Linux+ Study Guide
Compiled By Ipchain
For ProProfs Linux+ Certification Forums

TABLE OF CONTENTS

-> Chapter 1 - Introduction to Linux

What is Linux?

Linux is a clone of the Unix OS that has been popular in the academia and many business environments for years. It consists of a kernel, which is the core control software, and many libraries and utilities that reply on the kernel to provide features with which users interact.

What is a Workstation?

A **workstation** is a computer that is used primarily or exclusively by individuals, and is often referred to as desktop computer. These computers also require fairly good input/output devices.

What is a Server?

A **server** can mean one of two things: a **program** that responds to network requests from other computers, or the **computer** on which the server program runs. Servers have no need for user-oriented features, and often require a good CPU, along with a descent amount of RAM.

A brief Rundown of PC Hardware

Motherboard – It holds the CPU, RAM, and plug-in cards. It also includes the **BIOS**, which controls the boot process.

CPU – The computer's brain -- it performs most of the computations.

Memory – Holds data, which can include Linux software and the data on which that software operates. Different types of memory exist, and they vary in access and speed.

Disk Storage – Used to retain data. Slower than memory, but higher in capacity.

Video Hardware – Includes video card and monitor.

Input Devices – Devices that enable you to give commands to the computer such as the keyboard and mouse.

Network Devices – Most common are Ethernet cards. Usually used to link computers together.

Audio Hardware – Allow the system to play back sounds and digitize sounds using microphones, etc.

Hardware Requirements

- These might vary from distribution to distribution, but you can read more about it here -> (**http://www.tdl.com/~netex/linux-doc-project/install-guide/node30.html**)

Linux Distributions

- Many Linux distributions are derived from Red Hat or Debian.

1. **Conectiva Linux** – Targeted at users in South and Central America. More info can be found here -> (**http://www.conectivacom**)

2. **Debian GNU/Linux** – Well-liked by open source hard-liners, and those who like tinkering with the underlying text-based configuration files. More info can be found here -> (**http://www.debian.org**)
3. **Fedora Linux** – Free version of Red Hat Linux, more info can be found here -> (**http://fedora.redhat.com**)
4. **Gentoo Linux** – Supports recompiling everything with optimizations to suit your own hardware. More info can be found here -> (**http://www.gentoo.org**)
5. **Libranet GNU/Linux** – Adds improved GUI system administration tools. More info can be found here -> (**http://www.libranet.com**)
6. **Linspire** – Designed as a replacement for Windows on the desktop. More info can be found here -> (**http://www.linspire.com**)
7. **Lycoris –** Aims to be Linux for the desktop, much like **Linspire**. More information can be found here -> (**http://www.lycoris.com**)
8. **Mandrake Linux –** Developed as Red Hat with integrated **K Desktop Environment** (**KDE**). More information can be found here -> (**http://www.linux-mandrake.com/en/**)
9. **Red Hat Linux –** One of the oldest major distributions today, and one of the most influential. More information can be found here -> (**http://www.redhat.com**)
10. **Slackware Linux –** Oldest of the surviving Linux Distributions. Favors manual text-based configuration. More info can be found here -> (**http://www.slackware.com**)
11. **SuSe Linux –** Uses RPMs, but it's not otherwise based on Red Hat. More information can be found here -> (**http://www.suse.com**)
12. **Turbo Linux –** Began as a Red Hat derivative, but recent versions have lost much of this heritage. More information can be found here -> (**http://www.turbolinux.com**)
13. **Xandros Linux –** Adds a very user-friendly installation routine and GUI configuration tools. More information can be found here -> (**http://www.xandros.com**)
14. **Yellow Dog Linux** – Available exclusively for PPC systems, based on Red Hat. More information can be found here -> (**http://www.yellowdoglinux.com**)

The X Window System

- The **X Window System** (**X** for short) is Linux's GUI Environment. It is frequently supplemented by Desktop Managers such as the **K Desktop Environment** (**KDE** ; **http:///www.kde.org**) and the **GNU Network Object Model Environment** (**GNOME**; **http://www.gnome.org**).

Network Clients

- Users run network clients in order to access network resources. For example some of these clients might be:

- Web Browsers - > **Netscape** (http://www.netscape.com), **Mozilla** (http://www.mozilla.org), and **Opera** (http://www.opera.com)

- Mail Readers -> **Mutt** (http://www.mutt.org), and **Kmail** (part of KDE).

- FTP Clients -> **gFTP** (http://gftp.seul.org)

Audio/Visual Programs

- Graphic Viewers/Editors -> **XV** (http://www.trilon.com/xv/) and **GIMP** (http://www.gimp.org)

- **MP3/Multimedia Players** -> **X Multimedia System** (**XMMS**; http://www.xmms.org), and **XAnim** (http:// smurfland.cit,buffalo.edu/xanim/)

- **Audio/Video Editors** -> **Cinelerra** (http://heroines.sourceforge.net/cinelerra.php3), and **Linux Video Studio** (http://ronald.bitfreak.net).

Common Server Programs

- **Web Servers** -> Linux's most popular =>**Apache** (http://www.apache.org), **Zeus** (http://www.zeus.com), **Roxen** (http://www.roxen.com/products/webserver), and **thttpd** (http://www.acme.com).

- **Mail Servers** -> **Sendmail** (http://www.sendmail.org), **Exim** (http://www.exim.org), and **Postfix** (http://www.postfix.org)

Remote Login Servers

- A remote login server allows a user to log into the computer from remote locations.

Note: Most popular SSH implementation for Linux is **OpenSSH** (http://www.openssh.org). **Telnet** also allows remote logins, but it very insecure because it transmits passwords in clear-text.

Miscellaneous Servers

- Proxy servers such as **Squid** (http://www.squid-cache.org)
- Dynamic Host Configuration Protocol Servers (**DHCP**).
- Domain Name System (**DNS**) servers such as **BIND**.
- Remote configuration tools such as **webmin** (http://www.webmin.com)

Partitioning

1. **Primary Partitions –** As the name suggests, this is the primary partition for the system, all files are usually stored on this partition.

2. **Extended Partitions** – A special type of primary partition that serves as a placeholder for the next type --**logical partitions.**
3. **Logical Partitions** – Partitions that reside on extended partitions.

Note: For any one disk, you are limited to **four** primary partitions, or **three** primary partitions and **one** extended partition.

Linux Partition Requirements

- At bare minimum, Linux requires a **single** partition to install and boot. This partition is referred to as the **root** partition, or simply as /.

Common Partitions and Their Uses

Partition (Mount Point)	Typical Size	Use
Swap (not mounted)	½ or 2 times RAM Size	Serves as an adjunct to system RAM; is slow, but enables the system to run more or larger programs.
/home	200MB–200GB	Holds users' data files. Isolating it on a separate partition preserves user data during a system upgrade. Size depends on number of users and their data storage needs.
/boot	5–50MB	Holds critical boot files. Creating as a separate partition allows for circumventing limitations of older BIOS and boot loaders on hard disks over 8GB.
/usr	500MB–6GB	Holds most Linux program and data files; this is frequently the largest partition.
/usr/local	100MB–3GB	Holds Linux program and data files that are unique to this installation, particularly those that you compile yourself.
/var	100MB–200GB	Holds miscellaneous files associated with the day-to-day functioning of a computer. These files are often transient in nature. Most often split off as a separate partition when the system functions as a server that uses the /var directory for server-related files like mail queues.
/tmp	100MB-20GB	Holds temporary files created by ordinary users.
/mnt	N/A	/mnt isn't itself a separate partition; rather, it or its subdirectories are used as mount points for removable media like floppies or CD-ROMs.

Linux Filesystem Options

- Linux supports many filesystems, including the **extended filesystem** (**ext2** or **ext2fs**), the **third extended filesystem** (**ext3fs**), **Reiser** (http://www.namesys.com), the

Extend Filesystem, or **XFS** (http://linux-xfs.sgi.com/projects/xfs), and **The Journaled FileSystem** (http://oss.software.ibm.com/developerworks/opensource/jfs).

Note: The last four filesystems previously mentioned are Linux's four journalizing filesystems.

Important: Linux also supports many non-Linux filesystems including:

1. **FAT** – Used by **DOS** and **Windows**.
2. **NFTS –** Used by Windows NT/2000/XP.
3. **HPFS –** The high performance filesystem used by OS/2.
4. **The Unix Filesystem** (**UFS**; also known as the **Fast Filesystem**, or **FFS**) – Used by various versions of Unix.
5. **HFS** – The **Hierarchical Filesystem** used by **Mac OS**.
6. **ISO-9660** and **Joliet filesystems** used on **CD-ROMS**, and the **Universal Disk Format** (**UDF**), which is the successor to **ISO-9660.**

Partitioning Tools for Linux

1. **fdisk –** Linux's **fdisk** partitioning tool is more flexible than **DOS's FDISK**, but it uses a text-based user interface.
2. **PowerQuest's PartitionMagic –** This program provides a GUI interface and can create partitions that are prepared with **ext2fs**, **ext3fs**, **FAT**, **NTFS**, **or HPFS.** Available at (**http://www.symantec.com**)
3. **GNU Parted –** An open source alternative to **PartitionMagic**. It can create, resize, and move various partitions types, including the **Linux swap**. Available at (**http://www.gnu.org/software/parted/**)
4. **QTParted –** This program provides a **GUI** front-end to **GNU Parted**. Available at (**http://qtparted.sourceforge.net**)

Selecting an Installation Method to install Linux

- Booting

1. **Floppy –** If you have configured the **BIOS** to boot from a floppy disk before any other working boot medium, you can insert the boot floppy and turn on the computer to start the installation process.
2. **CD-ROM** - On a computer that's configured to boot from the **CD-ROM** before other bootable media, you can insert the **CD-ROM** in the drive, then turn on the computer, and the boot program automatically starts up.

Installation Media

- Linux can be installed from **CD-ROM** or **DVD-ROMS**, from a **network path**, from a **hard disk**, and from a floppy disk. Please choose your installation media carefully.

Methods of Interaction during Installation

1. **GUI** – You can use the graphical user interface to make changes and to configure your Linux installation.
2. **Text-Based Prompts** – In the event that a **GUI** is not available, you're given prompts and are asked questions in a text-based installation of Linux.
3. **Scripted Install** – You can use a script that includes the information you'd normally enter with the keyboard or mouse – partition sizes, networking options, packages to install, and so on. Once you've told the system where to look for this configuration file, it then proceeds and installs Linux without asking or prompting you for anything.

Installing Linux

- In order to install Linux, you have to go through several steps:

1. **Language Options** – You have to select the language in which you'd want Linux to be installed.
2. **Keyboard and mouse options** – Select the appropriate keyboard and mouse for your system when prompted.
3. **Partition creation** – Next up you have to create the partitions. Please take your time during this step, and carefully plan all your partitions prior to getting to this step.
4. **Network Configuration** – This is where you tell Linux about your networking hardware, and how it should be configured.
5. **Time and date options** – This is self-explanatory. You simply choose the time and date and hit enter.
6. **Package selection** – This could be a tedious task, depending on how many packages you'd need to be installed.
7. **X configuration** – If the computer is to be used as a workstation, chances are you'd want to configure **X**.
8. **Account creation** – Normally, you'll have to set the root password at this step.
9. **Boot loader configuration** – In order for Linux to boot, it needs to have a boot loader installed and configured properly.

Available Boot Loaders

- **LILO –** This boot loader can boot directly into the Linux kernel, and it can function as either a primary or secondary boot loader.
- **GRUB –** This boot loader is on the way to becoming the standard Linux boot loader.
- **OS Loader –** This is a secondary boot load that cannot directly boot into Linux, but it can boot a disk file that can contain **LILO** or **GRUB**, and hence boot into Linux indirectly.

- **System Commander** – This is the Cadillac of boot loaders, with some very advanced features. It is available from (**http://www.v-com.com**)
- **LOADIN** – It is a **DOS** program that can be used to boot **Linux** after **DOS** has already loaded.

Configuring LILO

- In order to configure **LILO**, it is necessary to edit the file **/etc/lilo.conf**

Sample lilo.conf file:

```
boot=/dev/hdb
prompt
delay=30
map=/boot/map
install=/boot/boot.b
default=linux
lba32
message=/boot/message
image=/boot/bzImage-2.4.10
        label=linux
        root=/dev/hdb8
        append="mem=256M"
        read-only
        other=/dev/hdb2
        label=windows
        table=/dev/hdb
```

* Please refer to **http://www.netadmintools.com/html/5lilo.conf.man.html** to see the different configuration options for **lilo.conf.**

Important: Once you've edited **lilo.conf** correctly, type **lilo** in order to activate those changes.

