Last updated June, 2000. Click here for updates.

Click here to see additional documents related to this study guide. Click here to receive free practice questions for LPI General Linux Part 2.

Contents

Cramsession™ for LPI General Linux Part 2

Abstract:

This Cramsession will help you to prepare for Linux Exam 102, LPI General Linux Part 2. Exam Topics include Hardware & Architecture, Linux Installation and Package Management, Kernel, Text Editing, Processing, Printing, Shells, Scripting, Programming, Compiling, X, Networking Fundamentals, Services and Security.

Notice: While every precaution has been taken in the preparation of this material, neither the author nor BrainBuzz.com assumes any liability in the event of loss or damage directly or indirectly caused by any inaccuracies or incompleteness of the material contained in this document. The information in this document is provided and distributed "as-is", without any expressed or implied warranty. Your use of the information in this document is solely at your own risk, and Brainbuzz.com cannot be held liable for any damages incurred through the use of this material. The use of product names in this work is for information purposes only, and does not constitute an endorsement by, or affiliation with BrainBuzz.com. Product names used in this work may be registered trademarks of their manufacturers. This document is protected under US and international copyright laws and is intended for individual, personal use only. For more details, visit our legal page.

© 2000 All Rights Reserved – BrainBuzz.com

brainbuzz.com

LPI General Linux Part 2

LPI Exam 102

Device Files

First Floppy	/dev/fd0
Second Floppy	/dev/fd1
First IDE drive	/dev/hda
Partitions	
First Primary	/dev/hda1
Second Primary	/dev/hda2
Third Primary	/dev/hda3
Fourth Primary	/dev/hda4
First Logical	/dev/hda5
	/dev/had....
Second IDE drive	/dev/hdb
Second IDE Primary	/dev/hdb1
First SCSI drive	/dev/sda
Second SCSI drive	/dev/sdb

- A swap partition can be a primary partition, or a logical partition, and a swap partition has no restrictions on where it needs to be placed on the hard drive.
- Swap partition size should be 1.5 to 2 times the amount of physical RAM in the computer.

LILO

In order to configure LILO you will need to edit the file/etc/lilo.conf. The following is an example of a LILO configuration file. The Linux root partition is on /dev/hda2, and Windows is installed on /dev/hdb1 (on the second hard drive).

Tell LILO the primary boot loader is on /dev/hda

boot = /dev/hda

The boot image to install

install = /boot/boot.b

Information for booting Linux

image = /vmlinuz # The kernel is in /vmlinuz

label = linux # This is the name you will see if you press “tab” at the LILO prompt

root = /dev/hda2 # /dev/hda2 is the root filesystem

The stanza for booting MS-DOS

other = /dev/hdb1 # Tell LILO about windows partition

label = msdos # name when at LILO prompt is "msdos"

table = /dev/hdb # The partition table for the second drive

- If you chose to install LILO to the MBR, LILO is responsible for either booting Linux or any other OS.
- LILO saves a backup copy of your original boot record in a file called /boot/boot.0300 (for IDE).

Tar

- Used to pack the entire contents of a directory or directories into a single file called a tarball which can then be backed up to tape or saved as a file.
- Tar preserves the entire directory organization including file ownership, permissions, links, and the directory structure.

The most commonly used tar functions are:

c - create an archive

x - extract files from an archive

t - list the contents of an archive

Additionally, there are commonly used options:

v - verbose

f *filename* - use the specified file

z - gzip/gunzip

Installing an Autoconfiguring Package

An autoconfiguring package usually has a filename like: ´foo-1.0.tar.gz' where the number is a version number. To install it, first you have to unpack the package to a directory someplace:

Use gunzip command to uncompress the package:

% gunzip foo-1.0.tar.gz

or

% tar –zxvf foo-1.0.tar.gz

If you used gunzip you will now have to use the tar command to extract the package archive:

% tar xf foo-1.0.tar

Now change into the foo directory and look for files like ´README' or ´INSTALL'

The README or INSTALL files will explain how to configure the package, but the general format is as follows:

% cd foo-1.0

% ./configure

% make

% make install

% su

make install

./configure invokes a shell script that is distributed with the package that configures the package for you automatically. It will probe your system through a set of tests that allow it to automatically generate a ´Makefile' from a template stored in a file called ´Makefile.in'. To install your software, you need to explicitly invoke ´make' again with the **target** ´install'

Placement of files during install:

Executables	/usr/local/bin
Libraries	/usr/local/lib
Header files	/usr/local/include
Man pages	/usr/local/man/
Info files	/usr/local/info

if you want to install the package to your home directory instead of /usr/local, you would use the 'prefix' option:

% configure --prefix=/home/foo

Managing Shared Libraries

Most of the programs on your system will be compiled to use shared libraries so each program doesn't have to have its own independent set of commonly used functions.

ldd is a command that will let you list the shared libraries for a certain executable.

ldd /usr/bin/X11/xterm

Programs will execute ld.so which is the component responsible for finding the shared library.

