Excerpted from:

The Accidental Administrator: Linux Server Step-by-Step Configuration Guide

First Edition

by Don R. Crawley
Linux+ and CCNA Security

Provided courtesy of

soundtraining.net
accelerated i.t. training

Seattle, Washington

www.soundtraining.net

SBN-10: 1-4536-8992-3
ISBN-13: 978-1-4536-8992-9

Chapter Two:
Linux Administration

"If you want to travel around the world and be invited to speak at a lot of different places, just write a Unix operating system."

--Linus Torvalds

GUI vs. CLI

Since the first graphical user interface (GUI) was created in the Xerox Palo Alto Research Center in the early 1970s, those of us who work in information systems and technology have debated its merits and liabilities.

The real issue is not whether to use a GUI or a command-line interface (CLI); it is about choosing a tool that works for you and helps you work most effectively. For most of us, that means that sometimes we'll use a GUI and sometimes we'll use a CLI.

I once had a student in a Linux workshop who said his nickname was "No GUI Louie". While I remember Louie as a very knowledgeable and capable IT pro, I have also had students who avoid the CLI because of its complexity without considering the power it affords an administrator. I think we limit ourselves when we arbitrarily limit the tools at our disposal by eliminating GUI or CLI tools. In my own work, I find that I use both the CLI and GUI, depending on the task at hand and my personal familiarity with the tools in question. (Okay, I use the command-line most often, but I'm very grateful for the GUI when performing unfamiliar tasks!)

GUI Pros	GUI Cons
Faster	Farther away from the "road"
Fewer typing errors	Less control
Less minutia	Java and other issues might make GUI unavailable
Safer (harder to make mistakes)	Some of the names and labels it creates are strange
Can help teach you CLI commands	Some people are more familiar with the CLI

Linux Directories

Everything in Linux/UNIX is based on the file system. The file system is comprised of various directories (Windows calls them "folders".) The root directory ("/") is at the base of the file system. Some directories may be on different partitions or drives, but they are still a part of the file system. Some directories may even be on completely different computers, perhaps running a completely different operating system, but they are still part of the file system. What follows is a list of some of the more commonly found directories in the Linux file system (all directories are not included on all systems):

- / is the root directory
- /bin/ and /usr/bin/ store user commands.
- /boot/ contains files used for system startup including the kernel.
- /dev/ contains device files

- /etc/ is where configuration files and directories are located.
- /home/ is the default location for users' home directories.
- /initrd/ is used to load required device modules and mount the initrd.img image file during system startup.
- /lib/ and /usr/lib/ hold library files used by programs in /bin/ and /sbin/.
- /lost+found/ holds orphaned files (files without names) found by **fsck**
- /mnt/ holds the mount points for file systems that were mounted after boot.
- /opt/ is used primarily for installation and unintallation of third-party software. Holds optional files and programs.
- /proc/ is a virtual directory (not actually stored on the disk) which holds system information required by certain programs.
- /root/ is the home directory of the superuser "root"
- /sbin/ and /usr/sbin/ store system commands.
- /tmp/ is the system temporary directory. All users have read+write access to /tmp/.
- /usr/ contains files related to users such as application files and related library files ("usr" is an acronym that stands for UNIX system resources).
- /var/ (as in "variable") holds files and directories that are constantly changing such as printer spools and log files.

The above is a brief overview of Linux/UNIX directories. For a more complete discussion of Linux/UNIX directory structures, search on "Filesystem Hierarchy Standard" at www.wikipedia.com.

Soundthinking point: Customizing the Desktop Background

Customize the Gnome or KDE background by right-clicking on the background and choosing **Change Desktop Background**.

Starting and Stopping Services

In Linux, the various services that together make up the entire operating system are called daemons (pronounced DEE-muns). There are daemons for the DNS name server (named), the Web server (httpd), DHCP (dhcpd), and so on. When you see odd names ending with the letter "d", you're most likely looking at a daemon name. Most of the daemons can be controlled through scripts located at /etc/init.d/. For example, to start the Web server from a terminal window, you would execute the command "/etc/init.d/httpd start". To stop it, you would execute the command "/etc/init.d/httpd stop".

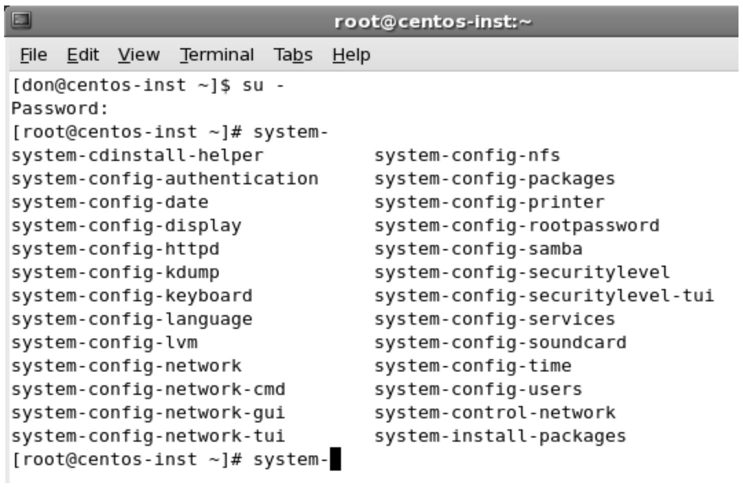
Different Linux distros might place the scripts in slightly different locations. Later in the book, you'll learn how to use the *find* command to locate such scripts as well as other files and directories.

The Red Hat System-Config and System-Control Utilities

Red Hat-based systems include the system-config and system-control utility, a text and graphics-based utility to help you configure and manage various aspects of your system. You can see the available utilities by opening a terminal window and, as root, typing the following command:

```
#system-
```

Do not hit enter, but instead touch the tab key twice and you'll see a listing of options.



```
[don@centos-inst ~]$ su -
Password:
[root@centos-inst ~]# system-
system-cdinstall-helper          system-config-nfs
system-config-authentication     system-config-packages
system-config-date               system-config-printer
system-config-display            system-config-rootpassword
system-config-httpd              system-config-samba
system-config-kdump              system-config-securitylevel
system-config-keyboard           system-config-securitylevel-tui
system-config-language           system-config-services
system-config-lvm                system-config-soundcard
system-config-network            system-config-time
system-config-network-cmd        system-config-users
system-config-network-gui        system-control-network
system-config-network-tui        system-install-packages
[root@centos-inst ~]# system-
```

Figure 1 The Red Hat system utility

The Shell

The shell is the interface between the user and the operating system. It acts as a keyboard interpreter, taking the keyboard input from the user and delivering it to the operating system. You can think of the shell as being the part of the operating system that allows you to interact with the kernel. The shell is the program that executes Linux or UNIX commands.

There are several shells available for use in the Linux and UNIX. The one most commonly used in Linux is the bash shell (Bourne Again Shell). Other shells include sh (Bourne Shell), csh (CShell), tcsh, csh, and ksh (Korn Shell).

You can change the shell by typing the shell's name at a command prompt.

For the purpose of this document, we'll focus on the bash shell.

The shell is not only a command interpreter, it is also a scripting language. Experienced Linux administrators often write scripts to automate frequently conducted system tasks.

Linux, like all multi-user operating systems, has an administrator account which is used for system configurations and operations. In Linux/UNIX, the administrator account is called "root" (equivalent to "admin", "administrator", or "supervisor" in other operating systems). "root" is often referred to as the "superuser" because of the account's unrestricted access to every area of the system and every aspect of the system's configuration.