Configuring GRUB

- The traditional location for the **GRUB** configuration file is **/boot/grub/menu.lst**. **Fedora**, **Gentoo**, and **Red Hat** are the exception to this rule, as they all use **/boot/grub/grub.conf**.

Sample **menu.lst** file:

```
default=0
timeout=3
splashimage=(hd0,2)/grub/splash.xpm.gz
title Linux (2.4.10)
   root (hd0,2)
```

```
    kernel /bzImage-2.4.10 ro root=/dev/hda8 mem=256M
    boot
title Windows
    rootnoverify(hd0,1)
    chainloader +1
    boot
```

Note: Because **GRUB** wasn't designed specifically for Linux, it introduces a new way of referring to hard disk, and their partitions. **GRUB** uses strings of the form **(hdx,y)**, where **x** is a disk number, and **y** a partition number.

* To read more about configuration options for **GRUB**. Please use the following link -> (**http://www.gnu.org/software/grub/manual/grub.html**)

-> Chapter 2 - Basic Linux Commands

The **ls** command.

This command is used to list the files and directories.

For example: **ls** /var lists all the files and directories located in /var.

Important switches to pass to **ls**:

-a or --all – Use these parameters when you want to view hidden files, usually represented by a period (.) at the beginning of their filenames.

-l (not a 1 but a lowercase L) – Use this parameter when you wish to display additional information such as the file's permission string.

-p or --file-type – Use this parameter to find out the type of file you're dealing with. The meanings are as follows:

/ means the file is a directory.
@ means the file is a symbolic link.
= means the file is a socket.
| means the file is a pipe.

Using wildcards with ls.

? Stands in for a single character. For instance **b??l** matches bull, ball, or any other four-letter filename that begins with **b** and ends with **l**.

***** Matches any character or set of characters, including no character. For instance b*k matches book, balk, and buck. It also matches bk, bbk, and backtrack.

[] Normally matches any character in the set. For instance, b[ao][lo]k matches balk and book, but not bulk. It's also possible to specify a range of values such as b[a-z]ck. This will match any four-character filenames of this form whose second character is a lowercase letter.

Finding the Current Directory

You can find the current directory (the one you're working on) at any time by using the **pwd** command.

Changing Directories

You can change directories by using the **cd** command.

Manipulating Files

Copying – Use the **cp** command to copy a file.

Syntax: cp [options] source destination

Common switches:

-f or --force –This parameter forces the system to overwrite any existing files without prompting.

-i or --interactive – This parameter causes **cp** to ask you before overwriting any files.

-p or –preserve – This parameter preserves ownership and permissions. Normally, when a file is copied it is owned by the user who issued the command.

-u or --update – This parameter tells **cp** to copy the file only if the original file is newer than the target file.

Moving – Use the **mv** command to move or rename files and directories.

Syntax: mv [options] source destination

Common switches: This command takes many of the same options as cp does with the exception of --preserve and --recursive.

Important: To rename a file, use **mv file file1**, where file is the original file, and file1 is the renamed file. To hide a file, use **mv file .file**, where file is the original file, and .file is the new hidden file.

Removing – Use the **rm** command to remove files and directories.

Syntax: rm [options] files

Caution: Please be careful when using this command because if you mistakenly typed **rm –R /**, it will destroy an entire Linux System.

Creating **hard** and **soft links** (also called **symbolic links**) – Use the **ln** command to create hard or soft links.

Syntax: ln [options] source link

Important: The **ln** command creates hard links by default. Use the **–s or --symbolic** option to create a soft or symbolic link.

Manipulating Directories

Creating Directories – Use the **mkdir** command to create a directory.

Syntax: mkdir [options] directory-names

Important: Normally, if you specify the creation of a new directory within another directory that does not exist, mkdir responds with a **No such file or directory** error. To overcome this issue, use the **–p or --parents** option.

Deleting Directories – Use the **rmdir** command to remove directories.

Syntax: rmdir [options] directory-names

Important: Use the **–p or --parents** to have rmdir to delete an entire directory tree. For example, typing **rmdir –p hello/there/pal/** causes **rmdir** to delete **hello/there/pal**, then **hello/there**, and finally **hello**, provided no other files or directories are present.

Locating Files

Finding Files – Use the **find** command to find files.

Syntax: find [path…] [expression…]

Common switches:

-name pattern – This option causes find to look for specific file names. Ex: **find / -name john** instructs find to look for all files named john. Use can also use wildcards here. See example below:

find /home –name "*.vb" – This command looks for any files with the *.vb extension in the /home directory.

-perm mode – This option instructs find to look for files with specify permissions. If you precede **mode** with a **+,** find locates files in which any of the specified permissions bits are set**.** If you precede **mode** with a **-,** find locates files in which all the specified permission bits are set.

-size n – This option instructs find to look for files with the specified file size, and **n** is specified in 512-byte blocks.

-gid GID – This option instructs find to search for files whose group ID (**GID**) is set to **GID.**

-uid UID – This option instructs find to search for files whose user ID (**UID**) is set to **UID**.

-maxdepth levels – This option is used when you want to limit find to search through a limited number of subdirectories.

The locate Command – This command is used to search for files by name.

The **locate** command works from a database that it maintains so it may not find recent files, or it may return names of files that no longer exist.

Syntax: **locate search-string** , where search-string is the string that appears in the filename.

The whereis Command – This utility used to quickly find program executables and related files like documentation, or configuration files.

Example: **whereis ls**

Examining File's Contents

The **grep** command is used to search for files that contain a specified string.

Syntax: grep [options] pattern [files]

Common switches:

-i or --ignore-case – This option instructs grep to perform a case-insensitive search instead of the normal case-sensitive search it performs by default.

Example: **grep named.conf /etc/** - This command searches the /etc/ directory in order to locate the filename named.conf.

The cat Command – This command is used to view the contents of a file.

Example: **cat filename.txt** will output the contents of filename.txt to the screen.

Example 2: **cat /home/user/hello /hello/user/there > newfile** – This command combines two files into one. It combines the contents of **/home/user/hello and /hello/user/there,** and creates **newfile** with all the information contained in the aforementioned files.

Example 3: **cat - > input.txt** – This command inserts everything you type into a file named input.txt. Once you're done inserting text, press Ctrl + D to exit.
The **more** and **less** Commands – Both commands are used to view the contents of a file.

Example: **more filename.txt**, and **less filename.txt**.

The **tail** Command – This command allows you to view the last 10 lines of a file by default. Use the **n** option to specify the number of lines you desire to view.

Example: **tail –n 15 /var/log/messages** – This command displays the last 15 lines in **/var/log/messages**.

Note: Use tail –f /var/log/messages to view event logs as they are logged in real time.

Redirection and Pipes

- To redirect the output of **ifconfig** to a file named ips.txt use the following command:

ifconfig > ips.txt

> sends the standard output of a command to a file you specify.

< replaces the standard input with the file you specify.

Example: **myscript < input.txt** – This command passes input.txt to a script named myscript to use as input.

***** To have a program use another program's output as input, use a **pipe**, represented by **|**

Example: **ps ax | grep ftp** – This command searches for the string ftp in the ps output.

File Access Components and Permissions

Example: **ls –l /home/john/filename.txt**

-rwxr-xr-x 1 john newbies 71234 Sep 8 03:40 /home/john/filename.txt

The previous output means the following:

The first component (**-rwxr-xr-x**) is the file permission string.

The second component (**1**) is the number of hard links. In this example, 1 means that only one filename points to this file.

The third component (**john**) is the name of owner of the file. Long usernames might be truncated.

The fourth component (**newbies**) is the group to which the file belongs.

The fifth component (**71234**) is the file size expressed in bytes.

The sixth component (**Sep 8 03:40**) is the file creation date.

The seventh component (**/home/john/filename.txt**) is the name of the file, along with the path to it.

Interpreting File Access Codes

Linux File Type Codes

File Type Codes

Code	Meaning
-	Normal data file; may be text, an executable program, graphics, compressed data, or just about any other type of data.
d	Directory; disk directories are files just like any others, but they contain filenames and pointers to disk inodes.
l	Symbolic link; the file contains the name of another file or directory. When Linux accesses the symbolic link, it tries to read the linked-to file.
p	Named pipe; a pipe enables two running Linux programs to communicate with each other. One opens the pipe for reading, and the other opens it for writing, enabling data to be transferred between the programs.
s	Socket; a socket is similar to a named pipe, but it permits network and bidirectional links.
b	Block device; a file that corresponds to a hardware device to and from which data is transferred in blocks of more than one byte. Disk devices (hard disks, floppies, CD-ROMs, and so on) are common block devices.

c	Character device; a file that corresponds to a hardware device to and from which data is transferred in units of one byte. Examples include parallel and RS-232 serial port devices.

Permission Example**: rwxr-xr-x**

These **nine** characters denote the file's permission and are broken up in blocks of **three** characters.

The first set **3** characters denotes the permissions for the **owner**. In the previous example the owner has **read** (**r**), **write** (**w**), and **execute** (**x**) permissions to the file.

The second set of **3** characters denotes the permissions for the **group**. In the previously example the group has **read** (**r**), and **execute** (**x**) permissions to the file.

The third set of **3** characters denotes the permissions for **all users**. In the previous example all users have **read** (**r**), and **execute** (**x**) permissions to the file.

Permissions in Octal Form

4 - Read
2 - Write
1 - Execute

So the aforementioned permission represented in **octal form** would be as follows:

755 (Owner has all permissions **-> 4+2+1, Group** has **Read** and **Execute -> 4+1, and All users** have **Read** and **Execute -> 4+1)**

*** SUID** programs are indicated by an **s** in the owner's execute bit position of the permission string, as in **rwsr-xr-x.** Set this option to allow users to run a program with the permission of whoever owns the program, rather than with the permissions of the user who runs the program.

*** SGID (Set group ID)** option is similar to the **SUID** option, but it sets the group of the running program to the group of the file. It's indicated by an **s** in the group execute permission as in **rwxr-sr-x**.

Sticky Bit – This is used to protect files from being deleted by those who don't own the files. The **sticky bit** is represented by a **t** in the **world** or **all users** execute bit position, as in **rwxr-xr-t**.

Changing the File Ownership and Permissions

To change a file's owner, use the **chown** command.

Syntax: chown [options] [newowner] [:newgroup] filename…

Example: **chown john:newbies hello.txt**

This command changes the file ownership of hello.txt to john, and changes the group to newbies.

To change a file's group, use the **chgrp** command.
The chgrp command takes the same options as **chown** does. Only the **root** user can change the owner of a file, but the owner of a file can change the group of a file to any group to which he/she belongs.

* The **root** user is the system administrator in Linux. In other words, the user with administrative privileges, and super powers.

Permissions Modification

Use **chmod** to modify a file's permissions.

Example: **chmod 755 file.txt**

Codes Used in Symbolic Modes

Permission Set code	Meaning	Change type code	Meaning	Permission to modify code	Meaning

u	owner	+	add	r	read
g	group	-	remove	w	write
o	world	=	set equal to	x	execute
a	all	X			execute only if file is directory or already has execute permission.
				s	SUID or SGID
				t	sticky bit
				u	existing owner's permissions
				g	existing group permissions
				o	existing world permissions

Setting Default Permissions

When a user creates a file, that file has default ownership and permissions. The default owner is, understandably, the user who created the file. The default group is the user's primary group.

The default permissions, however, are configurable and defined by the *user mask (umask)*, which is set by the umask command. This command takes as input an octal value that represents the bits to be removed from 777 permissions for directories, or from 666 permissions for files, when creating a new file or directory. Please see table below:

Umask	Created Files	Created Directories
000	666 (rw-rw-rw-)	777 (rwxrwxrwx)
002	664 (rw-rw-r--)	775 (rwxrwxr-x)
022	644 (rw-r--r--)	755 (rwxr-xr-x)
027	640 (rw-r-----)	750 (rwxr-x---)
077	600 (rw-------)	700 (rwx------)
277	400 (r--------)	500 (r-x------)

* To find out the current umask, simply type **umask.**

Using VI Editor

This topic is very lengthy so please take a look at the following link for a tutorial -> http://www.eng.hawaii.edu/Tutor/vi.html#modes

Useful commands in VI.