A file called /etc/ld.so.conf will contain a list of directories that ld.so searches to find the shared library files. (although ld.so will ALWAYS look in /usr/lib and /lib no matter what /etc/ld.so.conf contains).

ldconfig is a command that will examine the libraries in the directories specified, as well as in /etc/ld.so.conf, /usr/lib, and /lib in order to update the links and cache where necessary.

dpkg

The general format of a Debian package file (.deb) is:

packagename_packageversion-debversion.deb

Command Function:
-I Queries Package
-i Installs software
-l Lists installed software (equivalent to rpm -qa)
-r Removes the software from the system

Command Example	Function
dpkg -i package.deb	Installs package.deb
dpkg -I package.deb	Lists info about package.deb (rpm -qpi)
dpkg -c package.deb	Lists all files in package.deb (rpm -qpl)
dpkg –l	Shows all installed packages
dpkg -r package-name	Removes 'package-name' from the system (as listed by dpkg-l)

The priority of a package indicates how essential or necessary it is. Debian GNU/Linux classifies all packages into four different priority levels:

- **Required**
  - Packages must be installed for the system to operate correctly and have been installed as part of the base system.
- **Important**
  - Important packages are found on almost all UNIX-like operating systems.

- **Standard**
  - Packages that comprise the "standard,'' character based, Debian GNU/Linux system. The Standard system includes a fairly complete software development environment and GNU Emacs.
- **Optional**
  - Optional packages comprise a fairly complete system. The Optional system includes TeX and the X Window System.
- **Extra**
  - Extra packages are only useful to a small or select group of people, or are installed for a specific purpose. Extra packages might include such programs as electronics and ham radio applications.
  - Extra packages are abbreviated in dselect as Xtr.

dselect

Simple, menu-driven interface which helps install packages. It takes you through the package installation process in the order of the on-screen menu:

- **Access**

In this menu you choose the method to obtain and install the packages.

- **Update**

dselect reads the Packages database(described above) and creates a database of the packages available on your system.

- **Select**

Choose your the package you want and press Enter. To exit the Select screen after all of the selections are complete, press Enter. This returns you to the main screen if there are no problems with your selection. You must resolve those problems first. When you are satisfied with any given screen, press Enter.

Dependency conflicts are quite normal and to be expected.

- **Install**

dselect runs through the entire 800 packages and installs the ones that are selected.

Alien

http://kitenet.net/programs/alien

This is a program, which converts between package formats. From kitenet.net:

"A program that converts between the rpm, dpkg,stampede slp, and slackware tgz file formats. You can use alien to convert it to your preferred package format and install it.

Alien should not be used to replace important system packages, like sysvinit, shared libraries, or other things that are essential for the functioning of your system. "

RPM

The following is some great RPM documentation that was generously provided by Craig Kulesa. For more information, refer to his site here. Thanks Craig!

What's in an RPM package anyway?

The basic idea of package management is to group the tens of thousands of installed files on your computer into a manageable set of *packages,* which allow the administrator to easily install and uninstall them from the computer.

RPM, however, extends this notion substantially. The biggest advantage of an advanced package system like RPM is that each package contains the *knowledge* of what it takes to install itself on your computer. It contains information on what the package does, what other packages it may depend upon, what capabilities the package brings to your system, and where the files should go on your computer. Your operating system keeps a database of every package installed on your computer and its capabilities and dependencies.

Invoking RPM

Before showing you graphical "point-and-click" tools for managing packages, it's instructive to look at the **RPM** command itself. It's amazingly flexible and powerful! Documentation for RPM is very thorough --the man page is terse but a useful reference. The web sitehttp://www.rpm.org/ is a treasure-trove of HOW-TO's, and includes the excellent book *Maximum RPM*, which is perhaps the quintessential reference on RPM itself. It is mandatory reading for anyone wanting to build their own RPM packages.

RPM has a fairly simple syntax, but has many modifying options to allow you to tell it to do nearly anything you want. Rather than discuss the features at length, I'll mention the basic types of operations it can perform and highlight many handy examples you can use in your own arsenal.

Commonly-used RPM "Basic Modes"

-b = Build New Package

-q = Query information from installed or uninstalled packages

-U = Install/upgrade packages to the system

-F = Freshen packages existing on the system

-e = Uninstall packages from the system

-V = Verify the validity of packages on the system, or an uninstalled package

Now let's look at ways to invoke these modes in really cool ways!

