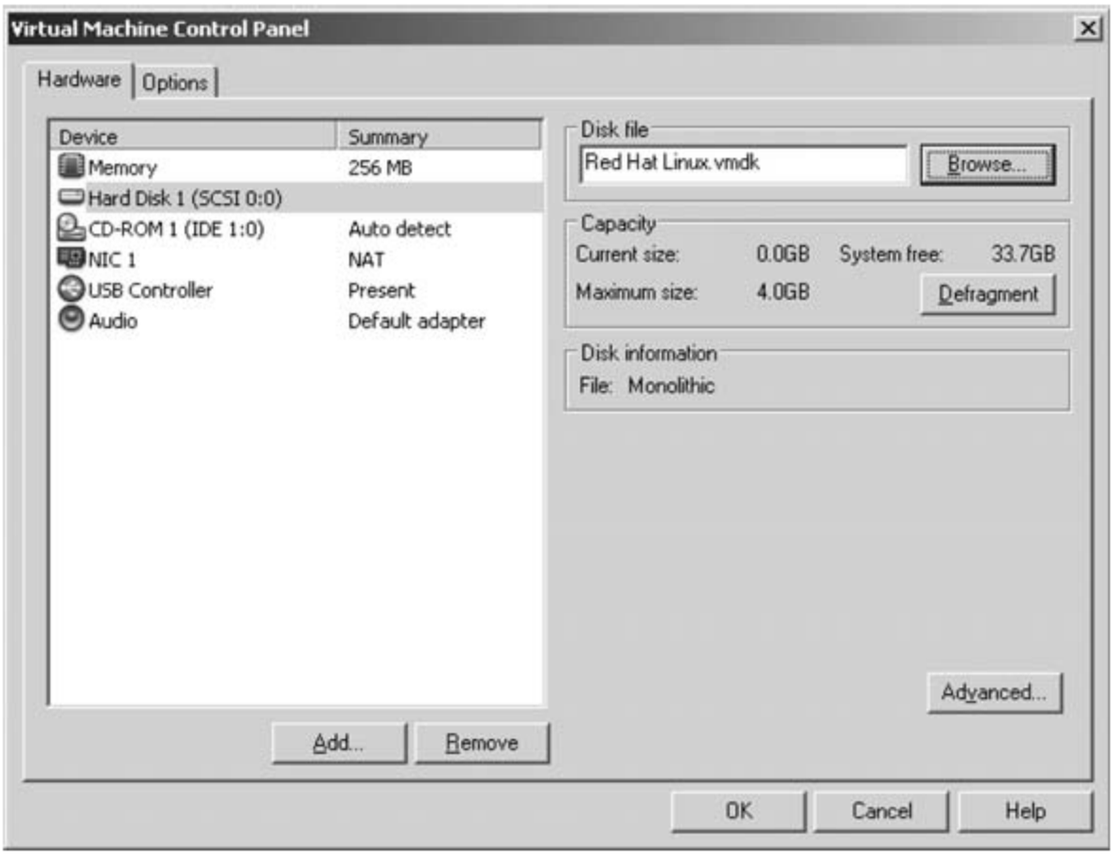


Рис. 9.34

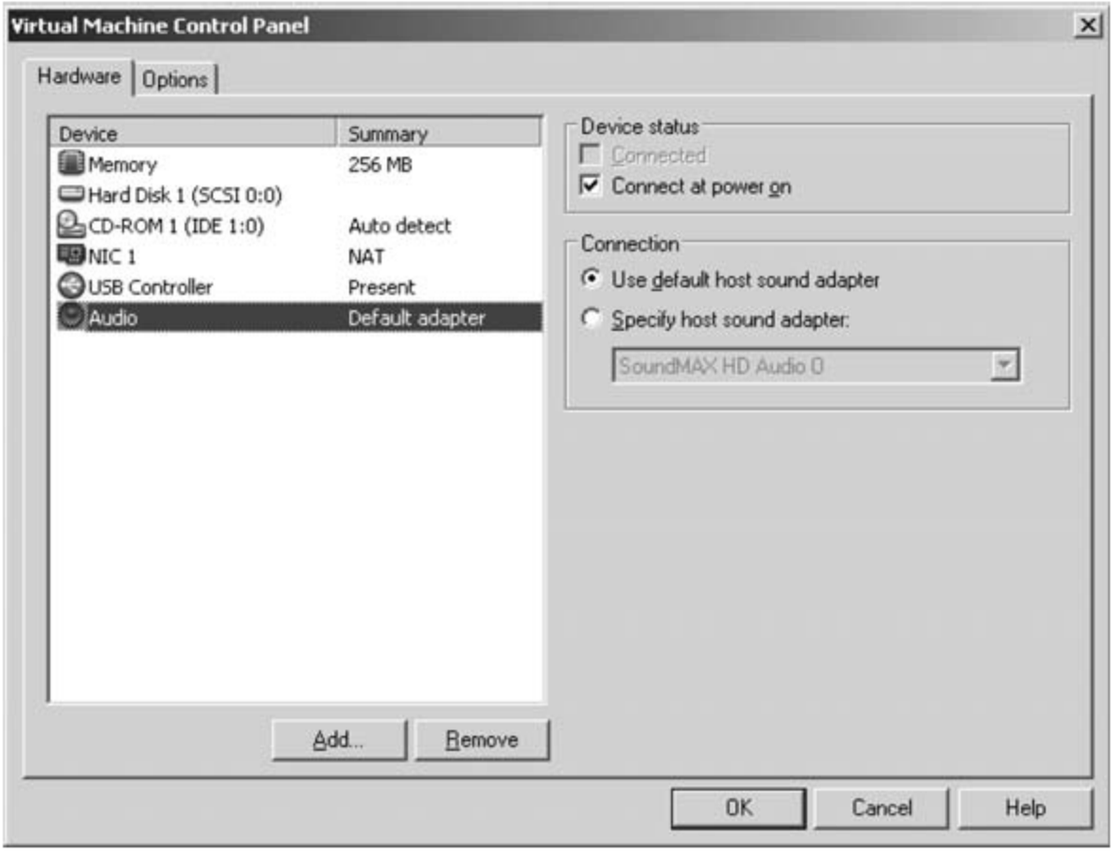


Рис. 9.35

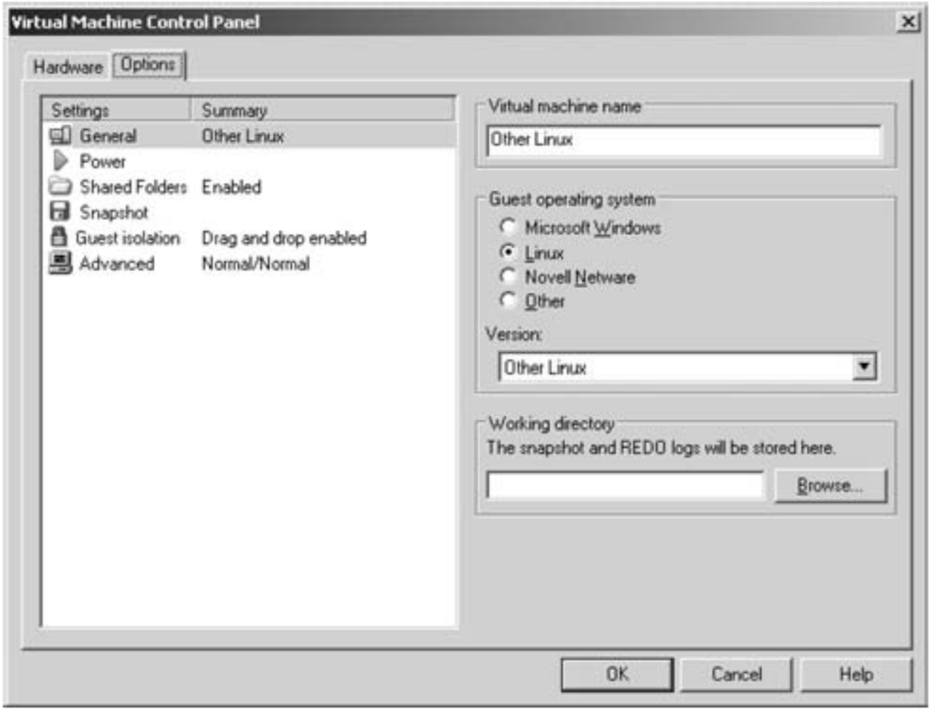


Рис. 9.36

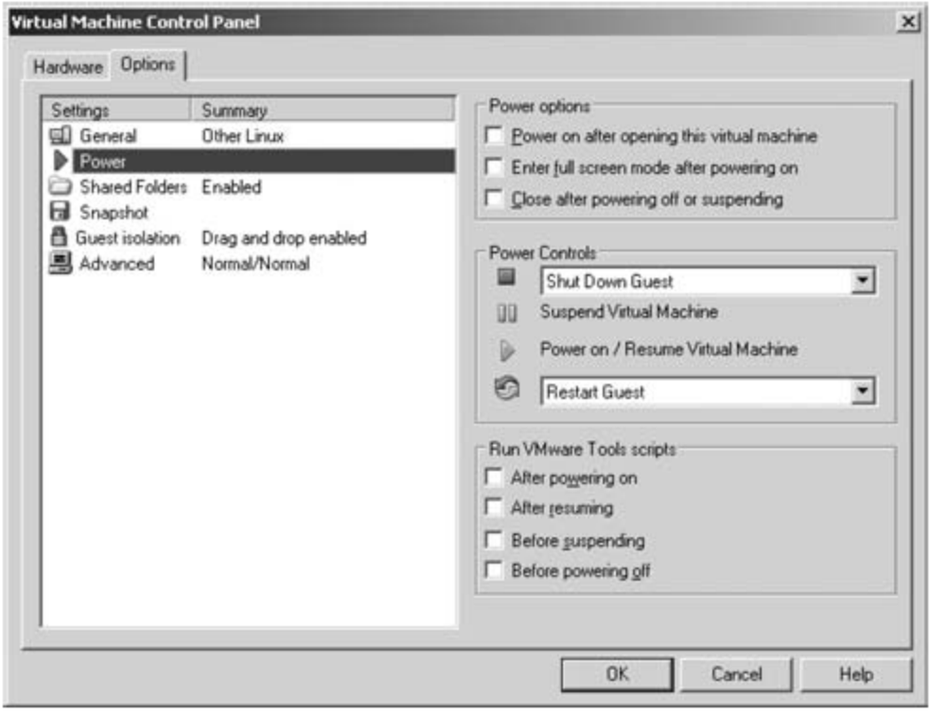


Рис. 9.37

Особое место среди этих настроек занимает Shared Folders — общие папки, доступные всем виртуальным