To search forward for a text in a file, type / in command mode

To search backward for a text in a file, type ? in command mode

To write to buffer and quit, type **:wq**

To replace all instances of one string by another, type **:%s/original/replacement**

Using SED

Books have been written about **SED** so I am only going to give you the basics. I strongly suggest you do some research on your own.

The command **SED** stands for 'stream editor', and it's a text editor that uses command-line commands rather than GUI operations like **VI**.

Common SED commands:

sed 's/regexp/replacement' -- Replace text that matches the regular expression (**regexp**) with **replacement**.

To do the same thing globally, append a g at the end. Example: **sed 's/regexp/replacement/g'**

Setting Environment Variables

To set an environment variable in bash, use the following command**: Variable=value**

Example: **NNTPSERVER=news.cnn.com**

* After you have set the variable, you need to export it in order to be able to use it. Please do so by typing **export NNTPSERVER.**

Note: For the **tcsh** shell, you have to use the **setenv** command to set the variables. For example:

setenv NNTPSERVER news.cnn.com

Adding Paths to existing variables

You can add paths to existing variables in bash by typing the following:

export PATH=$PATH:/hello/there

The previous command adds the path **:/hello/there** to the **PATH** environment variable.

For **tsch,** the command is slightly different. See below**:**

setenv PATH "${PATH}:/hello/there"

Some Environment Variables and their meanings

Variable Name	Explanation
USER	This is your current username. It's a variable that's maintained by the system.
SHELL	This variable holds the path to the current command shell.
PWD	This is the present working directory. This environment variable is maintained by the system. Programs may use it to search for files when you don't provide a complete pathname.
HOSTNAME	This is the current TCP/IP hostname of the computer.
PATH	This is an unusually important environment variable. It sets the *path* for a session, which is a colon-delimited list of directories in which Linux searches for executable programs when you type a program name. For instance, if PATH is /bin:/usr/bin and you type **ls**, Linux looks for an executable program called ls in /bin and then in /usr/bin. If the command you type isn't on the path, Linux responds with a command not found error. The PATH variable is typically built up in several configuration files, such as /etc/profile and the .bashrc file in

	the user's home directory.
HOME	This variable points to your home directory. Some programs use it to help them look for configuration files or as a default location in which to store files.
PS1	This is the default prompt in bash. It generally includes variables of its own, such as \u (for the username), \h (for the hostname), and \W (for the current working directory). This value is frequently set in /etc/profile, but it is often overridden by users.

-> Chapter 3 - User Management

Most linux systems contain restrictions on root logins, so they can only be done from the console**. This helps prevent outsiders from gaining access to a system over the network by performing brute-force attacks.**

Switching identities: The **su** command lets you temporarily acquire superuser privileges or take any other user's identity. All you have to know is the appropriate password.

Running an individual program as the superuser – Once configured, the **sudo** command allows you to run a single program as **root**.

Important: The **/etc/sudoers** file contains a list of users who may use the **sudo** command.

* Linux usernames are case sensitive, so **George** is not the same as **george**.

Important Files:

/etc/passwd - This file holds all the information about a user account.
/etc/group – This password holds all the groups certain users belong to.

* Use the **newgrp** command to create a new group. For Example: **newgrp elite** creates a new group called elite.

* Normally, the **/home** directory holds all the user accounts. For example if you add a user, it will be added to **/home/username** unless otherwise specified.

Adding Users

Use the **useradd** command to add new users to a Linux system.

Syntax: **useradd** [-c comment] [-d home-dir] [-e expire-date] [-f inactive-days] [-g initial-group] [-G group[,…]] [-m [-k skeleton-dir] | -M] [-p password] [-s shell] [-u UID [-o]] [-r] [n] username.

Example: **useradd mary** adds the mary user account to the system.

Most of the options are self-explanatory, but you are encouraged to look them up on your own to see what each and every one of them does.

Important: The file **/etc/login.defs** holds all of the options that are passed to useradd by default when no options are specified.

Modifying User Accounts

Use the **passwd** command to change a user's password, or to set a new one.

Syntax: **passwd** [-k] [-l] [-u [-f]] [-d] [-S] username

Example: **passwd john** sets a new password for john, or changes his current password.

Again, you're responsible to look at what each of the options does on your own.

* To modify a user account, use the **usermod** command.

For example: **usermod john –l jack** changes the username from john to jack.

Ownership Modification

The following command modifies the ownership of all files in john's home directory to jack **- > chown –R jack /home/john.**

Using chage

The **chage** command allows you to modify account settings relating to account expiration.

Syntax: **chage** [-l] [-m mindays] [-M maxdays] [-d lastday] [-I inactivedays] [-E expiredate] [-W warndays] username

* You are encouraged to look at what each of the options does on your own.

Important: The **.bashrc** file can be used to set user-specific bash shell features.

Examining the **/etc/passwd** file

Here's part of the output of a **/etc/passwd** file.

john:x:520:101:John Wilson:/home/john:/bin/bash

- The first field (**john**) is the username of the user.
- The second field (**x**) is the password of the user. In this case, it means the user has a password set, and it should be encrypted in the **/etc/shadow** file.

Note: A ***** in the password field means the user account is **disabled**.

- The third field (**520**) is the user ID of the user.
- The fourth field (**101**) is the group ID of the user.
- The fifth field (**John Wilson**) is usually a comment or the user's real name. In this case, it is the user's real name.
- The sixth field (**/home/john**) is the user's home directory.
- The seventh field (**/bin/bash**) is the type of shell.

Deleting User Accounts

To delete a user account, use the **userdel** command.

Example: To delete a user account and it's home directory, use **userdel –r john** . Otherwise, to delete only the user, and not its home directory or files, use **userdel john**.

Adding Groups

Use the **groupadd** command to add a new group.

Syntax: **groupadd** [-g GID [-o]] [-r] [-f] groupname

Example: **groupadd friends** adds a new group called **friends** to the system.

* You are encouraged to look at what each of the options does on your own.

Modifying Group Information

Use the **groupmod** command to modify group information.

Syntax: **groupmod** [-g GID [-o]] [-n newgroupname] oldgroupname

* You are encouraged to look at what each of the options does on your own.

Directly Modifying Group Configuration Files

Before you can manually edit the /etc/group file, you must understand what each field represents.

Sample output from cat /etc/group:

friends:x:402:james,chris,skip,quinn, alice, jnorm, fsufan, mana333

- The first field (**friends**) is the group name.
- The second field (**x**) means this system is using shadow passwords, and the user has an encrypted password in **/etc/shadow**.
- The third field (**402**) is the group ID.
- The fourth field (**james,chris,skip,quinn, alice, jnorm, fsufan, mana333**) is usually a list of users that are currently members of the group.

Deleting Groups

Use the **groupdel** command to delete a group from the system.

Example: **groupdel friends**

Enforcing User Password Security

Programs such as **Crack** exist to crack simple and not so complex user passwords ; therefore, it is very important to understand how you can make it very difficult for these programs to crack your password. Here are a few tips:

1. **Add numbers or punctuation** to your passwords. Adding $, #, 3, and so forth can make your password very difficult to crack.
2. **Mix the case**. It'll be very easy to crack a password if it's all in lowercase, so mix your case. Example: tHisISagood$passW8rd
3. **Order reversal**. Try to reverse the order in which symbols and letters are used, do not use all symbols in one place, and all letters in another.

Steps for Reducing the Risk of Compromised Passwords

- Use strong passwords.
- Change password frequently.
- Use shadow passwords.
- Keep passwords secret.
- Use secure remote login protocols – No ftp or telnet as they send your password in clear text.
- Be alert to shoulder surfing – Watch people around you.
- Disable unused accounts.

* Use the **pwconv** command to convert the /etc/passwd file to use shadow passwords.

* Use the **pwunconv** command to reverse the aforementioned command.

Configuring Access via PAM

Most servers employ PAM to do the bulk of the work of authenticating and authorizing access.

You can configure PAM through the file **/etc/pam.d/system-auth** file, or through the file corresponding to the server in **/etc/pam.d**, for example **/etc/pam.d/ftp**.

The **/etc/pam.d/system-auth** is a generic file that refers to the **pam_stack.so**.

* Some important login servers use the **/etc/pam.d/login** control file.

* Access control is handled by a PAM module called **pam_access.so**

* The PAM system searches the **/etc/security/access.conf** file for the first entry to match the user who is trying to get access to the system so you can configure several things on the specific file. The format is as follows:

[+|-] : user(s) : source(s)

+ - Grants access
- - Denies access.

Example: **-:john lucas:.cnn.com**

The previous example denies access to users **john** and **lucas** when they attempt to log in from **cnn.com.**

Controlling FTP access

The **/etc/ftpusers** contains a list of users who are not allowed to log in via ftp.

Controlling NFS (Network File System Access)

NFS provides user-level access controls in the sense that it passes UID and GID information stored on the server to the client. Its security focuses on limiting access to trusted clients.

This type of access is controlled in the **/etc/exports** file.

Example: /opt client1(rw),client5(ro)

In the previous example, **/opt** is the shared directory. **Client1** and **Client5** are the hostnames of the computers accessing the share, and **rw** and **ro** are the type of permissions to the shares (**ro = read-only**, **rw = read-write**).

Controlling Samba Access

Samba is a server package for the Server Message Block/Common Internet File System (SMB/CIFS) protocol suite, which is most commonly used for file sharing among Windows computers.

Samba access is configured through its configuration file, usually in **/etc/samba/smb.conf**.

Important Options:

The **guess ok = yes** parameter, if set, enables guess access without the use of a username and password.

The valid **users = john lucas pedro** parameter, if set, sets the valid users to john, lucas, and pedro respectively

Important: To check the **smb.conf** file, use the **testparm** command.

Note: Look up the other options on your own.

Controlling *root* Access

Due to the nature of the **root** account, you'd want to restrict it as much as possible.

The **/etc/securetty** file holds the terminals from which the root user is allowed to log in.

***** To configure **SSH** to deny the root user to log in directly, specify the following in the sshd configuration file at **/etc/ssh/sshd_config**:

PermitRootLogin no

Setting Filesystem Quotas

Quotas require both support in the **kernel** for the **filesystem** being used and various userspace utilities.

As of the early 2.6.*x* kernels, the **ext2fs**, **ext3fs**, and **ReiserFS** filesystems support quotas, but you must explicitly enable support via the Quota Support kernel option in the **filesystem** area when recompiling your kernel.

The two general quota systems available for Linux are as follows:

2.4.x kernels – They use what we call version 1 quotas.
2.6.x kernels – They use what we call version 2 quotas.

* To enable quota support, you must modify the appropriate partitions in the **/etc/fstab** file.
Example: /dev/hda1 /home ext3 **usrqupta,grpquota** 1 1

This line activates both, user quota and group quota support for the **/dev/hda1** partition.

It is often required to configure the quota package's SysV startup script to run when the system boots.

For that, the following command is typically used:

chkconfig quota on

* To edit quotas for a user, use the **edquota** command.

Example: **edquota john**

* Pass the **–t** option to **edquota** to specify the grace period**.**

* Use the **quotacheck** command to verify and update quota information on quota-enabled disks.

-> Chapter 4 - Storage devices and Partitions

The most common type of device today is the **ATA** hard disk**.** Such disks are represented and identified in Linux by **/dev/hdx**, where **x** is a letter from **a** onward.

* The **master** disk on the first controller is represented by **/dev/hda,** and the **slave** disk on that controller by **/dev/hdb.**

* The **master** disk on the **second** controller is represented by **/dev/hdc** and so forth**.**

Note: Drive letters can be skipped, for example you might have two master disks on their chains represented by **/dev/hda**, and **/dev/hdc**.

* **SCSI** devices are identified by **/dev/sdx**, and unlike SATA devices, they are assigned letters sequentially beginning with **a**.

As a result, a system with two **SCSI** disks will identify them as **/dev/sda** and **dev/sdb**, even if they have noncontiguous **SCSI ID** numbers.

* Both **ATA** and **SCSI** hard disks are commonly broken into partitions. For example:

/dev/hda2 identifies a specific partition on **/dev/hda**.
/dev/sdb3 identifies a specific partition on **/dev/sdb**.

Note: On x86 hardware, primary partitions are usually starting with a number from 1-4, and logical partitions are usually represented by a number from 5 on.
* The actual order of partitions on a disk does not need to correspond to their partition numbers, and there might be gaps between partitions.