Installing and Uninstalling Packages

Question	RPM Command
How do I install a new package, or upgrade it if it already exists?	**rpm -Uvh package.rpm**
If I have a directory containing updated packages, how do I upgrade *only* the ones currently on my system?	**rpm -Fvh *.rpm**
How do I delete an installed package from my computer?	**rpm -e package**

Querying Info from Packages

Question	RPM Command
How do I see a list of all installed packages on my computer?	**rpm -qa | less**
I don't know what a certain installed package does. How can I get it to tell me about itself?	**rpm -qi package**
How do I ask what files were installed by a certain installed package?	**rpm -ql package**
How do I get a-yet-uninstalled package to give me information about itself and the files it would install on my computer?	**rpm -qilp package.rpm**
There's a file on my computer called /usr/bin/weirdo. How can I find out which installed package it belongs to?	**rpm -qf /usr/bin/weirdo**
How do I find out which package installed /usr/bin/weirdo, AND how do I get information on that package and see all the other files it installed?	**rpm -qilf /usr/bin/weirdo**

Verifying Packages

Question	RPM Command
I deleted a few files by accident, but I don't know what they are. Can rpm show me which files in its database are now missing?	**rpm -Va**
I think I've been hacked! How do I check for files that have been modified or removed in any way?	**rpm -Va**
'rpm -Va' takes a long time. How do I just verify that a certain package is OK?	**rpm -Vv package**
How do I test the integrity of a yet-uninstalled package file?	**rpm -K --nopgp package.rpm**

Building Source Packages

Making your own RPM is beyond the scope of this quick reference(see the *Maximum RPM* book for that), but it's not unusual to want to build from RPM's that only contain source code, not binaries. This is especially relevant for non-Intel architectures like PowerPC, Alpha, and Sparc which may not have platform-specific RPM's available, only the source code RPM's.

Question	RPM Command
I've downloaded an RPM package containing source code. How do I make a binary RPM out of it, and then install it?	**Rpm –rebuild package.src.rpm**, then **rpm –Uvh package.rpm**

Miscellaneous

Question	RPM Command
How do I compress and rebuild my computer's RPM database?	**Rpm –rebuilddb**

Modules

Module

A file containing all of the code for a specific driver

depmod

Creates a dependency file for modules which can be used my modprobe

You normally run:

depmod –a

This will create dependencies for all modules listed in/etc/conf.modules

insmod

Loads the module into the kernel

lsmod

List all loaded modules, as well as their size (in units of4k)

rmmod

Unload a module or list of modules from the kernel (only if they are not in use at the time of removal)

rmmod –r will recursively remove the modules

modprobe

Will attempt to load the specified module as well as other modules it depends upon. If you list more than one module it will try the others only if the first one fails.

modprobe –a Will load all of the modules instead of just the first one

modprobe –r Will remove specified modules as well as their dependencies

Building a Kernel

Run either "make config", "make xconfig"(if you are running x and have tcl/Tk), or "make menuconfig" (if you have the curses library)

The above menu methods will ask you which components you wish to have as part of the kernel, or available as modules.

make dep

Gathers the dependencies for each source file and includes them in the various makefiles

make clean (if you have built a kernel from this source before)

make zImage (this will take awhile)

This builds the actual kernel

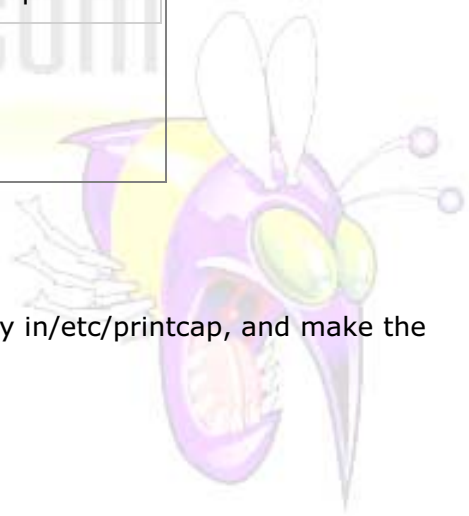
You should now have a kernel in /usr/src/linux/arch/i386/boot