машинам, через которые может вестись обмен информацией между разными виртуальными машинами (рис. 9.38). В данной виртуальной машине Shared Folders не определены.

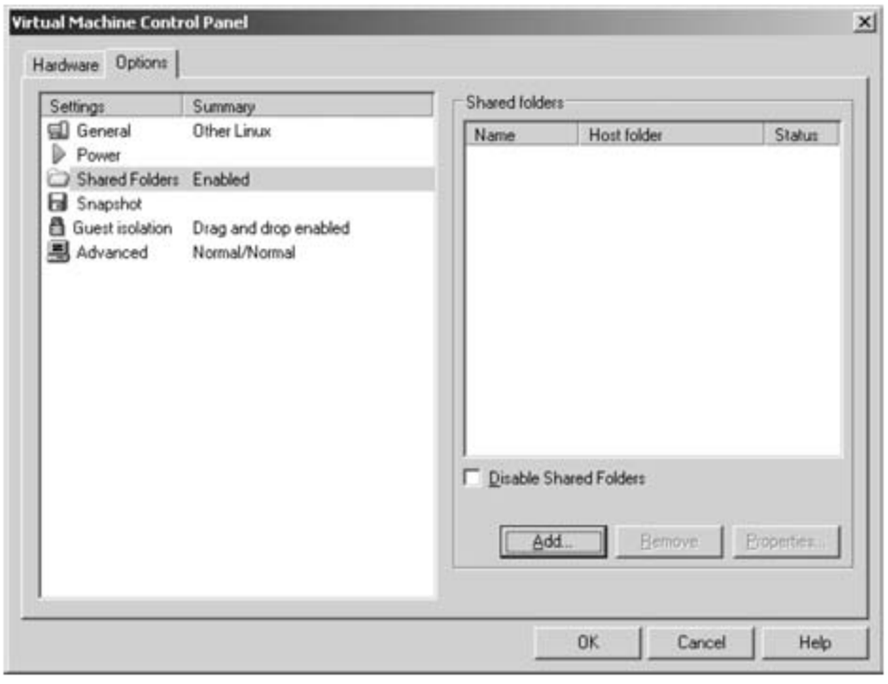


Рис. 9.38

Рассмотрим инсталляцию ОС, загрузка которой каждый раз будет происходить из iso-файла, находящегося на жестком диске host-машины. При создании виртуальной машины Debian преследовалась цель — обеспечить загрузку ОС Linux Debian без использования компакт-дисков. Для этого в конфигурации данной машины произведен отказ от использования физического диска и указан адрес образа ОС в iso-файле (рис. 9.39).