Example: **/dev/hda4** might appear before **/dev/hda3**.

Example2: A disk might have **/dev/hda2** and **/dev/hda4** but not **/dev/hda3**.

Floppy Disks

/dev/fd0 is typically recognized as the first floppy disk.
/dev/fd1 is typically recognized as the second floppy disk.

* The **fdformat** utility prepares a floppy disk for use.

Optical SCSI drives are usually represented by **/dev/scdx,** where **x** is a number from **0** on**.**

Magnetic Tape Devices

These devices are identified by a pair of device files as follows:

ATA Devices: /dev/htx and **/dev/nhtx,** where **x** is a number from **0** up**.**
SCSI Devices: /dev/stx and **/dev/nstx.**

Note: The **/dev/htx** file, when accessed, causes the tape to automatically rewind. On the other hand, **/dev/nhtx** is nonrewinding.

Partition Management and Maintenance

To create partitions, use the **fdisk** command**.**

Example: **fdisk /dev/had**

Common fdisk Commands

Command	Description
d	Deletes a partition**.**
n	Creates a new partition.
p	Displays (prints) the partition layout.
q	Quits without saving changes.
t	Changes a partition's type code.
w	Writes (saves) changes and quits.

Creating New Filesystems

Use the **mkfs** command to create a new filesystem.

Syntax: **mkfs** [-V] [-t fstype] [options] device [blocks]

* You can specify the filesystem type with the **–t fstype** option.

Filesystem Types:

- **Ext2** (for ext2fs)
- **Ext3** (for ex3fs)
- **Reiserfs** (for ReiserFS)
- **Xfs** (for XFS)
- **Jfs** (for JFS)
- **Msdos** (for FAT)
- **Minix** (for Minix)

* You are encouraged to look at what each of the other options for **mkfs** does on your own.

Note: Check the **/sbin** directory for files whose names begin with **mkfs** to see what filesystem-specific mkfs tools exist on your system.

For example: You can use **mkreiserfs** to prepare a **ReiserFS** partition, **mkfs.ext3** to prepare an **ext3** partition, or **mkdosfs** to prepare a **FAT** partition.

Checking a Filesystem for Errors

To check a filesystem for errors, use the **fsck** command.

Syntax: **fsck** [-sACVRTNP] [-t fstype] [- -] [fsck-options] filesystems

Example: **fsck /dev/sda1**

* Use the **–A** option to check all the filesystems marked to be checked in **/etc/fstab**.

Note: Linux runs **fsck** automatically at startup on partitions marked in **/etc/fstab**.

Evaluating Swap Space

To evaluate swap space, use the **free** command.

Syntax: **free**

* This command reports the Total Physical Memory and Swap Space available, along with what has been used, and the total amount of free space.

Note: As a general rule, your swap space should be **1.5-2** times as large as physical **RAM**.

Steps to add a Swap File

1. Use the **dd** command to create a swap file**.**
2. Use the **mkswap** command to initialize the swap file for use. For example: **mkswap /swap.file**
3. Use the **swapon** command to begin using the newly initialized swap space. For example: **swapon** /swap.file

Note: To make this swap space permanent, add an entry to **/etc/fstab** that looks like the following:

/swap.file swap swap defaults 0 0

Important: To use all of the swap spaces defined in **/etc/fstab**, use the **swapon –a** command.

* To deactivate the use of swap space, use **swapoff /swap.file** or **swapoff -a** to deactivate all.

Adding a Swap Partition

1. Allocate space for the new partition
2. Create a new partition and give it a type code of **0x82** ("**Linux Swap**").
3. Use **mkswap** to prepare the swap partition to be space space. For example: **mkswap /dev/sdc3**
4. Once the space has been prepared for use, add it manually using the **swapon** command. For example: **swapon /dev/sdc3**
5. To add it permanently, edit the **/etc/fstab** file.

Identifying Partitions

* To identify partitions on a specific disk device use the **fdisk –l device** command. For example **fdisk –l /dev/hda**

Mounting and Unmounting Partitions

To mount a filesystem to a mount point, use the **mount** command.

Syntax: **mount** [-alrsvw] [-t fstype] [-o options] [device] [mountpoint]

Example: **mount /dev/sdb5 /mnt/shared**

* The previous command mounts the contents of **/dev/sbd5** to **/mnt/shared**.

Important: Normally, only the root user may issue a **mount** command. To have an ordinary user be able to use the command, you need to add **user**, **users**, or **owner** option to the mount point in the **/etc/fstab** file.

Important Switches:

-a – The **-a** parameter causes mount to mount all the filesystems listed in **the /etc/fstab file**

* You are encouraged to look at what each of the other options does on your own.

Using umount

Use the **umount** command to unmount filesystems.

Syntax**: umount** [-afnrv] [-t fstype] [device | mountpoint]

Note: Like mount, use the **-a** option to unmount all filesystems.

Accessing SMB/CIFS Shares

The main package for this is **Samba**, and it comes with all major Linux distributions.

Samba includes two major client programs:

1. **smbclient** – To provide FTP-like access to remote shares.
2. **smbmount** – To actually mount the shares.

* To use **smbmount**, type **smbmount /server/share /mount/point,** where **server** and **share** are the name of the server and share respectively, and **/mount/point** is the local mount point you wish to use.

Note: You can also use the **mount** command to accomplish the same thing. For example: **mount –t smbfs /server/share /mount/point** does the same thing.

Finding Information about the Disk Space

Use the **df** command to find information about the current disk space and usage.

Forms of RAID

RAID 0 (striping) – The result is **improved performance** because disk accesses are spread across multiple physical disks. **Reliability** is not improved, though. Failure of a single disk in the array will cause **data loss**.

RAID 1 (mirroring) – Provides **redundancy** that can protect against drive failures, but **slows performance**, at least when it is implemented in the OS. (Some hardware RAID controllers can perform this task without a performance hit)

RAID 4/5 – Combines features of both **RAID 0** and **RAID 1:** It spreads the data across multiple disks and provides **redundancy**. This type of array uses parity bits to regenerate the data should a single drive fail.

RAID 6 – Provides protection for failure of **two drives**, rather than the one that can be handled by **RAID 4/5**. Uses an extra drive and is considered experimental as of early 2.6.x kernels.

Linux RAID Configuration

To use **RAID**, you must compile support for it into your kernel, and it also uses two packages, **raidtools**, and **mdadm**.

Note: **raidtools** is the tool covered in this guide. To read more about **mdadm**, please use the following link: **http://www.linuxcommand.org/man_pages/mdadm8.html**

*** raidtools** uses a configuration file, **/etc/raidtab** to define **RAID** arrays.

* To actually define your **RAID** configuration, you must use a file called **/etc/raidtab**. A simple **RAID 1** configuration looks like this:

```
raiddev /dev/md0

raid-level              1
nr-raid-disks           2
persistent-superblock   1
nr-spare-disks          1
device          /dev/sda1
raid-disk               0
device          /dev/sdb1
raid-disk               1
device          /dev/sdc1
spare-disk              0
```

* The previous configuration creates a **RAID 1** (raid-level) device that can be accessed as **/dev/md0**.

* You are encouraged to look more information about the options in **/etc/raidtab**.

* Once you have created your **/etc/raidtab** file, you must initialize the system by using **mkraid**. For example: **mkraid /dev/md0**

- The previous command reads /etc/raidtab and initializes the specified device(s) using the settings in that file.

Writing to Optical Discs

Many programs exist to help you with this task, some are mentioned below:

1. **X-CD-ROAST** (**http://www.xdcroast.org**)
2. **ECLIPt Roaster** (**http://eclipt.uni-klu.ac.at**)
3. **GNOME Toaster** (**http://gnometoaster.rulez.org**)
4. **K3B** (**http://k3b.sourceforge.net**)

* Please take a look at each and every one of them in more detail as you won't find any information related to them in this guide.

Using Command-Line tools to create images and burning them

Use the **mkisfos** command to generate iso9660 filesystems.

Example: **mkisofs -J -r -V "volume name" –o ../imagine.iso ./**

Read more about what this command does, and its appropriate switches here **->** **http://www.itg.uiuc.edu/help/mkisofs/mkisofs.htm**

* Once you have created an image, you can burn it with **cdrecord**. For example: **cdrecord dev=0,4,0 speed=2 ../image.iso**

* You can read more about **cdrecord** here -> **http://www.linuxcommand.org/man_pages/cdrecord1.html**

Common Backup Hardware

1. **Tapes** – They are the most popular choice for backing up entire computers.
2. **Hard disks** – Often used to back up computers, but not as popular as tapes.
3. **Removable disks** – The high cost per gigabyte and low capacity makes them suitable for personal backups, rather than system backups.
4. **Optical** – Extremely reliable and therefore well-suited for long-term archival storage.

Using cpio or tar to Back Up a Computer

The cpio program is one of the several tools that can be used to back up a computer. For example the following command creates an entire backup of the system, and copies it to the tape device at **/dev/st0**.

find / | cpio –oF /dev/st0

Common switches:

-o or --create option – This is the copy-out mode, and it creates an archive and copies files into it.
-i or --extract option – This is the copy-in mode, and it extracts data from an existing archive.
-p or --pass-through option – This is the copy-pass mode, and it combines the copy-out and copy-in modes, enabling you to copy a directory tree from one location to another.

* You are encouraged to look at what each of the other options for **cpio** does on your own.

Using tar to Back Up a System

The following command creates a backup of the directories **/home /boot /var**, and **/** to the tape device **/dev/st0.**

tar cvlpf /dev/st0 /home /boot /var /

* You can read more about tar and its options here - > **http://sunsite.ualberta.ca/Documentation/Gnu/tar-1.13/html_chapter/tar_toc.html**

Controlling a Tape Drive

Use the **mt** command to control a tape drive.

Syntax: **mt** –f device operation [count] [arguments]

Example: **mt –f /dev/nst0 rewind**

- The previous command rewinds the tape in **/dev/nst0**

* The device parameter is the tape device filename, and operations supports many things, including the following:

- **fsf** – Moves forward *count* files.
- **bsf** – Moves backward *count* files.
- **eod or seod** – Moves to the end of data on tape.
- **rewind** – Rewinds the tape.
- **Etc, etc**.

Performing an Incremental Backup

To use tar to perform an incremental backup, use the **--listed-incremental** option as follows:

tar cvplf /dev/st0 –listed-incremental /root/inc /home/ /

* The previous command stores a list of the files that have been backed up in **/root/inc**. The next time the same command is issued; tar will not back up files that have already been backed up.

Preparing for Disaster: Backup Recovery

Suppose that a user asks you to restore a file she/he has accidentally deleted. If you have performed a full backup of the system, it should be no problem for you. You can simply type the following to restore the file from tape:

cd /
tar xvlpf /dev/st0 /home/username/filename

or

cd /
cpio –ivF /dev/st0 /home/username/filename

-> Chapter 5 - Package Concepts

Any OS is defined largely by the files it installs on the computer. In the case of Linux, these files include the kernel; critical utilities stored in directories such as **/bin**, **/sbin**, **/usr/bin**, and **/usr/sbin**.

It becomes very hard to keep track of the packages installed, so to help you with that various package maintenance utilities have emerged. Some of these include the **RPM** (**RPM Package Manager**) and **Debian Package tools**.

Package Databases

- **RPM** installs its database in **/var/lib/rpm**
- **Debian** packages the database in **/var/lib/dpkg**

* Here's a great **RPM HOWTO** to help you with the coming section ->
(**http://tldp.org/HOWTO/RPM-HOWTO**)

The convention for naming RPM packages

packagename-a.b.c-x.arch.rpm

packagename – This is the name of the package, such as **mysql.**
a.b.c – This is the version number, such as 3.1.1a**.**
x – The number following the version number is the **build number** (also known as **release number**).
arch – The final component preceding the **.rpm** extension is the code for the package's architecture**,** with **i386** being the most common one**.**

Note: CPU-independent packages generally use the **noarch** architecture.

Important: Use the **--nodeps** switch with **rpm** to install the package anyways whenever you're given an error message about dependencies. Only this wisely, as it won't work if you don't have the appropriate files installed. Also, use **rpm –qa** to see all installed packages.

The rpm Command Set

Use the **rpm** command to install or upgrade packages from a shell prompt.