Text editing (vi), Processing, Printing

Input mode		**Screen movement**:	
i	insert text at cursor	ctrl-d	scroll forward half a screen
a	append text after cursor	ctrl-u	scroll back half a screen
I	insert text at beginning of line	ctrl-f	move to next screen
A	append text at end of line	ctrl-b	move to previous screen
o	open line below cursor	ctrl-l	redraw screen
O	open line above cursor		
Deletion and Change:		**Last line mode**:	
dw	delete a single word	:w	write buffer
cw	change a word	:q	Quit
dd	delete entire line	:q!	force quit without saving
R	replace line	:wq	write and quit (save)
D	delete from cursor position to end of line	:n	next file
C	change from cursor position to end of line	:r	read file
x	delete a single character	:e	edit file
r	replace a character	:f	file name
		:set	set options on
		:set no	set options off
		:!	escape to shell
		:n	go to line n

brainbuzz.com

Cursor movement:		**Miscellaneous**	
l	move right	u	undo last action
h	move left	/	search forward
j	move down	?	search back
k	move up	n	repeat search
$	move to end of line	.	repeat last command
^	move to beginning of line	yy	yank line
w	move to next word	p	put below cursor
e	move to end of word	P	put above cursor
1G	move to first line	ZZ	write and quit
nG	move to line n (where n is a number)		
G	move to last line		



Printing

To add a print queue to lpd, you must add an entry in/etc/printcap, and make the new spool directory under /var/spool/lpd.

An entry in /etc/printcap looks like:

LOCAL djet500

lp|dj|deskjet:\

:sd=/var/spool/lpd/dj:\

:mx#0:\

:lp=/dev/lp0:\

:sh:

This above printcap defines a spool called lp, dj, or deskjet, spooled in the directory /var/spool/lpd/dj, with no per-job maximum size limit, which prints to the device /dev/lp0, and which does not have a banner page (with the name of the person who printed, etc) added to the front of the print job.

LPD

Lpd is the line printer daemon (spool area handler) and is normally invoked at boot time from the arc(8) file. It makes a single pass through the printcap(5) file to find out about the existing printers and prints any files left after a crash. It then uses the system calls listen(2) and accept(2) to receive requests to print files in the queue, transfer files to the spooling area, display the queue, or remove jobs from the queue. In each case, it forks a child to handle the request so the parent can continue to listen for more requests.

LPR

Lpr uses a spooling daemon to print the named files when facilities become available. If no names appear, the standard input is assumed.

-l

Use a filter which allows control characters to be printed and suppresses page breaks.

-P

Force output to a specific printer. Normally, the default printer is used (site dependent), or the value of the environment variable PRINTER is used.

-h

Suppress the printing of the burst page.

-m

Send mail upon completion.

-r

Remove the file upon completion of spooling or upon completion of printing (with the -s option).

-#num

The quantity num is the number of copies desired of each file named.

LPQ

Reports the status of the specified jobs or all jobs associated with a user. Lpq, invoked without any arguments, reports on any jobs currently in the queue.

Options:

-P

Specify a particular printer, otherwise the default line printer is used (or the value of the PRINTER variable in the environment). All other arguments supplied are interpreted as user names or job numbers to filter out only those jobs of interest.

If lpq warns that there is no daemon present (i.e., due to some malfunction), the lpc command can be used to restart the printer daemon.

LPC

Used by the system administrator to control the operation of the line printer system. For each line printer configured in /etc/printcap, lpc maybe used to:

- Disable or enable a printer
- Disable or enable a printer's spooling queue
- Rearrange the order of jobs in a spooling queue
- find the status of printers, and their associated spooling queues and printer daemons.
    - Without any arguments, lpc will prompt for commands from the standard input
    - If arguments are supplied, lpc interprets the first argument as a command and the remaining arguments as parameters to the command
    - The standard input may be redirected causing lpc to read commands from file.

LPRM

- lprm will remove a job, or jobs, from a printer's spool queue.
- usinglprm is normally the only method by which a user may remove a job.

Options and arguments:

-Pprinter

Specify the queue associated with a specific printer (otherwise the default printer is used)

-

If a single ´-' is given, lprm will remove all jobs which a user owns. (note: If root employs this flag, the spool queue will be emptied entirely)

user

Causes lprm to attempt to remove any jobs queued belonging to that user

job #

Allows a user to remove a job from the queue by job #

Some Important Files Used in Printing

/etc/printcap

printer description file

/var/spool/*

spool directories /var/spool/*/minfree minimum free space toleave

/dev/lp*

line printer devices

/dev/printer

socket for local requests

/etc/hosts.equiv

lists machine names allowed printer access

/etc/hosts.lpd

lists machine names allowed printer access, but not under same administrative control.

/etc/passwd

personal identification.