When logged on as root using the bash shell, the prompt is a pound sign (#). When logged on as a regular user using the bash shell, the prompt is a dollar sign ($).

Shell commands in the Linux/UNIX world are usually case sensitive.

You can see your default shell with this command:

```
#echo $SHELL
```

Linux Profiles

There are two types of Linux profiles: system-wide and user-specific. System-wide configurations affect all users, while user-specific configurations affect only a single user. Normally, you must be root to change system-wide configurations.

User-Specific Profiles

User-specific profile settings are found in the user's home directory (/home/donc), but they're hidden by prepending a "." to the filename. Examples of profile files include:

- .bashrc
- .bash_profile
- .bash_history

There are many others. You can view the hidden files in any directory by using "ls -a".

System-Wide Configurations

System-wide configuration settings are almost entirely found in /etc. This is where you find files for configuring Apache, BIND DNS, and nearly any other aspect of Linux. For example, in Debian Linux, if you want to modify settings of your Apache web server, you would probably modify /etc/apache2/apache2.conf. If you are working with a Red Hat product, the file most likely is /etc/httpd/conf/httpd.conf. (The reason for using tentative language is because everything is configurable in Linux and the person who built your Linux system might have chosen to place the configuration files elsewhere.)

Administration Tools and Techniques

Working in Terminal

Most Linux systems configured as servers are managed in a command-line interface (CLI) and many Linux power-users prefer to manager even their desktop system in the CLI. Although the graphical user interface (GUI) tools available for use in many Linux distros have improved considerably over past versions, the CLI continues to provide the greatest power and flexibility for configuring and managing a Linux system. The other benefit to working in a CLI is that each Linux distro is much more similar in the CLI than in the GUI. For the purpose of this workshop, you will do most of your configurations in the CLI, thus allowing you to make smoother and simpler transitions from Debian to other distros such as SuSE, Ubuntu, RedHat, Slackware, or even traditional UNIX systems.

soundthinking Point: Desktop Environments

The default desktop environment in CentOS 5.x, Debian 4.0r5, and many other Linux distros is Gnome. The desktop environment is what provides the buttons, backgrounds, effects, and applications. Desktop environments will be covered later in the workshop, but for now understand that different desktop environments may have slightly different to significantly different menu locations than those described here.

Student Exercise 2.1: Commonly-Used Shell Commands

When you first logon to a Linux system, you may be in a GUI or in a command-line shell. If you are in a GUI, you can open a terminal window (a command-line shell) by clicking on Applications in the menu at the top of your desktop, then mouse over Accessories, and click on Terminal. Once you are in a command-line shell, you are placed in your home directory (/home/[username]). You can navigate to other directories by using the "cd" command, followed by the path to the desired destination.

1. If you are not already using the root account, use the su (switch user) command with the "-" switch to change to the root account and profile:
```
$su -
Password: p@ss5678
```
2. **#cd** changes the working directory. Enter the following command (bearing in mind that the "#" represents your prompt:
```
#cd /home
```
Notice that the prompt changes to display the current directory (home).

3. To return to your own home directory, type the following command:
```
#cd ~
```
You can also type "cd" to return to your home directory, but you should know that the tilde (~) represents your home directory. The tilde is often used in path statements to represent your home directory.

4. Now, enter the "pwd" command to print your working directory to your screen (output directed to the screen is known as standard output or stdout):

```
#pwd
```

5. You can go up one level in the directory hierarchy by using the command:

   ```
   #cd ..
   ```

   The ".." indicates the parent directory. All directories except for the root (/) directory have a parent.

6. Once again, enter the "pwd" command to print your working directory:

   ```
   #pwd
   ```

7. Once again, return to your own home directory. This time, simply enter "cd" with no tilde:

   ```
   #cd
   ```

8. Now issue an "**ls**" command to see the contents of the current directory. "ls" lists the contents of a directory.

   ```
   #ls
   ```

There are a variety of switches available for use with "ls":

Some commonly used options with "ls"	Results
ls –a	Lists all files including hidden files
ls –l	Long listing, includes permissions, owners, groups, etc.
ls –R	Lists sub-directories recursively
ls –sh	Shows file size (s) in human-readable format (h)
ls -1 (the number "1")	Displays one file per line
ls –d	Tells "ls" to list directory names, but not their contents

You can also use common shell metacharacters with "ls":

- * is the string wildcard
- ? is the character wildcard
- [] encloses a character set
- [-] is a character range
- { } is a string set

When you issue the "**cd**" command with no parameters, you will be returned to your home directory.

#**mkdir** creates directories.

Student Exercise 2.2: Creating Directories and Files

In this exercise, you will create a working directory which you will use for upcoming exercises. You will work with several commands to become familiar with some of the important tools related to directory and file management.

Student Exercise 2.2.1: Working with Directories

1. Log on to your system as the regular user you created during the installation process.

2. Click on the Applications menu, move your mouse over Accessories, and on the submenu that appears, click on Terminal.

3. In the terminal window, enter the following commands (remember that the "$" and the "#" are prompts and you should not type them at the beginning of the command):

```
$su -
Password: p@ss5678
#mkdir /demo
```

   You have just created a directory called "demo" which is a subdirectory under the root directory (/).

   Note: You can create multiple directories at the same time simply by separating their names with a space.

4. Display the contents of your working directory with the following command:

```
#ls
```

Notice that /demo is not displayed. The reason is that /demo is a subdirectory of the room directory. You are presently in a different directory.

5. Print your working directory to stdout (your screen) with the following command:

   ```
   #pwd
   ```

   Notice that you're in the super user root's home directory which is not where you created /demo.

6. Display the contents of the root directory with the following command:

   ```
   #ls /
   ```

   Notice that you now see the demo directory, along with several other directories that are all child directories under the parent /.

7. Enter the following command to change your working directory to /demo:

   ```
   #cd /demo
   ```

   #pwd You should now see that /demo is your working directory.

8. Enter the following commands:

   #mkdir demo1 demo2 demo3 You have just created three sub-directories in /demo called demo1, demo2, and demo3.

9. Now, list the contents of /demo with the following command:

   ```
   #ls
   ```

   You should now see the three subdirectories you just created.

To remove a directory, use the command "rmdir":

10. While still in /demo, remove the three directories you just created with the following command:

    ```
    #rmdir demo1 demo2 demo3
    ```

11. Use the "ls" command again to confirm that the three directories are removed:

    ```
    #ls
    ```

    The /demo directory should be empty.

12. You can also use wildcards to simplify file and directory management. Touch the up arrow on your keyboard several times. Notice that it repeats the last several commands. Stop when you see the command "mkdir demo1 demo2 demo3".

With "**`mkdir demo1 demo2 demo3`**" visible, press the Enter key to recreate the three directories.

13. Use the "ls" command to verify that the three directories have been re-created.
```
#ls
```

14. Now, use the "*" wildcard to simplify the rmdir process:
```
#rmdir demo*
```

15. Use the "ls" command to verify that the three directories have been deleted.
```
#ls
```

Student Exercise 2.2.2: Working with Files

Moving, Copying, and Deleting Files

The "mv", "cp", and "rm" commands are commonly used commands for basic file management.

`#mv [filename] [destination and filename]` moves a file to a new location. This is also used when you want to rename a file.

`#mv [current filename] [new filename]` renames a file.

`#cp [filename] [destination]` copies a file to a new location.

`#rm [filename]` deletes a file.

You can use the -f option to force a move, copy, or deletion without being asked for confirmation (Be careful when doing this!). You can use the -r option to move, copy, or delete recursively through directories. (Be *especially* careful when using the -r option with the -f option.)