Syntax: **rpm** [operation] [options] [package-files | package-names]

Common rpm Operations

Operation	Description
-i	Installs a package; system must *not* contain a package of the same name.
-U	Installs a new package or upgrades an existing one.
-F or –freshen	Upgrades a package only if an earlier version already exists.
-q	Queries a package - finds if a package is installed, what files it contains, and so on.
-V or -y or --verify	Verifies a package—checks that its files are present and unchanged since installation.
-e	Uninstalls a package.
-b	Builds a binary package, given source code and configuration files; moved to the rpmbuild program with RPM version 4.2
--rebuild	Builds a binary package, given a source RPM file; moved to the rpmbuild program with RPM version 4.2.
--rebuilddb	Rebuilds the RPM database to fix errors.
--**nodeps**	Performs no dependency checks. Installs or removes the package even if it relies on a package or file that's not present or is required by a package that's not being uninstalled.
--test	Checks for dependencies, conflicts, and

	other problems without actually installing the package.
--force	Forces installation of a package even when it means overwriting existing files or packages.

Important:

1. To verify that a package is installed, and display information about it, use **rpm –qi.**
2. To install or upgrade a package, use **–rpm –Uvh packagename.**
3. To upgrade a package only if an earlier version of it exists, use **rpm –Fvh packagename.**
4. To delete or uninstall a package, use **rpm –e packagename.**
5. To install a package ignoring its dependencies, use **rpm –i packagename --nodeps.**

Note: I won't cover the debian package for this guide, but I want you to know the tool used for debian is **dpkg**, and it has switches similar to rpm.

* You are responsible to find out the appropriate switches, and all other options.

The tar Command Set

The **tar** program is very complex, and it can be used for a lot of things, from compressing and extracting files, to backing up your system.

tar Command Switches

Command	Abbreviation	Description
--create	c	Creates an archive
--concatenate	A	Appends tar files to an archive
--append	r	Appends non-tar files to an archive
--update	u	Appends files that are newer than those in an archive
--diff or --compare	d	Compares an archive to files on disk
--list	t	Lists archive contents
--extract or --get	x	Extracts files from an archive
--exclude-from *file*	*X*	Excludes files listed in *file* from the archive
--gzip or --ungzip	z	Processes archive through gzip
--bzip2	j	(some older Processes archive through bzip2 versions used I or y)

Example: To extract files from a **tarball**, use the following command:

tar --extract --verbose --gunzip –file lynx-2.0.1.tar.gz

or

tar xvzf lynx-2.0.1.tar.gz

* As you can see, the second command does the same thing, and it's simpler.

Compiling and Installing from Source Code

Things you need:

1. **Appropriate compilers** – Normally, Linux uses the **GNU Compiler Collection** (**GCC**)
2. **Support libraries and header files –** Without the appropriate libraries and system files, you won't be able to compile anything.
3. **System Resources –** You obviously need to have enough RAM, and your CPU usage must be low.

* When compiling from source code, you usually need to type these commands:

1. **make** – To direct the compilation process.
2. **configure or ./configure** – To have the package auto-detect your computer's installed libraries and configure itself appropriately.
3. **make install or ./install** – To install the executables.

Starting Scripts or Executing Commands Automatically at Boot

* To have a script, command, or even a service start automatically, you can place the appropriate line to do so in **/etc/rc.d/rc.local** or **/etc/rc.d/boot.local**.

Starting and Stopping Services via SysV Scripts

When Linux starts, it enters one of several **runlevels**, each of which corresponds to a specific set of running services.

Level 0 – Reboots the machine
Level 1 – Used for single-user maintenance
Level 3 – Corresponds to a multiuser text-mode boot
Level 5 – Adds X to the mix (for **GUI** login prompt)

- Temporarily Starting and Stopping Services

* SysV startup scripts reside in particular directories – normally **in /etc/rc.d/init.d** or **/etc/init.d.** You may run one of these scripts, followed by an option like **start, stop, or restart** to affect the server's run status**.**

Example: **/etc/rc.d/init.d/named start**

- Permanently Starting or Stopping a Service

In addition to **/etc/rc.d/init.d** and **/etc/init.d** directories in which **SysV** scripts reside**,** Linux systems also host several other directories that contain symbolic links to these directories. These directories are named **/etc/rc.d/rcn.d** or **/etc/rcn.d,** where n is the runlevel number.

Within these directories, you will be files starting with letters **S and K.** In order to prevent a SysV script from starting up, replace the **S with a K** in the script's name**.**

* To adjust what services run in various runlevels, use the **chkconfig** command. For example:

chkconfig --level 5 named on

The previous command enables named in runlevel 5.

* Both **inetd** and **xinetd** are considered **superservers**. They are servers that listen for network connections intended for any of several other servers.

* You can control servers that launch via **inetd** through the **/etc/inetd.conf** file. For example, suppose that you have an entry in it that looks like this:

ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd –l

- This line starts the ftp server once inetd is started, to stop it simply add a # in front of it.

* You can control servers that launch via x**inetd** through the **/etc/xinetd.conf** file. For example, suppose that you have an entry in it that looks like this:

```
service ftp
{
    socket_type   = stream
    protocol      = tcp
    wait          = no
    user          = root
        server          = /usr/sbin/in.ftpd
        server_args     =-l
}
```

- This line starts the ftp server once xinetd is started, to stop it simply include the line **disable = yes** in a service definition.

Using *init* or *telinit* to Change the Runlevel

The init program is critical to Linux's boot process because it reads the **/etc/inittab** file that controls the boot process and implements the settings found in that file. It also sets the system's initial runlevel.

* You can use **telinit** to alter the runlevel after the computer has booted**.**
Syntax: **telinit** [-t time] runlevel

Note: You can discover what runlevel your computer is in with the **runlevel** command.

Shutting down the System

Use the **shutdown** command to shutdown or restart the system.

Syntax: **shutdown** [-t sec] [-arkhcfF] time [warning-message]

For example:

shutdown –r now

- The previous command causes the computer to restart immediately**.**

* You are encouraged to look at what each of the other options does on your own.

Permanently Changing the Runlevel

In order to permanently change the runlevel, you have to edit the **/etc/inittab** file.

For example, this file might look like this:

id:3:initdefault:

- Here you can clearly see it's starting in the **runlevel number 3**, to change it simply edit the number.

Running Jobs at Specific Times

Sometimes you might want to execute a command or run a script at a specific time, and that's where tools such as **cron**, and **at** can help you.

Cron is a daemon, which means it runs continuously looking for events that cause it to spring into action. It examines configuration files in **/var/spool/cron** and **/etc/cron.d** directories, and the **/etc/crontab** file.

Creating A System Cron Jobs

The **/etc/crontab** file controls the system cron jobs. Here's an example:

03 5 * * * root run-parts /etc/cron.daily

This line begins with five fields that specify the time. The fields are, in order, the minute(0–59), the hour (0–23), the day of the month (1–31), the month (1–12), and the day of the week (0–7; both 0 and 7 correspond to Sunday).

Note: An **asterisk** (*) matches all possible values.

- The previous command runs all scripts in **/etc/cron.daily** at 5:03am every day.

* You can read more about Cron here ->
http://www.deluxnetwork.com/linux/guides/crons.php

Using at

As I previously mentioned, the **at** command is also used to schedule jobs. This command takes a single option, a time.

Syntax: **at** time [date]

Note: Use the **atrm** command to delete a task, and the **atq** command to list the tasks for the current user.

* You can read more about the at command here ->
http://www.debianhelp.co.uk/schedulejobs.htm

Finding SUID or SGID Programs

You can **find** the find command to locate files with their **SUID** or **SGID** bits set. Remember these files allow normal users to run programs or commands as **root**, so they could be a thread if misused.

* Use **find / -perm +ug+s** to locate all **SUID** or **SGID** files on a computer.

Important: Be careful when setting **SUID** or **SGID** bits because they can be misused, and your system can be destroyed.

* Use **rpm –V packagename** to check for packages with the **SUID** or **SGID** bit set.

Listing and Managing processes in Linux

To list processes' status in Linux, use the **ps** command.

Syntax: **ps** [options]

Example: **ps -u john**

- The previous command displays all processes currently running owned by the user **john**.

Note: There are quite a few options for ps, so you are encouraged to look at them on your own. Also **ps** displays a lot more than the user's processes, it displays the cpu time, cpu priority, memory usage, etc.

* You can read more about **ps** here **->**
http://www.oreillynet.com/linux/cmd/cmd.csp?path=p/ps

Viewing CPU Usage in Real Time

To view the CPU and memory usage in real time, use the **top** command.

Syntax: **top** [options]

Example: **top –h**

The previous command displays help information.

Important: Hit **M** on the keyboard to change the display to sort by **memory usage**, and hit **P** to sort it by **CPU usage**, which is the default.

* You are encouraged to look at what each of the other options does for the top command on your own.

Viewing the Computer Uptime

To view the current uptime, simply type **uptime**

Restricting Processes' CPU Use

Use **nice** to launch a program with a **specific priority**, or use **renice** to alter the priority of a **running program.**

Syntax: **nice** [argument] [command [command-arguments]]

Example**: nice -n 11 myscript input.txt**

- The previous command runs a script called **myscript** at priority **11.** The file **input.txt** is also passed to the script.

Note: You can specify the **priority** in 3 ways, one of them was previously mentioned, and the remaining two are:
nice -11 myscript input.txt
nice –adjustment=11 myscript input.txt

- This command does exactly what the previous command did; it runs a script called **myscript** at priority **11.** The file **input.txt** is also passed to the script.

Note: If the adjustment value is omitted, the default is **10**.

Using renice

Syntax: **renice** priority [[-p] pids] [[-g] pgrps] [[-u] users]

Example: **renice 4 18912 –u alex don**

- The previous command sets the **priority** to **4** for **PID 18912**, and for all processes owned by **alex** and **don**.

Killing Processes

Use the **kill** command to terminate processes.

Syntax: **kill** –s signal PID

Note: The **-s signal** parameter sends the specified signal to the process. The most common signals are:

1. **SIGHUP** – This is represented by the number **1** and it causes many daemons to reread their configuration files.
2. **SIGKILL** – This is represented by the number **9** and it causes the process to exit without performing routine shutdown tasks.
3. **SIGTERM** – This is represented by the number **15** and it causes the process to exit but allows it to close open files.

Note: The **pid option** is of course, the **PID** of the process you want to kill.

Important: A variant on **kill** is **killall** , and it is used to terminate all processes.

Syntax: **killall** [options] [--] name […]

For Example: **killall vi**

- This command kills all running processes called **vi.**

Foreground and Background Processes

Use the **fg** command to restore a program you have temporarily paused to the foreground. For instance, say that you have used **Ctrl + Z** to pause a program because you needed to do something else, to return to that program, simple type **fg**.

Use **jobs** to obtain a list of jobs associated with a terminal. For instance, say that you have paused several programs, you can type **jobs** to view the process number, and then restore it to the foreground with **fg number**.

For Example: **fg 2**

- This command restores job number **2** to the foreground.

Use the **bg** command to send a process to the background. You can also send a process to the background by appending **&** at the end of the process.

For Example: **cat test.txt &**

- This command launches the **cat** command in the background, leaving you in control of your **xterm** window for other tasks.

- > Chapter 6 - Networking

Note: It is very important to be familiarized with Network+ objectives prior to reading this chapter. I will only go over some of the definitions briefly.

Basic Functions of Network Hardware

Network hardware is designed to enable two or more computers to communicate with one another. It also facilitates the transfer of data between computers.

Types of Network Hardware

1. **Ethernet** – Most common type of network hardware. Comes in several varieties ranging from 10Base2 and 10Base5 to 10BaseT and 100BaseT.
2. **Token Ring** – Clocks in at 16Mbps, although 100Mbps are also available. It is costlier than Ethernet, and has less hardware support.
3. **FDDI (Fiber Distributed Data Interface)** – It is a networking technology comparable to t100Base-T Ethernet in speed. Uses fiber-optic.
4. **HIPPI (High-Performance Parallel Interface**) – Provides 800Mbps or 1600Mbps speeds. Most commonly used to link computer clusters and supercomputers.
5. **LocalTalk** – Developed by Apple for Macintosh. Really slow at 2Mbps.

6. **Fibre Channel** – Supports both optical and copper media, with speeds between 133Mbps and 1062Mbps. Potential reach of approximately 10 kilometers.
7. **Wireless Protocols** – Most popular are **802.11b** (**Wi-Fi**), with speeds up to 11Mbps. Other standards include **802.11a** and **802.11g** with provide faster speeds. (Up to **54Mbps** if I recall correctly)

Central Devices

1. **Hubs** – Mirror all traffic to computers, and permit only half-duplex transmission.
2. **Switches** – Smart enough to send packets only to the intended destination, and permit full-duplex transmission.