/etc/printcap

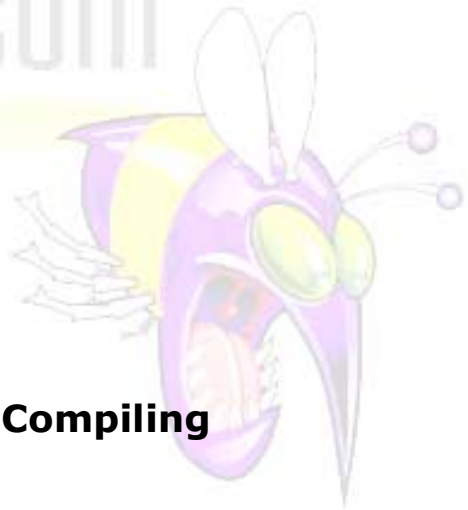
printer capabilities data base.

/usr/sbin/lpd*

line printer daemons.

/var/spool/output/*

directories used for spooling.



Shells, Scripting, Programming, Compiling

Bash Shell

/bin/bash

The bash executable

/etc/profile

The system wide initialization file, executed automatically at **login**

~/.bash_profile

The personal initialization file, executed automatically at **login**

~/.bashrc

The individual shell startup file, executed automatically at **shell startup**

~/.bash_logout

The individual logout file, executed automatically at **logout**

~/bash_history

Records session commands

~/.inputrc

Individual readline initialization file

csh and tcsh Shell

~/.cshrc

Executed automatically at shell startup.

~/.login

Executed automatically at login after .cshrc has been processed

~/.logout

Executed automatically at shell logout.

Creating Shell Scripts

Note: The following shell programming documentation is from an excellent **intro** by Andrew Arensburger and has been printed with his permission. Thanks Andrew!

The first line of any script must begin with #!, followed by the name of the interpreter.

Some versions of Unix allow white space between #! and the name of the interpreter. Others don't. Hence, if you want your script to be portable, leave out the blank.

A script, like any file that can be run as a command, needs to be executable: save this script as rotate log and run

chmod +x rotate log

to make it executable. You can now run it by running

./rotatelog

Environment commands

set - shows all variables (local and exported)

env - shows only exported variables

export - shows exported variables

unset - used to remove variables

Variables

sh allows you to have variables, just like any programming languages. Variables do not need to be declared. To set a sh variable, use

VAR=value

and to use the value of the variable later, use

$VAR

or

${VAR}

The latter syntax is useful if the variable name is immediately followed by other text:

```
#!/bin/sh
COLOR=yellow
echo This looks $COLORish
echo This seems ${COLOR}ish
```

prints

```
This looks
This seems yellowish
```

Variable Substitution

${VAR:-expression}

Use default value: if VAR is set and non-null, expands to $VAR. Otherwise, expands to expression.

${VAR:=expression}

Set default value: if VAR is set and non-null, expands to $VAR. Otherwise, sets VAR to expression and expands to expression.

${VAR:?[expression]}

If VAR is set and non-null, expands to $VAR. Otherwise, prints expression to standard error and exits with a non-zero exit status.

${VAR:+expression}

If VAR is set and non-null, expands to the empty string. Otherwise, expands to expression.

${#VAR}

Expands to the length of $VAR.

Flow control

sh supports several flow-control constructs, which add power and flexibility to your scripts.

if

The if statement is a simple conditional. You've seen it in every programming language. Its syntax is

```
if condition ; then
commands
[elif condition ; then
commands]...
[else
commands]
```

fi

That is, an if-block, optionally followed by one or more elif-blocks (elif is short for ´´else if''), optionally followed by anelse-block, and terminated by fi.

The if statement pretty much does what you'd expect: if condition is true, it executes the if-block. Otherwise, it executes the else-block, if there is one. The elif construct is just syntactic sugar, to let you avoid nesting multiple if statements.

#!/bin/sh

myname=´whoami´

if [$myname = root]; then

echo "Welcome to FooSoft 3.0"

else

echo "You must be root to run this script"

exit 1

fi

The more observant among you (or those who are math majors) are thinking, ´´Hey! You forgot to include the square brackets in the syntax definition!''

Actually, I didn't: [is actually a command, /bin/[, and is another name for the test command. This is why you shouldn't call a test program test: if you have ´´.'' at the end of your path, as you should, executing test will run /bin/test.

The condition can actually be any command. If it returns a zero exit status, the condition is true; otherwise, it is false. Thus, you can write things like

#!/bin/sh

user=arnie

if grep $user /etc/passwd; then

echo "$user has an account"

else

echo "$user doesn't have an account"

fi

while

The while statement should also be familiar to you from any number of other programming languages. Its syntax in sh is

while condition; do

commands

done

As you might expect, the while loop executes commands as long as condition is true. Again, condition can be any command, and is true if the command exits with a zero exit status.