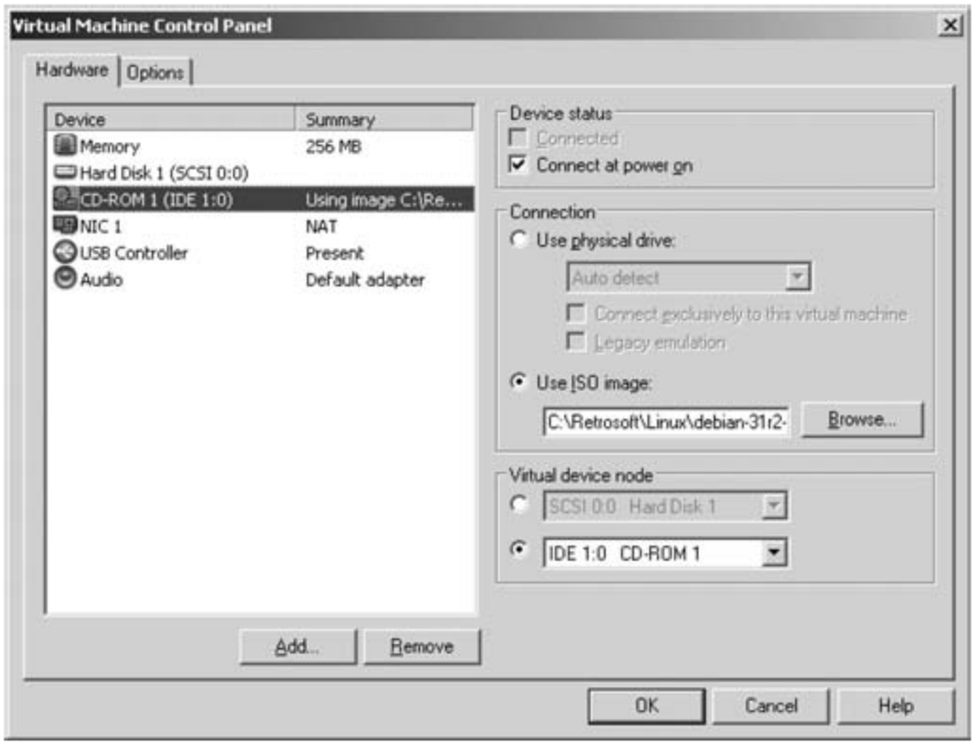


Рис. 9.39

Затем проводится создание виртуальной машины с загрузкой ОС Debian. После установки ОС основной является папка C:\Documents and Settings\AAK\Мои документы\My Virtual Machines\Debian, в которой все хранится в файлах (рис. 9.40).

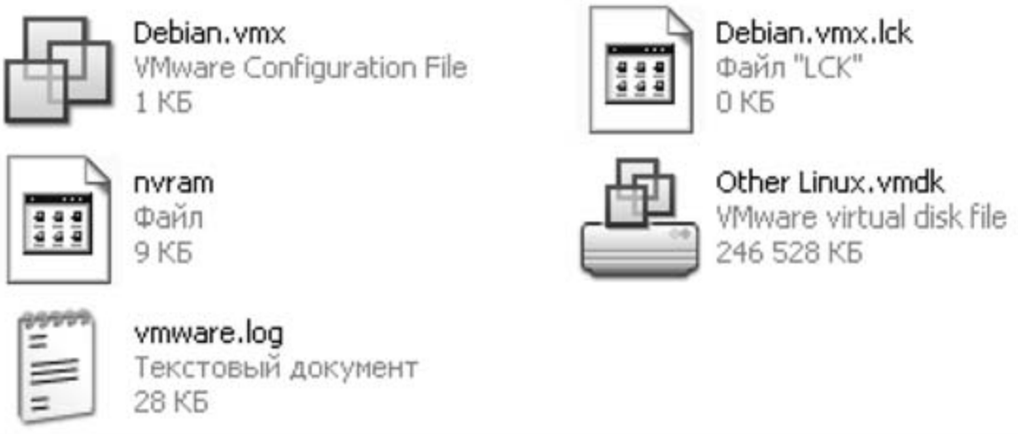


Рис. 9.40

После отключения виртуальной машины новая загрузка этой ОС осуществляется без использования CD-ROMa.