Network Packets

Modern networks operate on discrete chunks of data known as **packets**. Typically, each packet includes an **envelope**, and **a payload**.

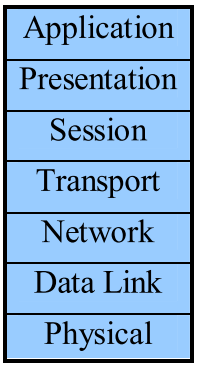
- There are different types of packets, for example Ethernet uses its own packet known as ***frame***.

Network Protocol Stack

The packing and unpacking of network data is frequently described in terms of a **protocol stack**.

The OSI Model

Open System Interconnection Model



Viewing Hardware Addresses

To view information about a specific network card in Linux, use the **ifconfig ethn** command, where **n** is a number starting at **0** which corresponds to the first NIC card.

For Example**: ifconfig eth0**

- The previous command displays information about the first NIC card in the system. Consequently, ifconfig **eth1** displays information about the second NIC card.

Viewing IP to Ethernet Address Mappings

Use the **arp** command to view IP addresses currently mapped to Ethernet Addresses (MAC Addresses for that matter).

For Example: **arp**

- The previous command displays a list of IP addresses and their corresponding MAC Address.

Important: Use **arp -a** to display all mappings.

Resolving Hostnames

To resolve hostnames, three utilities are available in Linux. They are **nslookup**, **host**, and **dig.**

Example: **nslookup proprofs.com**

Example 1: **host proprofs.com**

Example 2: **dig proprofs.com**

- All these three examples return proprofs' ip address**.**

***** You are encouraged to look at the different options to pass to **host** and **dig** on your own.

Important Files

/etc/hosts – This file contains all the static **IP->Host Mappings**.

Sample **/etc/hosts** file: **127.0.0.1 localhost**
192.168.1.18 linux.proprofs.com proprofs

- In the previous sample file, the name **localhost** is associated with **127.0.0.1** and the names **linux.proprofs.com** and **proprofs** are tied to **192.168.1.18.**

Important: Linux normally performs lookups in **/etc/hosts** before it uses **DNS.** You can modify this behavior by editing the **/etc/nsswitch.com** file and editing the **hosts** line**,** which lists the order of the files and **DNS options.**

DHCP
- Common **DHCP** clients in Linux are **pump**, **dhclient**, **and dhcpcd** (not to be confused with the Server, **dhcpd**)

- As it was previously mentioned, **dhcpd** is the Server.

Configuring Linux to use DHCP

- To configure Linux to use **DHCP**, add the **BOOTPROTO=dhcp** to the file called **/etc/sysconfig/network-scripts/ifcfg-eth0** in Red Hat for instance.

Using Linux to configure Static IP Addresses

- To have Linux use static ip addresses instead, add **BOOTPROTO=static** to the file **/etc/sysconfig/network-scripts/ifcfg-eth0**, and edit all other options.

* You are encouraged to look at all the lines in the aforementioned file, and understand what they do.

Important: In order for Linux to translate between IP addresses and hostnames, you must specify at least one **DNS** server in **/etc/resolv.conf**.

Bringing Interfaces Up and Down

- To bring an interface up in Linux, use the **ifconfig** command.

For Example: **ifconfig eth0 up 192.168.1.21 netmask 255.255.255.0**

- The previous command brings up **eth0** (**the first Ethernet card**), and links the specified **IP address** to it. It also assigns the specified **netmask**.

Note: It is very important to specify **"up" .**That's what tells **ifconfig** what to do with the **IP address.**

Example 2: **ifconfig eth0 down**

- The previous command marks the **eth0** interface as **down.** This simply means that the system **will not** attempt to transmit messages through that interface.

Note: Again it is very important to specify **"down".** This tells **ifconfig** to mark the interface down.

Adding a Default Route

- In order to add a **default route**, use the **route** command.

For Example: **route add default gw 192.168.1.1**
- The previous command adds a **default route** with a gateway of **192.168.1.1**.

Viewing the Routing Table

- In order to view the **routing table**, use the **route** command.

For Example: **route**

- The previous command prints the **current routing table** to the screen.

Delivering IP Addresses with DCHP

- If you want Linux to do deliver IP Addresses via **DHCP,** you must first install the DHCP server package, which is usually called **dhcp-server or dhcp**. This package normally includes a **SysV** script, such as **/etc/init.d/dhcpd.**

Note: The main DHCP server configuration file is **/etc/dhcpd.conf**, and you should be familiar with all the options presented on that file.

Delivering Hostnames with DNS

- To deliver hostnames with **DNS,** you must first install the DNS package, which is **BIND** (**Bekerley Internet Name Domain**). **BIND** installs a server under the filename **named**.

Note: The main DNS configuration file is **/etc/named.conf**, and you should be familiar with all the options presented on that file.

Configuring Mail Servers

- Configuring **sendmail** and **postfix** is outside of the scope of this guide, so you will have to read about it on your own.

Useful Links

1. **Configuring Sendmail** -> **http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch21_:_Configuring_Linux_Mail_Servers**
2. **Configuring Postfix** -> **http://www.postfix.org/BASIC_CONFIGURATION_README.html**

Managing Mail Queues

- To view the mail messages that are still waiting to be sent with sendmail, issue the command **sendmail –bp.**

- An equivalent command is **mailq**, and it does exactly the same as the previous command.

Note: If your **messages** regularly stay in the queue for very long, checking the mail log files in **/var/log/mail** may provide you with some clues.

- In order to have the mail server attempt to deliver all of the queued messages immediately, use the **sendmail -q** command.

Configuring Web Servers

- Configuring web servers is also outside of the scope of this guide. Simply know a few things about **Apache**, the most popular **Web Server** for **Linux.**

1. Its configuration file is usually found in **/etc/httpd/conf/httpd.conf.**
2. Once you edit the aforementioned file, you can specify many things such as the port it will listen on, the default location for it to serve web pages, etc.
3. You can also use **Virtual Hosts** with **Apache**. This simply allows one server that hosts multiple websites to share a single computer to do so.
4. **CGI scripts** usually reside on the **cgi-bin** directory.

Using X Programs Remotely

- Suppose that your network contains two machines. One computer is called **lion**, and the other one is called **zebra.** Now suppose that you want to sit at **lion**, and run programs located on **zebra**.

Steps to Follow

1. Log into **lion**, and start X is it is not already running. Also open a terminal such as **xterm**.
2. Type **xhost +zebra** in **lion's** terminal. This command tells **lion** to accept for the display in its **X** server data that originates from **zebra**.
3. Log into **zebra** from **lion**.
4. On **zebra**, type **export DISPLAY=lion:0.0**. This command tells zebra to use lion for the display of X programs.
5. After you're done, log off **zebra**, and type **xhost -zebra** on **lion**. This will tighten security.

Using an E-mail Client

- Using different e-mail clients is outside of the scope of this guide, please read about this on your own.

* Linux Supports a wide variety of e-mail clients. These include the following:

1. **Kmail** - (**http://www.kde.org**)
2. **Ximian Evolution** - (**http://www.novell.com/linux/ximian.html**)
3. **Mutt** - (**http://www.mutt.org**)

***** You are encouraged to read about these e-mail clients.

Note: When sending mail, you can compose it in a text editor and redirect it through the **mail** command, as in **mail support@proprofs.com < linux.txt.** You can even set a subject for the message with the **–s** option, for example: **mail –s "Linux Guide" support@proprofs.com < linux.txt.**

- You can also use the **mail** command to read your email, even though **pine** is preferred.

Configuring Routing

-Use the **route** command to add and delete routes.

Syntax: **route** { add | del } [-net | -host] target [netmask nm] [gateway gw] [reject] [[dev] interface]

For Example: **route add –net 173.20.0.0 netmask 255.255.0.0 gw 173.21.1.1**

- The previous command adds a route in which packets destined for the **173.20.0.0/16** subnet should be passed through the **173.21.1.1** router, which is not the default gateway.

Important: To enable Linux to act as a router and forward packets, you must modify a key file in the **/proc filesystem** as follows:

echo "1" > /proc/sys/net/ipv4/ip_forward

- The previous command enables **IP forwarding**. Permanently setting this option requires modifying **/etc/sysctl.conf**

Using NIS

- ***Network Information Service*** (**NIS**) is a protocol that's designed to simply user authentication and related services on a network of multiple Unix or Linux systems.

- Once you have told the NIS tools about your server, you must also configure linux to use NIS by editing the **/etc/nsswitch.conf** file.

Testing Basic Connectivity

- To test basic connectivity between hosts, use the **ping** command.

For Example: **ping yahoo.com** will ping yahoo to test connectivity.

Tracing a Route

- To trace a route between hosts, use the **traceroute** command.

For Example: **traceroute –n 10.2.3.4**

- The **–n** option to this command tells it to display target computer's **IP addresses** instead of **hostnames.**

Note: Use the **netstat** command to view information such as what ports are open, the hosts you're connected to, etc.

-> Chapter 7 - Security

Sources of Security Vulnerability

1. **Physical Access –** The last thing you want is someone walking down the aisle with your desktop, right?
2. **Stolen Passwords** – If a password falls into the wrong hands it can give the intruder access to the system. This can lead to greater issues when combined with local program bugs.
3. **Local Program Bugs –** The main risk comes from programs that are run with enhanced privileges – that is those that enable a **set user ID (UID)** or **set group ID (SGID)** bit.

   Important:

   - To search for **SUID** files, use **find / -perm +4000 –type f**
   - To search for **SGID** bits alone, use **find / -perm +2000 –type f**
   - To search for files that have their **SUID** and **SGID** bits set, use **find / - perm +6000 –type f.**

4. **Server Bugs** – Unlike local user program bugs, bugs in servers can cause security breaches even when the system has no local user accounts. For instance, a bug in the web server could enable a cracker to run arbitrary code as **root**.
5. **Denial of Service Attacks (DDOS)** – This type of attack doesn't need to involve an actual security breach; it simply denies you the use of your equipment by saturating it with packets.
6. **Encryption issues –** Unencrypted data can be intercepted and read on any intervening system, and sometimes on computers on the same network as the source or destination.
7. **The Human Element –** Many people are often victims of social engineering -- an attacker simply asks a legitimate user for a password pretending to be someone else.

Note: A particular type of social engineering is **phishing**. This involves sending bogus e-mails or setting up fake web sites that lure unsuspecting individuals into divulging sensitive financial information.

Physical Security

- Computer Security begins with physical security. If your computer is not protected against physical tampering or theft, then it becomes an easy target for abuse.

Note: An intruder with physical access can do just about anything with your computer so make sure your PC is secured.

Steps for Mitigating Damage from Physical Attacks

1. **Remove removable media** – A computer with no floppy drive, no Zip drive, and no CD drive will make it very harder for an intruder to be able to boot it.
2. **Restrict BIOS boot options** – Modern **BIOS** often have an option to enable or disable particular boot media, make sure you use this feature appropriately.
3. **Use BIOS passwords** – Setting a BIOS password will definitely delay the intruder's attack as they have to guess it.
4. **Secure the computer** – Replace normal screws with screws that required special tools.
5. **Secure the room** – Use locks on doors or any other type of security device.
6. **Use data encryption** – Believe it or not, others can see all the information that is transmitted if it is not encrypted.

Firewalls

Packet-filter firewalls – This type of firewalls work by blocking or permitting access based on low-level information in individual data packets, such as source and destination IP addresses and ports.

Proxy Filters – This type of firewalls work by partially processing a transaction, such as Web page access, and block or deny access based on high-level features in this transaction.

Linux Firewall Software

Linux uses **ipfwadm**, **ipchains**, and **iptables** tools to configure firewall functions.

Note: The 2.4.x and later kernels series include the ability to use older tools, but only as compile-time option.

- You can configure a firewall in any of several ways

1. **Manually** – Read up on the syntax of the tool used to configure your kernel and write your own script.
2. **With a GUI configuration tool** – Several tools are available to help you with this, some of them are **Firestarter** (**http://firestarter.sourceforge.net**) and **Guarddog** (**http://www.simonzone.com/software/guarddog**).
3. **With the help of a Website** – You can enter information about your system, and the following website generates a firewall script -> **http://linux-firewall-tools.com/linux/** **.**

Common Server Ports

- The **/etc/services** file lists service names and the posts associated with them.