A while loop may contain two special commands: break and continue.

break exits the while loop immediately, jumping to the next statement after the done.

continue skips the rest of the body of the loop, and jumps back to the top, to where condition is evaluated.

for

for var in list; do The for loop iterates over all of the elements in a list. Its syntax is

commands

done

list is zero or more words. The for construct will assign the variable var to each word in turn, then execute commands. For example:

#!/bin/sh

for i in foo bar baz "do be do"; do

echo "$i"

done

will print

foo

bar

baz

do be do

A for loop may also contain break and continue statements. They work the same way as in the while loop.

case

The case construct works like C's switch statement, except that it matches patterns instead of numerical values. Its syntax is

case expression in

pattern)

commands

;;

...

esac

expression is a string; this is generally either a variable or aback quoted command.

pattern is a glob pattern (see globbing).

The patterns are evaluated in the order in which they are seen, and only the first pattern that matches will be executed. Often, you'll want to include a ´´none of the above'' clause; to do this, use * as your last pattern.

The X Display Manager

xdm

can be configured with configuration files located in/usr/X11R6/lib/X11/xdm on your Linux system.

xdm-config

File which configures how the login screen appears to users, andXsetup_0 is used to tell xdm what programs should be launched when X is started.

Every X Server has a display name of the form:

- hostname:displaynumber.screennumber

  hostname

  Specifies the name of the machine to which the display is actually, physically connected (If host isn't specified local host is used)

  Display number

  used if the X Server is controlling more than one keyboard and monitor unit, for instance, a network of X terminals.

  Screen number

  specifies which monitor in a multiple monitor setup should be used

XF86Setup

- provided by the Xfree86 team
- starts a VGA X server with 16 colors which allows you to then select your video card, monitor, and mouse types

Config XF86

- Text based setup program which will generate a config file from your answers
- Good to use if XF86Setup can't be run for some reason

Xconfigurator

- Provided by RedHat

.Xdefaults

Can contain every option you prefer for every type of program you run in X

Troubleshooting X

x –showconifg

Displays the chipset names known to your X server

x –probeonly

Have the X server probe your chipset for you

Starting X

startx actually calls xinit which is the program responsible for stating the X server and clients

Port Assignments: (only a partial list of most popular)

Keyword Decimal Description

chargen 19/tcp Character Generator

chargen 19/udp Character Generator

ftp-data 20/tcp File Transfer [Default Data]

ftp-data 20/udp File Transfer [Default Data]

ftp 21/tcp File Transfer [Control]

ftp 21/udp File Transfer [Control]

ssh 22/tcp SSH Remote Login Protocol

ssh 22/udp SSH Remote Login Protocol

telnet 23/tcp Telnet

telnet 23/udp Telnet

domain 53/tcp Domain Name Server

domain 53/udp Domain Name Server

bootps 67/tcp Bootstrap Protocol Server

bootps 67/udp Bootstrap Protocol Server

bootpc 68/tcp Bootstrap Protocol Client

bootpc 68/udp Bootstrap Protocol Client

tftp 69/tcp Trivial File Transfer

tftp 69/udp Trivial File Transfer

gopher 70/tcp Gopher

gopher 70/udp Gopher

finger 79/tcp Finger

finger 79/udp Finger

http 80/tcp World Wide Web HTTP

http 80/udp World Wide Web HTTP

kerberos 88/tcp Kerberos

kerberos 88/udp Kerberos
rtelnet 107/tcp Remote Telnet Service
rtelnet 107/udp Remote Telnet Service
pop3 110/tcp Post Office Protocol - Version 3
pop3 110/udp Post Office Protocol - Version 3
sunrpc 111/tcp SUN Remote Procedure Call
sunrpc 111/udp SUN Remote Procedure Call
ident 113/tcp
auth 113/tcp Authentication Service
auth 113/udp Authentication Service
nntp 119/tcp Network News Transfer Protocol
nntp 119/udp Network News Transfer Protocol
imap 143/tcp Internet Message Access Protocol
imap 143/udp Internet Message Access Protocol
snmp 161/tcp SNMP
snmp 161/udp SNMP
snmptrap 162/tcp SNMPTRAP
snmptrap 162/udp SNMPTRAP

Subnet Mask table

# of Bits	Dotted Decimal	Hexadecimal	# of Hosts
16	255.255.0.0	FF-FF-00-00	65534
17	255.255.128.0	FF-F8-00-00	32766
18	255.255.192.0	FF-FF-C0-00	16382
19	255.255.224.0	FF-FF-E0-00	8190
20	255.255.240.0	FF-FF-F0-00	4094
21	255.255.248.0	FF-FF-F8-00	2046
22	255.255.252.0	FF-FF-FC-00	1022
23	255.255.254.0	FF-FF-FE-00	510
24	255.255.255.0	FF-FF-FF-00	254
25	255.255.255.128	FF-FF-FF-80	126
26	255.255.255.192	FF-FF-FF-C0	62

27	255.255.255.224	FF-FF-FF-E0	30
28	255.255.255.240	FF-FF-FF-F0	14
29	255.255.255.248	FF-FF-FF-F8	6
30	255.255.255.252	FF-FF-FF-FC	2

ifconfig command

up

this option activates an interface (and is the default).

down

this option deactivates an interface.