The rm command is absolute and, once invoked, cannot be undone. Best practice is to always use the -i (interactive) option with "rm" which prompts you to confirm you really do want to delete the file.

The touch command is used to change file timestamps, but it is also a handy way to create empty files.

1. Using the "touch" command, create three new, empty files:
```
#touch file1 file2 file3
```

2. Now, issue an "ls" command to see the contents of the current directory.

3. The "mv" command (move) is used when you want to move or rename a file.

While still in /demo, issue the following command to rename file1:

```
#mv file1 file4
```

4. Use the “ls” command to view the contents of the directory. The former file1 should now appear as file4.

5. The “cp” command (copy) copies files from one location to another. While still in /demo, issue the following command to copy file4 from /demo to /:

   ```
   #cp file4 ../.
   ```

   (The ../. tells the system to copy file4 to the parent directory (..) and use the same name on the copy as the original (/.).

6. Use the “ls” command to view the contents of /demo and notice that file4 is still in /demo. Then, use the “ls /” command to view the contents of the root directory and you should see the copy of file4.

7. The “rm” command (remove) deletes files. Use the following command to remove file4 from the root directory:

   ```
   #rm /file4
   ```

   (Notice that you are prompted to confirm the deletion.)

8. Use the “ls” command with a wildcard to check the root (/) directory for any files whose names start with fil:

   ```
   #ls /fil*
   ```

   You should see a message stating “No such file or directory”.

9. Now, use wildcards and options to remove multiple files without being prompted. While in /demo, issue the following command to remove all files whose names start with fil:

   ```
   #rm -f fil*
   ```

10. Use the “ls” command with a wildcard to check /demo for any files whose names start with fil:

    ```
    #ls fil*
    ```

    As with the previous step, you should see a message stating “No such file or directory.

Other Helpful Commands

#su – [username] is the switch user command. The hyphen switches the present working directory to the new user’s home directory. When used with no parameters, the “su” command switches to “root”.

#**pwd** displays the present working directory's full path

#**ls** lists the directory contents

#**cat [filename]** concatenates files and prints on the standard output (usually the display screen)

#**less [filename]** (from the man page) **less** is a program similar to **more**, but which allows backward movement in the file as well as forward movement. Also, **less** does not have to read the entire input file before starting.

#**more [filename]** is a program that filters text to allow paging through a file one page at a time

whereis is a helpful command for finding configuration files and executable programs. It does not search through user directories.

Try this:

```
#whereis ifconfig
```

find is another helpful command that will search based on various criteria including file name, file size, modification date, and permissions. The find command can only be issued by a user who has permission to view the target files and directories.

Try this:

```
#find [filename within the current directory]
```

There are many options available for use with find:

`#find / -type d -name conf` will find all the directories named "conf"

`#find / -user donc` will find all files owned by "donc"

`#find / -name donc` will find all files with the same name as "donc"

`#find -name 'index.html'` would search for any file named index.html in the current directory and any subdirectory.

`#find / -name 'index.html'` would search for any file named index.html in the root directory and all subdirectories from root

`#find -name 'sshd*'` would search for any file beginning with the text string "sshd" in the current directory and any subdirectory.

`#find -name '*' -size +500k` would search for any file larger then 500k.

locate is also a command that is useful for finding files on a Linux system. It uses a database when searching for files, so it's faster than find.

You can use locate like this:

```
#locate [filename]
```

Files that have been created recently, however, may not be in the database. You can force an update of the database like this:

```
#updatedb
```
or
```
#locate -u
```

du is a way of estimating disk usage. When used with no arguments, du reports the disk space for the current directory. By default, disk space is printed in units of one kilobyte (1024 bytes). For example, to find out which directories are largest, use this command:

```
#du -S | sort -n
```
(The "S" switch tells it to report the size of each directory separately, not including subdirectories. The pipe (|) redirects the output of "du" to the "sort" utility. The "-n" switch sorts numerically.)

dmesg is a program that helps users print out bootup messages:

```
#dmesg | less
```

This command will pipe to "less"

An alternative is to pipe the dmesg output to a file. Try this:

```
#dmesg > boot.messages
```

You will find the boot.messages file in the present working directory. Try using cat, more, and less to view the contents of the file.

The "who" command displays currently logged on users:

`#who` displays currently logged on users, their terminal, and their login times.

`#who -u` adds idle time.

`#whoami` displays the name of the user initiating the command.

Viewing the contents of a file

Command	Syntax	What it does
cat	cat [filename]	"cat" is for "concatenate", cat displays the contents of file(s) named in the command
file	file [filename]	"file" identifies the type of file as directory, text, or binary.
head	head [filename]	"head" shows the top ten lines of the named file. You can change the number of lines shown by using the -n option (where "n" is the number of lines you wish to display).
tail	tail [filename]	"tail" shows the bottom ten lines of the named file. As with "head", you can change the number of lines shown by using the -n option (where "n" is the number of lines you wish to display).
more	more [filename]	"more" shows the contents of a file, one page at a time. You can see additional pages by pressing the space bar or view additional lines, one at a time, by pressing the enter key.
less	less [filename]	"less" is similar to "more" in that it shows the contents of a file, one page at a time, but "less" allows you to move forward and backward through the file using the arrow keys.
wc	wc [filename]	When used with no options, "wc" displays the number of lines, words, and characters in the named file. Options are available which allow you to specify bytes, characters, lines, and words.

Editing configuration files

In addition to managing a Linux system by executing various commands in the CLI or using tools in graphical interface, you will also need to frequently modify various configuration files.

There are several text editors which are commonly used to edit the Linux configuration files. In this document, we will be using "vim", a programmers' text editor. "vim" is an enhanced version of "vi". Most people use "vim", but refer to it as "vi". This exercise will help you become more comfortable with "vi" (pronounced "VEE-eye"), a traditional text editor found on most Linux and UNIX systems. Although many people consider "vi" to be somewhat awkward, its wide availability makes a fundamental understanding of its basic commands well worthwhile.

The traditional “vi” text editor has been replaced on most systems with “vim” (“vi” improved). The command set is substantially the same for both “vi” and “vim”. On most systems, the “vi” command has been aliased to “vim”.

To open a file with Vim, type the following command:

```
#vim [filename]
```

Operation within Vim is done with a variety of commands, some of which are listed here:

> **`:set nu`** displays line numbers along the left margin
>
> **`:q!`** quits without saving
>
> **`:wq`** writes and quits (saves and quits)
>
> **Arrow keys** can be used to move the cursor or letter keys can be used:
>
> > **h** to go left
> >
> > **j** to go down
> >
> > **k** to go up
> >
> > **l** to go right
>
> **G** goes to the end of the file
>
> **nG** (where “n” is a line number) goes to the specified line in the file

Vim has many more commands and options available. Help is available by typing **:help**.

Other Commonly Used Text Editors

Emacs

Emacs is a class of text editors, known for their extensibility. Emacs has more than 1000 editing commands. It also supports the use of macros to automate work by combining commands. The name is based on Editor MACrosS.

Development of emacs began in the mid-70s and continues actively as of this writing (early 2010).

Gedit

Gedit is the default text editor for the Gnome desktop environment. It supports syntax highlighting and is designed to be a very clean, easy-to-use editor. Gedit is available for both the Linux/Unix and the Windows platforms.

Notepad ++

Notepad ++ is a text editor for Windows. It is often used as a replacement for the built-in Notepad text editor. It offers several advantages over Notepad including tabbed windows, line numbering, and syntax highlighting.

Student Exercise 2.3: Working with vim: Using the vim Tutorial

1. In Terminal, enter the following command to start a VIM tutorial:
   ```
   $vimtutor
   ```
2. Work at least through lesson four.