Common Ports and Protocols

Port Number	TCP/UDP	Protocol	Sample Server Programs
20 & 21	TCP	FTP	ProFTPd, WU FTPd
22	TCP	SSH	OpenSSH, lsh
23	TCP	Telnet	in.telnetd
25	TCP	SMTP	sendmail, Postfix, Exim, qmail
53	TCP and UDP	DNS	BIND
67	UDP	DHCP	DHCP
69	UDP	TFTP	in.tftpd
80	TCP	HTTP	Apache, httpd
88	TCP	Kerberos	MIT Kerberos, Heimdal
109 and 110	TCP	POP (versions 2 & 3)	UW IMAP
111	TCP and UDP	Portmapper	NFS, NIS, etc
113	TCP	auth/ident	identd
119	TCP	NNTP	INN, Leafnode
123	UDP	NTP	NTP
137	UDP	NetBIOS	Samba
138	UDP	NetBIOS	Samba
139	TCP	NetBIOS	Samba
143	TCP	IMAP 2	UW IMAP

177	UDP	XDMCP	XDM, KDM, GDM
220	TCP	IMAP 3	UW IMAP
389	TCP	LDAP	OpenLDAP
443	TCP	HTTPS	Apache
445	TCP	Microsoft DS	Samba
514	UDP	Syslog	syslogd
515	TCP	Spooler	BSD LPD, LPRng, cups-lpd
636	TCP	LDAPS	OpenLDAP
749	TCP	Kerberos	MIT Kerberos, Heimdal

Note: Privilege ports have numbers below **1024.**

The Linux Packet Filter Architecture

- In the **2.4.x** and later kernels, Linux uses a series of "**tables**" to process all network packets it receives or generates. Each table consists of "**chains**", which are series of pattern-matching rules.

Note: We will be concentrating on the **filter** table, although there are others such as **nat**, and **mangle**.

***** The **filter** table consists of three chains: **INPUT**, **OUTPUT**, and **FORWARD**.

- All of the chains have a default policy. This policy determines what happens to a packet if no rule explicitly matches it. The default for a default policy is **ACCEPT**, which accepts the packets.

Important: For more secure configurations, change the default policy to **DROP** or **REJECT**.

- DROP reduces network bandwidth, and the system's visibility on the network.
- REJECT can improve the performance for some protocols, such as **auth/ident**.

Creating Firewall Rules

- Use the **iptables –L –t filter** to list the current configuration of the **filter** table.

- To flush all rules from the **FORWARD** chain and change the default policy for that chain to **DROP** use the following commands:

iptables –t filter –F FORWARD
iptables –t fitler –P FORWARD DROP

- As you can see, **-F** is used to flush the rules, and **-P** to modify the default policy.

Note: To add rules to a chain, use the following command: **iptables [-t table] –A chain selection-criteria –j TARGET**

- The **selection-criteria** can be any of the following:

1. **Protocol** – The **--protocol** or **-p** option allows you to specify the protocol, they can be **tcp**,**udp**,**icmp,** or **all.**
2. **Source Port** – The **--source-port** or **-sport** option allows you to specify the source port. You can also specify a range of ports, for example **1000:3000** indicates ports from **1000** to **3000.**
3. **Destination Port** – The **--destination-port** or **–dport** works much like **the --source port** option.
4. **Source IP Address** – The **--source** or **-s option** filters on the **source IP address.** You can specify a single IP address, or an entire subnet such as **192.168.1.0/24.**
5. **Destination IP Address** – The **--destination** or **-d** option works just like the **--source** or **-s** option.

***** Other options do exist, and you are encouraged to look them up on your own.

General Example: **iptables –A input –p udp –dport 25 –s 192.168.1.0/24 –j ACCEPT**

- The previous command opens traffic to **UDP** port **25** from the **192.168.1.0/24** network.

Controlling Access via TCP Wrappers

- TCP Wrappers is configured through two files, **/etc/hosts.allow** and **/etc/hosts.deny.**

Note: If a system that is listed in both files, **/etc/hosts.allow** takes **precedence.**

- As the name suggests, **/etc/hosts.allow** contains a list of computers that are **allowed** access to the system in a particular way, and **/etc/hosts.deny** contains a list of computers that are **denied** access to the system.

Important: Both files use the same basic format -> **daemon-list : client-list**

Note: The **daemon-list** is a list of servers, and the **client-list** is a list of computers which are either allowed or denied access to the servers.

For Example: **192.168.1. EXCEPT 192.168.1.3** when placed in **/etc/hosts.deny** file blocks all computers in the **192.168.1.0/24** network **except** for **192.168.1.3**.

* You are encouraged to read more on this topic.

Symptoms of Intrusion

1. **System Slowdown –** Intruders might run programs in your computer so it might respond slowly.
2. **Increased network activity –** Intruders can use your computer to perform **DDOS** attacks against other computers.
3. **Changed program behavior –** A sudden change in a program's behavior could indicate that it has been slightly modified by a hacker.
4. **System or software crashes** – If your system or a particular program starts crashing all of the sudden, some files could have been modified by an intruder.
5. **Mysteriously altered data files** – If ordinary data files are changed without your concern, one possible explanation is intrusion.
6. **Missing or corrupted log files** – Intruders often delete or corrupt log files to avoid being detected. Detection if any of this could mean that your system has been compromised.
7. **Off-site complains** – If individuals from other sites contact you about attacks from your side or any other suspicious behavior, your system could have been compromised.

Intrusion Detention

Using Snort (http://www.snort.org) – Snort is a very powerful packet sniffer -- a program that monitors packets directed to its host computer (or to other computers on the local network).

- **Snort** can also function in a more sophisticated rule -- as an **intrusion detection system** (**IDS**)

Important: **Snort** is configured through the **/etc/snort/snort.conf** file. It can be launched by typing **snort**, and it logs messages to **/var/log/snort**.

Using PortSentry (http://sourceforge.net/projects/sentrytools/) – PortSentry is another **IDS** (Intrusion Detection System) which is similar to **snort**, in the sense that both are designed to alert you to suspicious network activity. It also runs on individual computer system to monitor access attempts to their own port; whereas snort can monitor entire networks.

Note: **PortSentry** can actively block network scans, so in some sense, **PortSentry** is more like a firewall than an **IDS**.

- After installing **PortSentry**, you can configure it by editing its **portsentry.conf** file, which is usually located in **/etc** or **/etc/portsentry**.

Using Tripwire (http://www.tripwire.org) – Tripwire is a utility that records a set of information about all the important files on a computer, including various types of checksums and hashes. You can compare this information later and see if a file has been modified.

Note: **Tripwire** is controlled through two configuration files; **tw.cfg** and **tw.pol**, which often reside in **/etc/tripwire**.

Important: Issue the command **tripwire --init** to have it generate initial checksums and hashes on all of the files it's configured to monitor.

Note: Issue the command **tripwire --check** to check the current state of the system against the database. Also, issue the command **tripwire --update** to update the database.

Using chkrootkit (http://www.chkrootkit.org) – chkrootkit is the closest thing in the Linux world to a Windows virus scanner. It scans the files to see if anything has been compromised, or if a root kit has been used.

Note: To run chkrootkit, simply type **chkrootkit.**

Using Package Manager Checksums

- Package managers – most notably RPM maintain checksums on all their installed packages. As a result, they can be used as intrusion detention tools.

For Example: **rpm -V lynx**

- The previous command outputs the files that have changed in some way.

Monitoring Log Files

Log files can provide clues to intrusions so they should be monitored regularly. Things you should look for include:

1. **Suspicious logins** – A user that's on vacation with no network access should not be logging in.
2. **Repeated login failures** – This can imply that a brute-force attack was attempted.
3. **Missing Entries** – As previously mentioned, hackers often try to hide their presence by deleting entries in log files. Be very careful about this!
4. **Entries for servers that should not be running** – If you're certain that a server should not be running but you found that it was run; find out the reason behind it.

Checking for Open Ports

- One tool that can be helpful in spotting stray servers is **netstat**. For example issue the command **netstat –ap** for the purpose of spotting stray servers.

Using Remote Network Scanners

- Network scanners, such as **Nmap** (**http://www.insecure.org/nmap/**) or **Nessus** (**http://www.nessus.org**) can scan for open ports on both, the local computer, and on remote computers.

Note: **Nessus** can also check for known vulnerabilities, so if you leave it running long enough it can actually inform you about them.

Note 2: You can issue the command **nmap -sT** linux.proprofs.com to have **Nmap** scan the target machine

Reviewing Accounts

- From time to time, it is critical to review the system and user accounts in order to search for suspicious entries. Some of them might include:

1. **Unknown accounts** – If you're the only system administrator and you come across an account you have not added, changes are your system has been compromised.
2. **Accounts with UID of 0** – Linux uses **UID of 0** to represent **root**, so any accounts with this UID have root privileges on your system.
3. **System accounts with passwords** – Normally these accounts do not require passwords, if you see a system account with a password; one possibility is that your system might have been compromised.

Imposing User Resource Limits

- Imposing limits is possible through a module called **pam_limits.**

Note: To impose limits, it is required to edit the **/etc/security/limits.conf** file. This file consists of four fields.

The domain – This field describes the entity to which the limit applies. It could be **usernames, groups** in the form of **@groupname**, or an **asterisk (*)** which matches **everyone.**

Hard or soft limits – Hard limits are imposed by administrators and cannot be exceeded; soft limits are imposed by users and can be temporarily exceeded. Use a **dash** (**-**) to signify both, a hard and soft limit.

The limited item – The item field specifies the type of item being restricted. It can be **cpu**, **core**, **data**, **fsize**, **maxlogins**, etc.

The value – The final field specifies the value that's to be applied to the limit.
For Example: **@friends hard cpu 3**

- The previous command applies a hard **CPU** limit of three minutes to the **friends** group. Members of this group can run programs, but if a program runs longer than three minutes it will be terminated.

-> Chapter 8 - Documentation

Documenting the Installation

-Your documentation should begin with information about the core system installation. This information will help if you ever need to reinstall the OS from scratch, or reproduce the configuration of an existing system.

Information you should include:

1. **Linux distribution and version –** If a new administrator comes in, it will be useful for him to know this sort of information.
2. **Hardware selections –** Should a specific piece of hardware fail, you would know what to look for.
3. **Disk partitions –** The hard disk partitioning scheme can be quite important. This information can be vital if any given partition becomes corrupt. To find information about partitions on a particular hard disk, use **fdisk –l /dev/hda > ptable.txt.** This command records all the partition information from **/dev/hda to** a file named **ptable.txt**.
4. **Installed software –** When installing Linux from scratch, it is useful to record what packages you chose to install. On system where Linux has been previously installed, use **rpm –qa > packages.txt** to generate a report with all packages currently installed on the system.
5. **Install**-time configuration options – During installation, make changes are made. It is important to record these changes for future reference.

Maintaining an Administrator's Log

- It is important to record certain things on a notebook when you're administering a system. These things include:

1. **Initial configuration** – Very important to have for future reference.
2. **Package installations** – Sometimes we install many different packages from tarballs and we forget to record what we had installed. This is one of the reasons why we should record this type of information as installing packages from tarballs leave no record of such packacges.
3. **Configuration file edits –** Record anything that you edit in a configuration file, and please summarize the changes. This will come in handy should problems begin to occur.
4. **Filsystem changes –** Save any filesystem changes such as moving programs around, and resizing partitions.

5. **Kernel recompilations** – It is very important to record the kernel version when recompiling the kernel, that way you know what you're dealing with should bugs be found.
6. **Hardware changes** – Whenever you add, remove, replace, or reconfigure hardware please make sure you record these changes.

Note: Please do **not** record any passwords in a log, especially the **root** password.

Note: From time to time it is very important to back up important configuration files. These files often reside on the **/etc directory** so issuing the following commands will create a backup of the whole directory:

mount /dev/fd0 /mnt/floppy
tar cvfz /mnt/floppy/backup.tgz /etc

***** You should be familiar with these two commands by now so I won't explain what they do.

Tools to Establish Normal Performance

Note: To document the **CPU load**, use the **uptime** command. This command tells you how long the system has been running, along with three load averages. These three values correspond to the **CPU** use over the past **1**, **5**, and **15** minutes.