[-]arp

this option enables or disables use of the address resolution protocol on this interface

netmask <addr>

this parameter allows you to set the network mask of the network this device belongs to.

Networking Configuration Files

/etc/resolv.conf

Main configuration file for name resolution. There are three keywords typically used. They are:

domain

this keyword specifies the local domain name

search

this keyword specifies a list of alternate domain names to search for a hostname

nameserver

You can list either a single name server or multiple name servers

- An example /etc/resolv.conf might look something like:

domain somecollege.edu.ca

search science.somecollege.edu.au

nameserver 192.168.1.200

nameserver 192.168.1.201

/etc/host.conf

Governs the order in which to resolve names. For example the following host. conf file would check the hosts file first and then query dns.The "multi on" entry allows you to have multiple IP addresses for your machine.

order hosts,bind

multi on

/etc/hosts

Contains host name to IP address mappings. If you configure some entries in the hosts file and you have the order of your host.conf file checking hosts first, you will try to resolve any hostnames by checking the hosts file first and if the entry is not found you will then resort to querying a dns server.

/etc/hosts

127.0.0.1 localhost l

www.somedomain.com

/etc/networks

Allows you to refer to dotted decimal IP's by a name rather than IP. This is handy when you are using the route command because you can refer to networks such as "localnet" instead of 192.168.1.0.

/etc/services

The /etc/services file is a simple database that associates a human friendly name to a machine friendly service port. Each entry is comprised of three fields separated by any number of white space (tab or space) characters. The fields are:

name port/protocol aliases # comment

name

a single word name that represents the service being described.

port/protocol

this field is split into two subfields.

port

a number that specifies the port number the named service will be available on. Most of the common services have assigned service numbers. These are described in RFC-1340.

protocol

may be set to either tcp or udp. It is important to note that an entry of 18/tcp is very different from an entry of 18/udp and that there is no technical reason why the same service needs to exist on both. Normally commonsense prevails and it is only if a particular service is available via both tcp and udp that you will see an entry for both.

aliases

other names that may be used to refer to this service entry.

Networking Services

/etc/inetd.conf

Responsible for starting services, typically ones that do not need to run continuously, or are session based (such as telnet, or ftpd).

Whenever you make changes to inetd.conf you must restart inetd to make the changes effective.

ps –aux|grep inetd

(now note the process number)

kill –HUP <process id of inetd>

Lines in inetd.conf can be commented out with a #

Access to programs started by inetd can be easily controlled by the use of TCP_WRAPPERS.

TCP_WRAPPERS

TCP_WRAPPERS is controlled from two files:

/etc/hosts.allow

/etc/hosts.deny

hosts.allow is checked first, and the rules are checked from first to last. If it fails to find a rule denying you entry it then by default lets you. A good idea is to make the last rule in your /etc/hosts.deny as follows:

ALL: 0.0.0.0/0.0.0.0

This means all services, all locations, so any service not explicitly allowed is then blocked (don't forget that the default is to allow).

Sendmail

You can use sendmail from either the command line, or through a front end mail reader.

Sendmail uses three MAIN configuration files: (there are many more)

Sendmail.cw

This file is used to store the names of hosts which will receive email

Sendmail.cf

This is a very complex configuration file for Sendmail.

***.mc**

This is used to help the administrator configure sendmail so hopefully you won't have to create your own sendmail.cf

Apache

Basic configuration files:

httpd.conf

Contains entire server processing related commands.

arm.conf

Contains server resource information

access.conf

Per directory configuration

All of these files are found in /etc/httpd/apache/conf or/etc/httpd/conf

httpd

-C "directive" Process directive BEFORE reading config files

-c "directive" Process directive AFTER reading configuration files

-d directory Alternate server root directory

-f file Config file

-l List compile in modules

The DNS Database Files

In order to get DNS running you are going to need entries in some text files that will map the host names to ip addresses and vice versa. These files are typically referred to as db files and contain your resource records. Most people will name their files db.yourdomain (which would hold all entries for yourdomain.com), and db.192.168.1 for your reverse lookup zone. The other db files you might need are: db.cache and db.127.0.0. The db.cache file tells your server where the servers for the "." (root) domain are located so you can resolve queries for which you are not authoritative. Thedb.127.0.0 entry is for your loopback address. You will also need a boot filewhich will contain the information on where each of your configuration files are located.