There is a VIM cheat sheet on the following page. Feel free to copy it and tape it to the side of your monitor.

soundtraining.net
accelerated i.t. training

vim cheat sheet

Some common vim commands

Press the <ESC> (escape) key to ensure you're in normal mode, then:

`:q!` quits without saving

`:wq` saves and quits (write quit)

`x` deletes individual characters

`i` inserts text

`dw` deletes to the end of a word (d2w deletes two words, d3w deletes three words, etc.)

`d$` deletes to the end of a line

`dd` deletes an entire line (2dd deletes two lines, 23dd deletes 23 lines, etc.)

`u` undoes the last command

`U` fixes an entire line

`<CTRL>R` redoes the command

`p` puts the last deletion after the cursor

`r` replaces the character under the cursor

`cw` is the "change word" command, that deletes the word (from the cursor to the right) and places you in "insert" mode

`c$` is the "change line" command, that deletes the line (from the cursor to the right) and places you in "insert" mode

`<CTRL>g` shows your location in a file

`<SHIFT>G` moves to the end of the file, [number]<SHIFT>G moves to the line number specified in the command, for example 1<SHIFT>G moves to line #1.

`/[search term]` searches forward through a file for the search term. For example, "**`/apache`**" will search for the next instance of the word "apache" in the file

`?[search term]` searches backwards through a file for the search term. For example, "**`?apache`**" will search for the last instance before the cursor of the word "apache" in the file

`:s/[old]/[new]` will replace the next instance of "old" with "new". For example, **`:s/blue/red`** will replace the next instance of "blue" with "red".

`:s/[old]/[new]/g` will replace the every instance of "old" on the current line with "new". For example, :s/blue/red will replace the every instance of "blue" with "red".

`:#,#s/[old]/[new]/g` will replace every instance of "old" with "new" in the range of lines specified with the # sign.

`:!` allows you to execute external commands

`:set nu` turns on line numbering

`:nohlsearch` turns off highlighting of search terms

Using grep

“grep” is a filtering utility used in the ‘nix world to aid in searches. “grep” is one of the most useful tools in Information Systems and Technology.

Some examples:

- #**`grep red blue`** will display lines of text from the blue file that contain the word “red”
- #**`rpm –qa | grep smb`** will display all installed RPMs with “smb” in their name

Here is a handy way to use grep. Suppose you need to find a file (or files) containing a particular text string. Use the grep with the –r and –H options to find all files containing that particular string (remember that everything in Linux is case sensitive). By default, grep only prints the text string. If you’re looking for files containing the text string, you must tell grep to print the filename, too. The –H command does that.

In the following statement, -H prints the filename and –r searches recursively from the starting point (lists/) for the text string PHPMAILERHOST:

```
$ grep -Hr PHPMAILERHOST lists/
```

This is the output from the previous command:

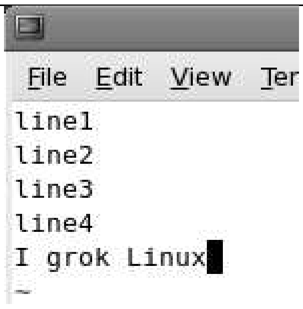
```
lists/config/config.php:define("PHPMAILERHOST",'');
```

You can include the option “n” in your search to display the line number in the file where the string appears.

Student Exercise 2.4: Conditional Searching

In this exercise, you will search for a unique text string within a file buried deep within a directory tree.

1. Create a deep directory tree with the following command in a terminal window:
   ```
   #mkdir -p /demo/demo1/demo2/demo3
   ```
   (The "p" switch creates parent directories when they do not already exist.)
2. Using "vi", create a file called "deepfile" in the demo4 directory:
   ```
   #vi /demo/demo1/demo2/demo3/deepfile
   ```
3. Enter five lines of text in the file as shown in the screen capture.
4. When you're finished, use the key combination of ESC, then :wq to save the file and close "vi".



```
line1
line2
line3
line4
I grok Linux
~
```

5. While still in a terminal window, enter the following command to find the text string "I grok Linux":
   ```
   #grep -Hr "I grok Linux" /demo
   ```
6. The command should return to stout (standard output) the following response:
   ```
   /demo/demo1/demo2/demo3/deepfile:5:I grok Linux
   ```
   (In the above output, the path is displayed, followed by the line number in the file where the text string appears, followed by the text string.)
   If your results differ, check spelling, remembering that text in a Linux terminal window is case-sensitive.

Using the alias command

The alias command is a shell function that allows you to substitute one command for another. Aliases are also handy for assigning default arguments to commands, such as ensuring that the "-i" (interactive) option is always used with the "cp" and "mv". The syntax for the alias command is:

> **`#alias [new command]="[command with arguments]"`**
>
> **`#alias cps="cp -s"`** would create the new alias "cps" which would always invoke the "cp" command with the symbolic link argument.

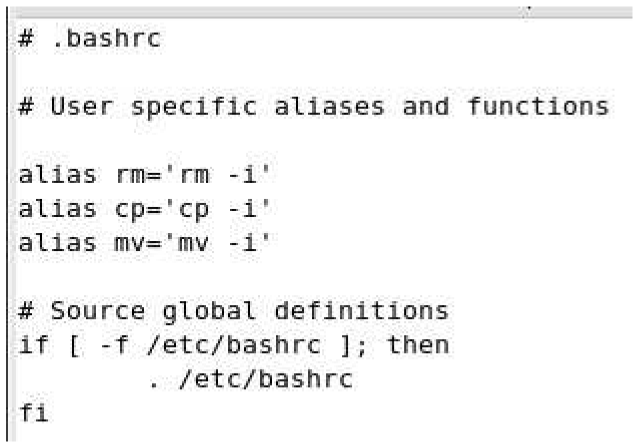
You can see existing aliases by issuing the alias command with no options at a command prompt.

Aliases can be removed with the unalias command:

#**unalias cps** will remove the “cps” alias.

Making Aliases Persistent

If you simply use the alias command to create an alias, the aliases are only in effect for your current session. To make them persistent across logons, add them to your profile by modifying ~/.bashrc. In the following screen capture, you can see how three aliases were added to the file, making them persistent across logons and system boots. Note the leading period in the filename (.bashrc) which makes it a hidden file in Linux.



```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

Figure 2 Creating persistent aliases

Starting Services

As mentioned earlier, many daemons (services) are started from shell scripts located in the /etc/rc.d/init.d directory. You can view the various services by navigating to the directory and issuing the “ls -l” command. You’ll notice that the files within the directory are all scripts (you can tell by the execute permission on each file). You can execute scripts from within the directory by preceding the script name with “./” or from any directory as follows:

Start a service:

```
#/etc/rc.d/init.d/./nfs start
```

Stop a service

```
#/etc/rc.d/init.d/./nfs stop
```

Restart a service

```
#/etc/rc.d/init.d/./nfs restart
```

Soundthinking point: Red Hat’s “service” tool

Linux distros based on Red Hat such as RHEL, CentOS, and Fedora also include a script called "service" which is located in /sbin. The "service" script will do essentially the same thing as the above commands, but in a simpler form:

```
#service nfs start
#service nfs stop
#service nfs status
#service nfs restart
```

When using a RedHat-based product such as RHEL, Fedora, CentOS, you can see the various services available for use with the service command by typing "service", then touching the tab key twice.

Note: When you compile applications from source, you will not be able to use tools such as those listed on this page without creating customized scripts for each compiled application. That is one of the benefits of using applications compiled by the distro vendor instead of compiling them yourself.

To start applications you compiled yourself, you must find the script or binary that starts the application. Usually, there is a README file included with the source code that will tell you the default installation paths. Alternatively, you can use the "find" utility to search the filesystem for files by name.

Linux Compression and Archiving Tools

Archiving is the process of storing multiple files in one file to simplify backup, moving, and transfer. Compression uses various algorithms to store files and directories in a way that consumes less space on the disk or tape. A common practice is to create an archive file and then compress it.

"tar" is a commonly-used archiving utility. It saves many files together in a single archive for tape or disk and allows the restoration of individual files from that archive.

Soundthinking point: The Acronym "tar"

The acronym "tar" comes from the phrase "tape archive".

Tar usage:

#**`tar`** [option] [file]

#**`tar -cf demofile.tar file1.txt file2.txt file3.txt`** will create the archive demofile.tar including the files file1.txt, file2.txt, and file3.txt.

#**`tar -tvf demofile.tar`** will list all files in the archive demofile.tar verbosely.

#**`tar -xf demofile.tar`** will extract all files from archive "demofile.tar".

Common compression tools in RedHat Linux include (.gz), bzip2 (.bz2), and zip (.zip). The compression tools used with each type of file are gzip, bzip2, and zip. The uncompression tools used with each type of compression are gunzip, bunzip2, and unzip.

Compression usage:

```
#gzip [filename]

#gunzip [filename]

#bzip2 [filename]

#bunzip2 [filename]

#zip [filename]

#unzip [filename]
```

bzip2 is recommended due to its high compression rate and availability on most UNIX/Linux systems. bzip2 can also create a single compressed file from multiple files.

zip and unzip are compatible with the Windows file compression utility PKZIP versions 2.04 and later.

Student Exercise 2.3: Archiving and compressing

In the following exercise, you will learn how to create a "tarball" and compress it using common archiving and compression tools.

1. Enter the following commands to switch user to root and change directory to /demo:
   ```
   $su -
   Password:p@ss5678
   #cd /demo
   ```

2. Once again, use the touch command to create three files in /demo:
   ```
   #touch file1 file2 file3
   ```
3. Enter the "ls" command to confirm the presence of file1, file2, and file3.

4. Create a tarball (tape archive) of the three files called files.tar with the following command:
   ```
   #tar cvf files.tar file*
   ```

5. View the contents of the tarball with the following command:
   ```
   #tar tvf files.tar
   ```

6. View the size of the tarball with the following command:

```
#ls –l
```

7. What is the size of files.tar?

   ____

   It should be about 10,240 bytes.

8. Compress the tarball with the following command:
   ```
   #gzip files.tar
   ```

9. Touch the up arrow twice to cycle back to the “ls –l” command and press Enter.

10. Notice that files.tar has been renamed to files.tar.gz, indicating that it’s a gzip compressed file. What is the size of files.tar.gz?

    ____

    It should now be about 141 bytes.

11. Now, uncompress files.tar.gz with the following command:
    ```
    #gunzip files.tar.gz
    ```

12. Touch the up arrow twice to cycle back to the “ls –l” command and press Enter.

13. Notice that the .gz extension has been removed and files.tar is back to its original size.

14. Remove the tarball with the following command:
    ```
    #rm -rf files.tar
    ```

You can perform similar operations with bzip2 using the commands bzip2 and bunzip2. tar also allows you create a tarball and compress it in a single operation using the “z” switch for gzip or the “j” switch for bzip2:

1. While in /demo, execute the following command:
   ```
   #tar cvfj files.tar.bz2 file*
   ```

2. Use the “ll” command to display the contents of /demo.

3. Notice that files.tar.bz2 is now compressed. What is its size?

   ____

4. What was the size of files.tar.gz?

   ____

   In this case, the tarball compressed with bzip2 should be slightly smaller than the tarball compressed with gzip.

5. Use the following command to view the contents of files.tar.bz2:
   ```
   #tar jtvf files.tar.bz2
   ```

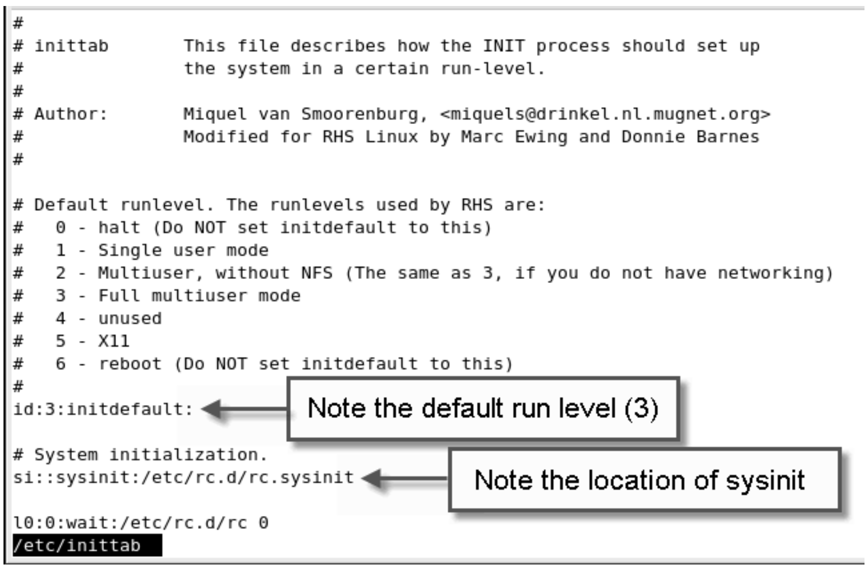
6. Notice that, even though compression has been applied to the tarball, you can still view the contents using the “t” and “j” option with tar.

7. You’re finished with the compression and archiving exercises, so you can delete the tarball:
   ```
   #rm –f files.tar.bz2
   ```

The Linux Boot Process

A Common Linux Boot Process

- The system is checked by the system BIOS which launches the first stage boot loader on the primary hard disk’s MBR
- The first stage boot loader is loaded into memory by itself and it then launches the second stage boot loader from the /boot partition
- The kernel is loaded into memory by the second stage boot loader. Necessary modules are loaded by the second stage boot loader which also mounts the root partition read-only
- Control of the boot process is transferred by the kernel to the /sbin/init program. “init” runs with a process ID of 1 and spawns all other processes.
- “init” reads /etc/inittab to find the location of the “sysinit” script.
- Services and user-space tools are loaded by the /sbin/init program which also mounts all partitions listed in /etc/fstab
- The login prompt appears



```
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
/etc/inittab
```

Figure 3 /etc/inittab from a Red Hat system

The init process

- The sysinit is the first script that runs on a Linux system
- Sysinit controls the most basic Linux services such as mounting the file systems specified in /etc/fstab, enabling the swap partition, setting system-wide environment variables, setting the system time, and other important operating system tasks.
- After sysinit is run, the system is started in the default-run-level specified in the "inittab" file in the /etc directory. Run level 3 is the most common run level.
- Any services defined in the default-run-level are started. The services are controlled by scripts within the rc[X].d directory (where “X” is the chosen run level).

Run Levels

The number of run levels varies from distro to distro, as do the default settings in each run level. What follows is an example of the default configurations for run levels in a system running a Red Hat-based distribution:

- runlevel0
  - Shut down the system
  - Do not set the inittab value to runlevel0
- runlevel1
  - Single-user mode
- runlevel2
  - Multi-user mode, but no NFS support
- runlevel3 (the most commonly used run level and usually the best choice for servers)
  - Multi-user mode without “X”
- runlevel4
  - Not used
- runlevel5 (good for end-user workstations, but not recommended for servers)
  - X11
- runlevel6
  - Reboot
  - Do not set the inittab value to runlevel6

A Debian-based system also has seven run-levels. Run levels 0, 1, and 6 are the same as in a Red Hat-based system. Run levels two through five are identical, but can be configured in whatever way you desire. The default configuration boots the system into run level 2 which is configured as full multi-user mode with graphics (X windows).

You can view the current run level with this command:

```
#runlevel
```

The display will indicate the current and previous run level(s), separated by a space.

You can change the current run level with this command:

`#init` [desired run level] or **`#telinit`** [desired run level]

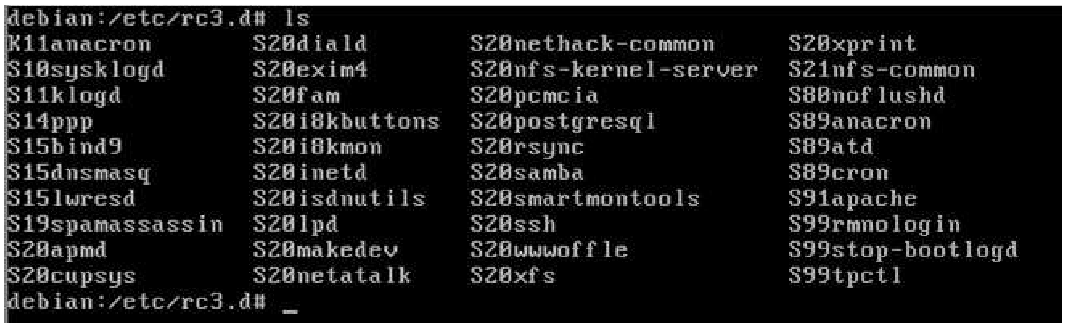
Controlling the boot process

Change the default run level by modifying /etc/inittab.

It is possible to more finely control the boot process by modifying scripts within “rc” directories.

- In Red Hat Linux, they are in the /etc/rc.d directory
- In SuSE Linux, they are in /etc/init.d/rc
- In Debian Linux, they are in /etc

There is an “rc” directory that corresponds to each run level. For example, rc3.d corresponds to run level 3. Look for the corresponding directory to the run level you wish to modify. Within that directory, you’ll find scripts for each of the services on the system. Each script name includes an “S” or a “K”. Scripts which start with “S” start indicated daemons with the directory’s run level. Scripts which start with “K” kill daemons within the directory’s run level. (Scripts in an rc directory are executed in alphabetical, then numerical order.)



```
debian:/etc/rc3.d# ls
K11anacron        S20diald         S20nethack-common     S20xprint
S10sysklogd       S20exim4         S20nfs-kernel-server  S21nfs-common
S11klogd          S20fam           S20pcmcia             S80noflushd
S14ppp            S20i8kbuttons    S20postgresql         S89anacron
S15bind9          S20i8kmon        S20rsync              S89atd
S15dnsmasq        S20inetd         S20samba              S89cron
S15lwresd         S20isdnutils     S20smartmontools      S91apache
S19spamassassin   S20lpd           S20ssh                S99rmnologin
S20apmd           S20makedev       S20wwwoffle           S99stop-bootlogd
S20cupsys         S20netatalk      S20xfs                S99tpctl
debian:/etc/rc3.d# _
```

Figure 4 Scripts within an rc directory

Notice in the screen capture that there a number of scripts starting with the letter “S”. These are the scripts that are started when this run level is chosen. Note that the “S” is followed by a number, which indicates the order in which the scripts run. If certain processes were to be killed (such as anacron), they would be listed with their name preceded by a “K”.

You’ll also notice scripts in /etc/rc.d called rc, rc.local, and rc.sysinit. The rc script is responsible for starting and stopping services when runlevels change, rc.sysinit runs once at boot time before all other rc scripts, and rc.local runs after all the other init scripts. You can put your own initialization stuff in rc.local instead of working through the System V runlevels.

Using chkconfig to Manage Startup Daemons (Services)

The chkconfig utility is a convenient way to manage and monitor startup daemons. It updates and queries runlevel information for system services. There are several switches available with chkconfig, however, commonly used switches include:

`chkconfig --list` which lists all of the services which chkconfig knows about, and whether they are stopped or started in each of the runlevels.

`chkconfig --level [levels] [service name] [on|off]` which starts or stops the specified service at the specified runlevel(s). For example, the following command would configure the system to automatically start the Web server (httpd) at runlevels 3 and 5 upon system boot:

```
#chkconfig –level 35 httpd on
```

Soundthinking point: Change Terminals

As a true multi-user operating system, Linux allows you to change terminals and even log on as a different user. When in the CLI, simply use the key combination of Alt-F[#] to change to a different terminal. Type *`tty`* to display which terminal is active.

System shutdowns and reboots

As with any operating system, it is very important to shut Linux down properly. During the shutdown process, services and daemons are stopped and file systems are unmounted in an orderly fashion. An improper shutdown can cause corrupted file systems and other problems that may actually prevent the system from booting successfully.

The use of the “sync” command is recommended to flush the filesystem buffers, thus synchronizing them with the hard disk.

Although it is possible to shut down the system by typing “init 0” or reboot by typing “init 6”, it is recommended to use the shutdown command to perform an orderly shutdown. There are several switches that can be used with the shutdown command:

`shutdown -r` reboots the system

`shutdown -h` halts the system

`shutdown -h 10` shuts down and halts the system in 10 minutes

`shutdown -r 23:00` forces a reboot at 11:00 p.m.

`shutdown -c` cancels a scheduled shutdown

`shutdown -c Ignore the last shutdown message` cancels a scheduled shutdown and broadcasts the message “Ignore the last shutdown message”.

`shutdown -k 10` doesn’t shut the system down, but broadcasts the message that the system will go into maintenance mode in 10 minutes.

Many distros also support the use of the “halt” command and the “reboot” command instead of requiring you to type the entire shutdown command with switches.

Soundthinking Point: Working in Terminal (No X)

When working in full multi-user mode, but with no X Windows, you can scroll up and down through the screen output by using the key combination of Shift-PageUp or Shift-PageDown.

X Windows

X Windows is the underlying technology used in the Linux/UNIX world to support graphics. There are several versions of X, but the one most commonly used with Linux is XFree86.

Think of X as the foundation for the Graphical User Interface (GUI). On top of the foundation (X) is the user environment. As with X, there are several different user environments available, but the two most common are KDE and GNOME. The user environments provide such things as icons, buttons, desktop backgrounds, and user applications.

Starting the GUI

When in a command-line terminal, use the command "**`startx`**" to start X windows.

To kill X, use the key combination of ctrl+alt+backspace.

Configuring "X"

In Red Hat, you can run the XFree86 configuration tool by issuing the command:

```
#system-config-display
```

Getting help

Linux includes ample, built-in help including "man" pages, "info", "help", and "apropos".

man

man formats and displays the online manual pages. There are manual pages for nearly every command imaginable. Unfortunately, many of the man pages assume a fairly extensive background in UNIX, therefore they often require research beyond the initial man page.

The man pages are divided into sections. Many man entries appear in only one section, but some appear in multiple sections such as when a command and a library function have the same name. The sections that are most likely to be of interest to system and network administrators are sections 1, 5, and 8.

- Section 1: user commands (introduction)
- Section 2: system calls (introduction)
- Section 3: library functions (introduction)
- Section 4: special files (introduction)
- Section 5: file formats (introduction)

- Section 6: games (introduction)
- Section 7: conventions and miscellany (introduction)
- Section 8: administration and privileged commands (introduction)
- Section L: math library functions
- Section N: tcl fuctions (Tool command language, a dynamic programming language)

You can view a man page as follows:
```
#man chown
```

You can specify a particular section as follows:
```
#man 1 chmod
```

The above command would display only section 1 (the user commands section) of the manual for the chmod command. chmod is also a system call, so if you wanted to see the man page for the system call "chmod", you would need to enter the following command:
```
#man 2 chmod
```

info

info is an on-line manual reader used by the GNU Project to document utilities. It is similar to man (and often produces identical documents), but offers a standardized set of commands for viewing the documentation. The info utility does not assume as great a depth of UNIX knowledge as man.

Basic usage is the same as man:
```
#info chown
```

The above command will display the info page for the chown command.

Info divides its help into nodes instead of sections. A node, like a section in man, describes a specific topic at a specific level of detail. In a moment, you will work through the first few steps of a tutorial on using info.

help

The --help option is included with most GNU utilities. It displays command options and other information about the utility:
```
#ls --help
```

The above command shows options and other information about the ls command.

apropos

apropos looks in the description sections of man pages for text strings. When executed, apropos will return every man page with whose description contains the specified text string:
```
#apropos edit
```

The above command displays a list of every man page with a description which contains the text string “edit”.

apropos is helpful when you know what you want to do, but you are not certain of the appropriate utility or command to accomplish it.

Student Exercise 2.4: Getting Help

This exercise will familiarize you with the various commands available for getting help including man, info, apropos, and --help.

Student Exercise 2.4.1: Working with man

1. In a terminal window, enter the following command:
```
$man ls
```
2. Press Enter. What happens?

   ____

   The screen output should display the next line in the man page.
3. Now press the space bar. What happens?

   ____

   The screen output should display the next page in the man page.
4. Use the arrow keys to move up and down through the page. When you’re finished, touch “q” to quit. (You can also use the pageup and pagedown keys to move through the page.)
5. Enter the following command:
```
$man 1 chmod
```
6. What do you see in the upper left-hand corner of the screen?

   ____

   You should see CHMOD(1) indicating that you are in the man section 1, the user commands section.
7. Touch “q” to quit.
8. Enter the following command:

```
$man 2 chmod
```

9. Now, what do you see in the upper left-hand corner of the screen?

   ____

   You should see CHMOD(2) indicating that you are in the man section 2, the system calls section.

10. Touch "q" to quit the man page.

Student Exercise 2.4.2: Working with apropos

Now, you will use the apropos utility to search for man pages pertaining to a particular topic.

1. Start by entering the following command to see the man page for sudo:
   $man sudo
2. Read the description and notice that it includes the word "sudo". Touch "q" to quit the man page.
3. Now, enter the following command:
   ```
   $apropos sudo
   ```
4. Notice in the output that sudo is listed, along with every other command whose description includes the word "sudo".
5. Also, notice that sudo is listed, not only by itself, but as part of other commands. In addition to the obvious difference that there are different commands, notice that each command is followed by a number. The number indicates which section of man pages contains that particular documentation.

Student Exercise 2.4.3: Working with info

Next, you'll use the info utility to view help for commands and learn how to navigate info pages by working through the first part of an "info" tutorial.

1. Enter the following command to see the info page for chmod:
   ```
   $info chmod
   ```
2. The info page for chmod opens.

3. Touch the “h” key to start a brief tutorial for info.
4. Work through screen 1.4 of the tutorial.
5. Touch the “q” key when you’re finished.

Student Exercise 2.4.4: Working with –help

This exercise will show you how to use --help with GNU utilities:

1. Enter the following command:
   ```
   $chmod –help
   ```
2. Notice that the help screen, albeit abbreviated, shows you the proper syntax for using the chmod command.
3. Enter the following command:
   ```
   $ls –help
   ```
4. Notice that the help screen fills more than one screen. Use the key combination of Shift-PageUp and Shift-PageDown to move up and down through the Terminal window.

Chapter Four:
File and Directory Management

"Linux: the choice of a GNU generation"

--ksh@cis.ufl.edu

Linux File System

Debian Etch, RedHat, SuSE, and other distributions use the ext3 file system by default. ext3 is a journaling file system which offers greater stability and reliability than predecessor file systems. Another popular journaling file system is ReiserFS. V3 of reiserfs is used as the default filesystem for Lindows, FTOSX, Libranet, Xandros and Yoper. Ext4 is also now available and becoming a default file system in new Linux versions.

Linux can also read and/or write to many other file systems including FAT, FAT32, NTFS, HPFS, and others. Partitions are mounted onto existing directories called "mount-points".

Linux uses a tree model to organize directories and files. Directories are the basic unit of storage in the Linux file system. Directories can contain files or other directories. In the same way that a tree cannot exist without its roots, the Linux file system starts at root. Root is designated by "/". (The term "root" is used in three different ways in Linux: "Root" is the name of the superuser, to identify the superuser's home directory [/root], and to indicate the root of the file system [/]. It can be difficult to know which "root" someone is talking about. It helps to be clear about what is meant when referring to "root".)

Linux File Types

When you issue the "**`ls -l`**" command, Linux will display a listing of files along with information about the files. The far left hand column of the listing indicates the type of file. Three common file types are regular files, links, and directories.

Regular files

Regular files are the most common file type on Linux or UNIX systems. They can be used to store various types of data including text that you can read or binary data that can be executed by the system. It is often helpful to identify more information about the file than whether it is a regular file or not. For example, you might want to know whether the file is an ASCII text file or a shell script. You can use the "file" command to identify the file type.

```
    $file [filename]
```

Links

Links are files that point to other files on the system. There are two types of links: hard links and symbolic links.

Hard links are a special type of directory entry that have certain limitations:

- Hard links can only point to a file; they cannot point to a directory.
- They cannot be distinguished from the file to which they are pointing.

Hard links are created with the "ln" command:

```
    $ln [source] [target]
```

Symbolic links are special files that store a pathname to another file.

Symbolic links are created with the “ln” command, combined with the “-s” option:

```
$ln -s [source pathname] [target]
```

You can think of symbolic links as being similar to shortcuts in Microsoft Windows.

Directories

Directories are containers that hold various types of files or other directories. Directories are used for organizing the file system.

Mounting a Device

In order to make a device such as a CD-ROM or floppy drive available to the file system, it must be “mounted” to an existing mount point within the file system. Before using the “mount” command, ensure that the desired mount point already exists within the file system. A common place to locate mount points is within the /mnt directory (but they can be placed anywhere). To mount a device to the mount-point:

```
#mount /dev/cdrom /mnt/cdrom
```

You can navigate to the newly mounted device with the “cd” command:

```
#cd /mnt/cdrom
```

Before ejecting CDs or floppy disks, you must unmount them from the file system. To unmount a mount-point:

```
#umount /mnt/cdrom
```

Note that, before a mountpoint can be unmounted, you must “CD” out of the directory which you wish to unmount.

Partitions can be mounted automatically on boot through the fstab file, which is located at /etc/fstab.

Understanding Mount Points

You can think of mount points as a way of accessing a partition. Recall that in Linux, everything is oriented around the file system. The first SCSI drive on a computer might be known as /dev/sda, the second as /dev/sdb, and so on. The first IDE drive would be known as /dev/hda. Partitions are numbered, so the first partition on the first SCSI drive would be /dev/sda1, the second partition would be /dev/sda2, etc.

You cannot, however, access partitions through /dev files; you must create mount points which are simply a means of gaining access to a partition through the computer’s file system.

A basic partitioning scheme will usually have three partitions: /, /boot, and a swap partition. Often, server administrators will create separate partitions for other purposes as shown below:

Mount Point	Purpose
/boot	Contains boot loader, kernel and related files
/	Root of the file system
/usr	UNIX system resources (usr) is where you find program and related files
/home	Users' home directories and profiles
/var	Variable size files including logs and print spools. Also home to WWW and FTP files.
/tmp	Temporary files

Managing File and Directory Permissions

Linux uses three types of file/directory permissions. For files:

- Read means that you can view a file's contents.
- Write means that you change or delete the file.
- Execute means that you can run the file as a program.

For directories:

- Read means you can list the contents of the directory.
- Write means you can add and remove files in the directory.
- Execute means you can list information about the files in the directory.

Permissions are assigned to both users and groups

- Read permission: Whether the file can be read or the directory contents can be listed
- Write permission: Whether the file can be modified or written to or whether changes can be made to the contents of a directory. For example, without write permission, you cannot create, delete, nor rename a file

- Execute permission: For files, whether the file can be executed. For directories, this is the permission to enter, search through the directory, or execute a program from the directory

You can list file or directory permissions by using the "ls" command with the "-l" option, for example: **ls –l.** On Red Hat-based systems, you can also use the alias "**ll"**. When you list files and folders using the "–l" option, you'll see a display like this:

```
#d-rw-rw---    1 jbach jbach  150  March 10  08:08 file1.txt
```

The first column (drw-rw----) is actually ten columns which can be divided into four groups:

- The first group is a single column used to identify the type of entry. The options are:
    - "d" which indicates a directory
    - "l" is a symbolic link to another program or file elsewhere on the system
    - "-" is a regular file
- The second group is three columns used to identify the permissions of the owner
- The third group is three columns used to identify the permissions of the owner group
- The fourth group of three columns identifies the permissions of the world (everyone).

The three permissions columns are, in order: read (r), write (w), and execute (x). If the permissions are expressed as "-rw-rw----", then the entry is a file ("-") whose owner and owner group has read+write permissions, but not execute and the rest of the world is denied access.

Changing permissions

Use the "chmod" command to change permissions. You can set permissions for the user (u), group (g), and others (o). Permissions can also be set for all (a).

Permissions are set using +, -, and =.

+ adds the permission, - removes the permission, and = sets the permission as specified and can be used to copy permissions.

For example:

- #`chmod u+x file1` adds the execute permission for the user owner on file1.
- #`chmod g-w file2` removes the write permission for the group owner on file1.
- #`chmod a+r file3` adds the write permission for everyone on file3.
- #`chmod o=u file4` copies the user permissions for file4 to the world.

Octal (Numeric) permissions

Octal permissions are simply a form of shorthand for assigning access to files and folders.

- Read = 4

- Write = 2
- Execute = 1
- No access = 0

Use **chmod** to assign permissions using the numeric system. For example:

#**chmod 644 file1** would assign owner read+write (6=2+4), the owner's group and everyone would have read permission (4).

Special Permissions

- Sticky bit:
  - Can be used on "world writable directories" to prevent users from deleting other users' files

Assigning Special Permissions

- #**chmod 1766 [directory]** (1 makes it sticky)

Student Exercise 4.1: Managing file and directory permissions

In this exercise, you will use various commands to set and edit file and directory permissions.

Student Exercise 4.1.1: Viewing Permissions

1. Change to the superuser (root) account with the switch user command:

```
$su -
Password:p@ss5678
```

2. Navigate to the "/" directory and use the "ls –l" command to verify the existence of /demo. If it is not present, use the "mkdir" command to create a new directory called "demo":

```
#mkdir demo
```

3. Now, use the "cd" command to navigate to the "demo" directory, then use the "ls -l" command to verify the existence of file1, file2, and file3. If they are not present, use the "touch" command to create three files:

```
#cd demo

#touch file1 file2 file3
```

4. Use the following command to view the permissions for the three files you just created:

   ```
   #ls -l
   ```

5. What are the permissions on each of the files for the user?

   ____

   Each of the files should have "rw" permission for the user.

6. What are the permissions on each of the files for the group?

   ____

   Each of the files should have "r" permission for the group.

7. What are the permissions on each of the files for the world?

   ____

   Each of the files should have "r" permission for the world (other).

Student Exercise 4.1.2: Changing Permissions Using Alphabetic Expressions

In this exercise, you will use alphabetic and octal syntax to modify file permissions using the chmod command.

1. While still in /demo, execute the following command to display the permissions for the files:

   ```
   #ls -l file1
   ```

2. Notice that only information about file1 is displayed because you modified the "ls –l" command by appending "file1" to the end of the command. What are the permissions for the user on file1?

   ____

   Again, it should be "rw" for the user on file1.

3. Now, use the following command:

   ```
   #chmod u+x file1
   ```

4. Now, execute the following command to display the permissions for the file:

   ```
   #ls –l file1
   ```

5. What are the permissions now for the user on file1?

   ____

   The permissions should now be “rwx” for the user.

6. Execute the following command:

   ```
   #chmod g+w file2
   ```

7. Now, what are the permissions for the group on file2?

   ____

   The permissions should be “rw-“ for file2.

8. Execute the following command:

   ```
   #chmod a+x file*
   ```

9. What happened to the permissions on all files in the directory?

   ____

   All files should now have “x” permission in addition to any pre-existing permissions.

10. Execute the following command:

    ```
    #chmod o=u file3
    ```

11. Use the ls –l command to view the new permissions. What happened to the permissions for the world on file3? Are they the same as for the user?

    ____

    The permission for the world (other) should now match the permissions for the user.

Student Exercise 4.1.3: Octal (Numeric) Permissions

In this exercise, you will practice managing permissions using octal settings instead of alphabetic expressions.

1. Execute the following command:
   ```
   #chmod 644 file*
   ```
2. Using the **`ls -l`** command, display the changed permissions. What are the new permissions for the files?

   ____

   The new permissions should be "rw" for the user and "r" for the group and the world (other).
3. Execute the following command:
   ```
   #chmod 777 file1
   ```
4. Again, use the **`ls -l`** command to display the permissions. What happened?

   ____

   The permissions for file1 are now "rwx" for user, group, and world.

Setting Default Permissions

The **`umask`** command is the user file-creation mask command which allows you set default permissions.

The **umask** command uses an octal value that is the inverse of the values used with the **chmod** command. In other words, if you wish to set permissions for a directory to full for the owner, read for the group, and nothing for the world, you would use **chmod** as follows:

```
#chmod 740
```

To set the default permissions for *all future files and directories* created to full for the owner, read for the group, and nothing for the world, use the **umask** command with a value that is the inverse of the value used with **chmod:**

```
#umask 037
```

Note that this is a universal command and cannot be applied to a single directory.

Disk Configuration Tools

#**`fdisk /dev/hda`** starts the disk configuration utility “fdisk” (“hda” represents the first IDE drive, “sda” would represent the first SCSI drive on the system.)

Using fdisk returns a different prompt than the customary Linux command prompt:

- Command (m for help):**p** displays your disk partitions.
- Command (m for help):**d** schedules partitions for deletion (if you make a mistake and don't want to delete a partition, you can simply type **q** to quit without saving)
- Command (m for help):**l** lists known partition types
- Command (m for help):**m** lists available commands

Related commands

- **`fdisk -l`** displays information about partitions on a hard drive
- **`fdisk -t`** sets the file system for a partition
- **`mkfs`** will format a partition
- **`fsck`** will repair a corrupted file system
- **`fsck /mbr`** will repair a corrupted Master Boot Record

soundthinking point: partition management tool

The open source tool gparted is a great tool for managing disks and partitions.