9.3.6. Работа на виртуальной машине

Для более удобной работы с VMware в виртуальной машине надо установить утилиты VMware Tools. При работе с ВМ следует иметь в виду, что клавиатура и мышь работают по-разному в окне ВМ и в окне host-компьютера. Для входа в окно виртуальной машины нужно установить курсор на черное поле окна и щелкнуть левой кнопкой мыши. Курсор мыши исчезнет, а виртуальная машина будет слушаться нажатия клавиш.

Для возвращения — выхода в окно VMware — применяется комбинация клавиш Ctrl-Alt. Если необходимо переключить виртуальную машину в полноэкранный режим работы, так чтобы не были видны границы окна VMware, надо щелкнуть по кнопке Full Screen на панели VMware. Чтобы вернуться снова в режим работы в окне, воспользуйтесь комбинацией клавиш Ctrl-Alt.

Инсталляция нового программного обеспечения внутри виртуальной машины VMware осуществляется точно так же, как и на обычном компьютере:

1) запустить виртуальную машину. На всякий случай проверить, что виртуальная машина имеет доступ к дисководу CD-ROMa

или дисководу гибких дисков (в зависимости от того, какой из них потребуется в процессе инсталляции);

2) вставить установочный диск в соответствующий дисковод и запустить программу установки.

Для выхода из любой ОС проделать обычные шаги по выходу из ОС внутри ВМ, затем выключить питание клавишей Power Off и выйти из VMware:

1) выполнить процедуру остановки (Shutdown) операционной системы, запущенной на виртуальной машине (находясь внутри виртуальной машины). Например, в Windows 95 щелкнуть по клавише Пуск (Start), выбрать пункт Выключить компьютер и щелкнуть по клавише OK;
2) после того как ОС выгрузится (окно VMware становится черным), щелкнуть по клавише Power off на панели VMware. Затем можно закрыть VMware, воспользовавшись пунктом меню File / Exit.

Список литературы

1. *Армстронг (мл.) Д.* Секреты Unix : пер. с англ. М. : ИД «Вильямс», 2001.
2. *Брюс У., Туррот П., Черникофф Д.* Microsoft Windows XP. Средства повышения производительности : пер. с англ. М. : СП ЭКОМ, 2003.
3. *Гультяев А.К.* Виртуальные машины: несколько компьютеров в одном (+ CD). СПб. : Питер, 2006.
4. *Кэррие Б.* Криминалистический анализ файловых систем : пер. с англ. СПб. : Питер, 2007.
5. *Мюллер Дж.* Оптимизация Windows XP : пер. с англ. СПб. : Питер, 2006.
6. *Мюллер Дж., Чоудри И.* Microsoft Windows 2000. Настройка и оптимизация производительности : пер. с англ. М. : ЭКОМ, 2001.
7. *Назаров С.В.* Администрирование локальных сетей Windows NT/2000/.NET. М. : Финансы и статистика, 2003.
8. *Назаров С.В.* Операционные среды, системы и оболочки. Основы структурной и функциональной организации : учеб. пособие. М. : КУДИЦ-ПРЕСС, 2007.
9. *Негус К.* Linux. Библия пользователя : пер. с англ. М. : ИД «Вильямс», 2007.
10. *Прайс Д., Гандэрлой М.* Visual C#.NET. Полное руководство : пер. с англ. Киев : ВЕК+ ; НТИ ; СПб. : КОРОНАпринт ; М. : Энтроп, 2004.
11. *Рихтер Дж.* Windows для профессионалов : пер. с англ. СПб. : Питер ; М. : ИТД «Русская редакция», 2003.
12. *Русинович М., Соломон Д.* Внутреннее устройство Mikrisoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс : пер. с англ. М. : ИТД «Русская редакция» ; СПб. : Питер, 2006.
13. *Синчак С.* Windows XP. Настройка и разгон (+ CD). СПб. : Питер, 2006.
14. *Смит Д., Наир Р.* Архитектура виртуальных машин // Открытые системы. 2005. № 5, 6.
15. *Стахнов А.А.* Сетевое администрирование Linux. СПб. : БХВ-Петербург, 2004.
16. *Таненбаум Э.* Современные операционные системы : пер. с англ. СПб. : Питер, 2002.
17. *Чекмарев А.Н., Вишневский А.В., Кокорева О.И.* Microsoft Windows Server 2003. Русская версия / под общ. ред. А.Н. Чекмарева. СПб. : БХВ-Петербург, 2005.
18. *Шалин П.А.* Реестр Windows XP спец. справ. СПб. : Питер, 2006.
19. *Шрайбер С.* Недокументированные возможности Windows 2000. : пер. с англ. СПб. : Питер, 2002.