Note 2: To document the **Memory Load**, use the free command. This command reports the total memory available, the memory in use, and the total amount of free memory.

Note 3: To document **Disk Use**, use the **df** command. This command displays all partitions currently mounted, and the total space used and available on them.

Collecting System Statistics

- To collect system statistics and performance as well, you can use a tool named **sar.**

Note: This tool might not be installed in your system by default, so you would need to install a package called **sysstat.**

Important: While the **sar** program accepts many options, usually you just specify an **interval** in **seconds**, and the **number** of **samples** you want to collect.

For Example: sar 1 5

- The default output shows the **CPU use**. This command collects **5** samples at **1** second intervals of the CPU usage.

* Many other options are available for **sar**, you are strongly encouraged to look at them on your own.

Log Files

- The traditional Linux system logger is **syslogd**, which is often installed from a package called **sysklogd.** This daemon handles messages from servers, and other user-mode programs.

Note: Once installed, the daemon must be configured. Use the **/etc/syslog.conf** file to configure it.

Logging Options

- It is important to be familiar with the format of **/etc/syslog.conf** so here's the syntax:

facility.priority action

- In the previous line, **facility** is a code word for the type of program or tool that has generated the message to be logged**; priority** is a cord word for the importance of the message; and **action** is a file, a remote computer, or other location that's to accept the message.

Note: Valid codes for **facility** are **auth, authpriv, cron, daemon, kern, lpr, mail, mark, news, security, syslog, user, uucp, etc.**

Note: Valid codes for **priority** are **debug, info, notice, warning, warn, error, err, crit, alert, emerg, and panic.**

* The **error**, **warn**, and **panic** priority names have been deprecated, so use their equivalents instead (**warning = warn, error = err, and panic = emerg**).

Important: Priorities from **highest** to **lowest** -> **emerg, alert, crit, err, warn, notice, info, debug**)

Important: When you specify a priority of **alert** for example, the system will log messages classified as **alert** and **emerg** as it logs messages of the specified priority or higher.

Note: To overcome the aforementioned problem when logging, you can precede the priority code by an **equal sign =,** as in **=notice**, which in turns tells the system to log only this type of messages.

Important: Also an **exclamation mark** (**!**) reverses the meaning of a match. For instance, **!warning** causes messages **below** warning priority to be logged. At last, an **asterisk** (*****) refers to all priorities.

Note: You can specify multiple selectors for a single action by separating them by a **semicolon** (**;**).

- Most commonly the *action* is a filename, usually in **/var/log**. Some other possibilities exist though, for example **/dev/console** displays the data on the screen, a **remote** machine preceded by an at-sign (**@**), and a list of **users currently logged** into the system, specified by an **asterisk** (*).

Example 1: **mail.*** **/var/log/mail** – This line sends all log entries referring to **mail** to a file named **/var/log/mail**.

Example 2: ***.emerg** * - This line sends all emerg-level messages to all users currently logged in the system.

Example 3: **kern.warn** **@logs.proprofs.com** – This command sends all **warning** messages from the **kernel** to a **remote system**.

* You are encouraged to read more about this on your own because it is critical that you know logging.

Rotating Log Files

- The most common log rotation tool is a package called **logrotate. This** program is called regularly via a **cron job**, and its configuration file resides in **/etc/logrotate.conf.**

* Configuring this file is outside of the scope of this guide, you are encouraged to read more about it on your own, though.

Reviewing Log Files to Identify Problems

- Reviewing log files can be very useful for a system administrator. You can review the logs to verify heavy loads, detect intrusion, monitor normal system functioning, find missing entries, view error messages, etc.

Note: As it was previously mentioned in other chapters, you can use **head** or **tail** to view the first and last lines of a file. For example, take a look at the following commands:

head -n 30 top.txt – This command displays the first **30** lines of a file called **top.txt**
tail –n 30 end.txt – This command displays the last **30** lines of a file called **end.txt**

Using Man Pages

- Man pages provide succinct summaries of program functions. They can be accessed simply by typing **man,** followed by the name of a command.

Section Number	Description
1	Executable programs and shell commands.
2	System calls provided by the kernel.
3	Library calls provide by program libraries.
4	Device files (usually stored in /dev)
5	File Formats
6	Games
7	Miscellaneous
8	System administration commands (run by root)
9	Kernel routines

Note: Keep in mind that **man passwd**, and **man 5 passwd** will display different results.

Important: Another command to find more information about a specific command is **info.** This command differs from man in that it displays a pretty neat output, usually a **hyperlinked format.**

Syntax: info command

For Example: **info ls**

Important: To find the latest **kernels** from source code**, use ->** **(http://www.kernel.org)**

Note: Another resource that's extremely helpful in finding information about Linux is the **Linux Documentation Project (LPD; http://www.tldp.org)**

VERY Important: Kernel sources are usually located in **/usr/src. Documents** on the other hand are often located in **/usr/doc** or **/usr/share/doc.**

-> Chapter 9 - Hardware

Common Resource Settings

Device	Common IRQs	Common DMA Channels	Common I/O Ports (Hexadecimal)
System Timer	0	N/A	0040–005F
Keyboard Controller	1	N/A	0060–006F
Second Interrupt Controller (for IRQs 8–15)	2	4	N/A
Real-Time Clock	8	N/A	N/A
Math Coprocessor	13	N/A	00F0-00FF
PS/2 Mouse Port	12	N/A	N/A
RS-232 Serial Port 1 (/dev/ttyS0)	4	N/A	03F8–03FF
RS-232 Serial Port 2 (/dev/ttyS1)	3	N/A	02F8–02FF
Parallel Port 1 (/dev/lp0)	7	3	0378–037F or 03BC–03BF or 0778–077F
Parallel Port 2 (/dev/lp1)	5	N/A	0278–027F or 0678–067F
USB Port	9 or 10	N/A	FF80–FF9F
SoundBlaster-Compatible Sound Card	5	1, 5	0220–0233, 0240–0253, or 0260–0263
Floppy Disk Controller	6	2	03F0–03F5
ATA Controller	1	14	01F0–01F7
ATA Controller	2	15	0170–0177

Note: Once Linux boots, you can check the hardware resources as follows:

cat /proc/interrupts displays the IRQ use information.
cat /proc/dma displays the DMAs used by devices.
cat /proc/ioports displays the I/O port use.

Configuring Power Management

- Two sets of system power management utilities exist**: Advanced Power Management** (**APM**), and **Advanced Configuration and Power Interface (ACPI).**

* Both of these rely on kernel features to work, and Linux can either use **APM** or **ACPI**, but **not both**.

APM

- In order to use **APM**, you need a package called **apmd**, which ships with most Linux Distributions. **APM** uses a daemon called **apmd**, which runs in the background waiting for an event to occur.

Note: Once running, apmd monitors the system's battery status, and if the battery's charge is too low, it kicks the system into a **suspend** mode.

* To control **APM** features manually, you can use the **apm** utility. By typing this command you can control the basic power management information, such as how much battery is left. You can also use switches such as **-s** and **-S** to have the system go into suspend and standby modes respectively.

Note: **Suspend mode** shuts off power to most devices, leaving only the **CPU** and **memory** operating. **Standby** leaves more devices powered up, which provides faster system recovery.

ACPI

- **ACPI** uses a daemon called **acpid**. This daemon is controlled through files in **/etc/acpi/events** directory.

Note: Files in **/etc/acpi/events** usually begin with a line such as **event=**, followed by another line such as **action=**.

* Events are not standardized, so you need to check your own system in order to determine what important event names are. To do so you can check the **/proc/acpi/event** file when you perform an **ACPI-related** event, such as closing the lid of a laptop.

PCMCIA Devices

- Because laptops don't have **ISA** or **PCI** slots, they have to use **PCMCIA** (**Personal Computer Memory Card International Association**).

Note: Card Services are controlled through configuration files in **/etc/pcmcia**. This directory contains scripts for different types of services, such as Network and IDE.

* You are encouraged to read more about these type of cards on your own, please use the following website to do so -> (**http://www.pc-card.com**).

Configuring USB Devices

- To support **USB** devices, Linux provides a set of drivers in the **USB** support section of the kernel configuration. You must compile **USB** drivers for your particular device in order to use it.

Note: Some **USB** devices are supported through a special **/proc** filesystem directory, **/proc/bus/usb**.

Linux Printing

- Linux printing is built around the concept of a **print queue**. This is a sort of holding area where files wait to be printed.

- Users submit print jobs via a command called **lpr**. This command sends the print job into a specified queue.

- This **queue** corresponds to a directory on your hard drive, typically in a subdirectory of **/var/spool/lpd** or **/var/spool/cups** directory.

Note: Because Linux printing run as daemons, they must be started before they're useful. This is normally done through scripts in **/etc/rc.d** or **/etc/rc?.d** (where **?** is a runlevel number). You should look up scripts that contain strings such as **lpd**, or **cups** in their names to learn what the system is running.

* You can also find out what you system is running by using the **ps** command. For example:

ps ax | grep cups will reveal whether your system has **cups** running or not.

Note: **The /etc/printcap** is the heard of the **LPRng** system. This file contains a list of printer definitions.

* You are encouraged to read more about this particular file on your own, to do so please use the following link - > (**http://sunsite.ualberta.ca/Documentation/Misc/LPRng-3.5.2/LPRng-HOWTO-5.html**)

* After you have reconfigured your print queues, you may need to restart your printer daemon. This can be accomplished by issuing the following command as **root**:

/etc/rc.d/init.d/lpd restart

Configuring CUPS

- **CUPS** doesn't rely on the **/etc/printcap** configuration file as much. It uses its own configuration files in the **/etc/cups** directory.

Note: You can add and delete printers in **CUPS** by editing the file **/etc/cups/printers.conf** which contains a list of printer definitions. You may also need to change some settings for **CUPS**, and this can be accomplished by editing its main configuration file **/etc/cups/cupsd.conf**.

* You are encouraged to look at what each lines does in these configuration files on your own. Configuring **CUPS** is outside of the scope of this guide.
Important: To configure **CUPS** using the web, you can enter **http://localhost:631** in a Web browser on a computer running **CUPS**.

Monitoring and Controlling the Print Queue

- To submit jobs and to examine and manipulate a Linux print queue, you can use the following utilities: **lpr, lpq, lprm, and lpc**. All these commands can take the **-P** parameter to specify the print queue on which they operate.

For Example: **lpr -Pproprofs -m pprof welcome.txt**

- The previous command sends an email to the user **pprof** when **welcome.txt** prints in the **proprofs** queue, should it be busy at the time the print job is sent to it.

* You are encouraged to look at the different parameters you can pass to lpr on your own.

Viewing Print Queues

- In order to view print queues, you the **lpq** command.

For Example: **lpq -Pproprofs**

- The previous command displays the current **proprofs'** queue.

Removing Print Jobs

- To remove a print job, you can use the **lprm** command.

For Example: **lprm 718**

- The previous command removes job number **718** from the queue.

Note: When a **username** is passed to lprm, it removes all print jobs for that particular **username**.

Important: If a user passes a **dash** (**-**) to **lprm**, it removes all print jobs for that particular user. If the **root** passes a **dash** (**-**) to **lprm**, it removes all print jobs for all users.

Controlling the Print Queue

- The **lpc** command **starts, stops, and reorders** jobs within print queues.

For Example: **lpc topq sister 718**

- The previous command adjusts job number **718** on the **sister** queue so that it is the first to print.

* You are encouraged to look at what each of the parameters for **lpc** does on your own.

Scanner Hardware

- **Flatbed scanners –** Most common type of scanners. Typically larger than a notebook, with a horizontal glass plate on which you place a document you want to scan.
- **Hand scanners –** Less expensive than **flatbed** scanners. Produce poor results, and are very portable.
- **Film scanners –** Designed to scan film. Smaller than flatbed scanners, but with very high resolutions.
- **Drum scanners –** High in quality and price. Rotates the document to be scanned on a high-speed cylinder.

Linux Scanner Software

- **SANE (The Scanner Access Now Easy) –** This is the primary Linux scanner tool. More info can be found here -> (**http://www.sane-project.org**)
- **VueScan –** This program is a shareware program intended for film scanning. More info can be found here -> (**http://www.hamrick.com**)
- **OCR Shop –** Designed to perform optical character recognition (**OCR**) in Linux. More info can be found here -> (**http://www.vividata.com**)

For Questions or Comments visit:
ProProfs Free Linux+ Certification Section