This file will basically tie all of the db files together. Here is asample /etc/named.conf file:

```
Directory "/var/named"
Zone "." {
Type hint
File "db.cache";
};
zone "yourdomain.com" {
type master;
file "db.yourdomain.com";
};
zone "1.168.192.in-addr.arpa" {
type master;
file "db.192.168.1";
};
zone "0.0.127.in-addr.arpa" {
type master;
file "db.127.0.0"'
};
```

Resource record representations in master files share a common format, which is

```
[domain] [ttl] [class] type rdata
```

Fields are separated by spaces or tabs. An entry may be continued across several lines if an opening brace occurs before the first newline, and the last field is followed by a closing brace. Anything between a semicolon and a newline is ignored.

domain

This is the domain name to which the entry applies. If no domain name is given, the RR is assumed to apply to the domain of the previous RR.

SOA

The Start Of Authority record signals that the records following the SOA RR contain authoritative information for the domain.

serial

This is the version number of the zone file, expressed as a single decimal number. Whenever data is changed in the zone file, this number should be incremented. This is how secondary name servers are aware of changes to the zone.

refresh

Interval for how often the secondary should check with the primary nameserver for zone changes

retry

Determines the interval during which a secondary server should retry contacting the primary server if a request or a zone refresh fails

expire

Specifies when a secondary name server should discard its copy of the zone database because the primary nameserver could not be contacted. This is to ensure the copy of the zone database does not contain invalid data.

minimum

This is the default ttl value for resource records that do not explicitly specify one. Other name servers should discard the RR after this time

A

This associates an IP address with a hostname. Each host should only have one A record. All other hostnames are aliases and must be mapped onto the canonical hostname using a CNAME record

NS

This points to a master name server of a subordinate zone

CNAME

This associates an alias for a host with its canonical hostname. The canonical hostname is the one the master file provides an A record for; aliases are simply linked to that name by a CNAME record, but don't have any other records of their own.

PTR

Used to associate names in the in- addr.arpa domain with hostnames. This is used in the reverse lookup zone where you are mapping IP addresses to hostnames (normally you map hostnames TO addresses)

MX

This RR announces a mail exchanger for a domain. The syntax ofan MX record is

[domain] [ttl] [class] MX preference host

HINFO

Provides information on the system's hardware and software. Its syntax is

[domain] [ttl] [class] HINFO hardware software

NFS

Setup:

The **client**:

mount -t nfs server.whatever.edu:/export ./your_mnt_point

You can automate these mounts by placing a line in our/etc/fstab file which will provide us a list of the devices to mount (both local and network) at boot time.

nfsserver:/usr/local /usr/local nfs nolock 0 0

Also, make sure you are running the RPC portmapper. (rpcinfo–p)

The **server**:

In addition to the portmap program (which is required by both the client and server) you will also require rpc.mountd and rpc.nfsd to be running. These daemons will look at a file called /etc/exports to locate what directories to export via NFS. A simple /etc/exports file would look like:

/FooBarDir

The /etc/exports file offers more options which can be located in man exports. A more involved /etc/exports file might look like:

/home *.domain.site.edu(rw)

/private (noaccess)

/usr/local *.domain.site.edu(ro)

At this point we launch portmap, rpc.mountd, rpc.nfsd and check now with our /usr/sbin/exportfs and /usr/sbin/showmount. If working correctly we now have exported filesystems which our NFS client machines can access.

SAMBA

The /etc/smb.conf file lets you specify a variety of options that control Samba's operation. The install script for Samba establishes a simple /etc/sbm.conf that may meet your requirements.

Samba includes a tool called swat that lets you view and change options by using your Web browser, which is generally much easier than using a text editor.

The swat tool verifies the values of parameters you enter and provides online help. To access swat, point your browser to port 901 of your system.

To configure your Samba server from **SWAT**, you click on simulated tool bar entries:

Globals lets you configure global Samba variables(options)

Shares lets you configure file shares

Printers lets you configure shared printers

Status lets you view the status of the Samba server

View lets you view the smb.conf file

Password lets you add and delete users and change user passwords

Daemons needed for running Samba:

smbd (The SMB daemon)

nmbd

The *nmbd* program is Samba's NetBIOS name and browsing daemon

Samba Commands:

Smbclient

Used to communicate with SMB Servers

smbclient //server_name/share_name [password][-options]

Testparm

checks an smb.conf file for obvious errors and self-consistency

testparm [options] configfile_name [hostname IP_addr]

Testprns

checks a specified printer name against the system printer capabilities (printcap) file. Its command line is:

testprns printername [printcapname]

Smbstatus

lists the current connections on a Samba server