Emirene Joy Castillo BSIT-3A

## 3– Exploring The System

Now that we know how to move around the file system, it's time for a guided tour of our Linux system. Before we start however, we’re going to learn some more commands that will be useful along the way:

* + ls – List directory contents
  + file – Determine file type
  + less – View file contents

### More Fun With ls

The ls command is probably the most used command, and for good reason. With it, we can see directory contents and determine a variety of important file and directory at- tributes. As we have seen, we can simply enter ls to see a list of files and subdirectories contained in the current working directory:

[me@linuxbox ~]$ **ls**

Desktop Documents Music Pictures Public Templates Videos

Besides the current working directory, we can specify the directory to list, like so:

me@linuxbox ~]$ **ls /usr**

bin games kerberos libexec sbin src etc include lib local share tmp

Or even specify multiple directories. In this example we will list both the user's home di- rectory (symbolized by the “~” character) and the /usr directory:

[me@linuxbox ~]$ **ls ~ /usr**

/home/me:

Desktop Documents Music Pictures Public Templates Videos

/usr:

bin games kerberos libexec sbin src etc include lib local share tmp

We can also change the format of the output to reveal more detail:

[me@linuxbox ~]$ **ls -l**

total 56

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Desktop

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Documents

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Music

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Pictures

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Public

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Templates

drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Videos

By adding “-l” to the command, we changed the output to the long format.

#### Options And Arguments

This brings us to a very important point about how most commands work. Commands are often followed by one or more *options* that modify their behavior, and further, by one or more *arguments*, the items upon which the command acts. So most commands look kind of like this:

**command -options arguments**

Most commands use options consisting of a single character preceded by a dash, for ex- ample, “-l”, but many commands, including those from the GNU Project, also support *long options*, consisting of a word preceded by two dashes. Also, many commands allow multiple short options to be strung together. In this example, the ls command is given two options, the “l” option to produce long format output, and the “t” option to sort the result by the file's modification time.

[me@linuxbox ~]$ **ls -lt**

More Fun With ls

We'll add the long option “--reverse” to reverse the order of the sort:

[me@linuxbox ~]$ **ls -lt --reverse**

Note that command options, like filenames in Linux, are case-sensitive.

The ls command has a large number of possible options. The most common are listed in Table 3-1.

*Table 3- 1: Common ls Options*

|  |  |  |
| --- | --- | --- |
| **Option** | **Long Option** | **Description** |
| -a | --all | List all files, even those with names that begin |
|  |  | with a period, which are normally not listed |
|  |  | (i.e., hidden). |
| -A | --almost-all | Like the -a option above except it does not |
|  |  | list . (current directory) and .. (parent |
|  |  | directory). |
| -d | --directory | Ordinarily, if a directory is specified, ls will |
|  |  | list the contents of the directory, not the |
|  |  | directory itself. Use this option in conjunction |
|  |  | with the -l option to see details about the |
|  |  | directory rather than its contents. |
| -F | --classify | This option will append an indicator character |
|  |  | to the end of each listed name. For example, a |
|  |  | “/” if the name is a directory. |
| -h | --human-readable | In long format listings, display file sizes in |
|  |  | human readable format rather than in bytes. |
| -l |  | Display results in long format. |
| -r | --reverse | Display the results in reverse order. Normally, |
|  |  | ls displays its results in ascending |
|  |  | alphabetical order. |
| -S |  | Sort results by file size. |
| -t |  | Sort by modification time. |

#### A Longer Look At Long Format

As we saw before, the “-l” option causes ls to display its results in long format. This for- mat contains a great deal of useful information. Here is the Examples directory from an Ubuntu system:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| -rw-r--r-- | 1 | root | root | 3576296 | 2007-04-03 | 11:05 | Experience ubuntu.ogg |
| -rw-r--r-- | 1 | root | root | 1186219 | 2007-04-03 | 11:05 | kubuntu-leaflet.png |
| -rw-r--r-- | 1 | root | root | 47584 | 2007-04-03 | 11:05 | logo-Edubuntu.png |
| -rw-r--r-- | 1 | root | root | 44355 | 2007-04-03 | 11:05 | logo-Kubuntu.png |
| -rw-r--r-- | 1 | root | root | 34391 | 2007-04-03 | 11:05 | logo-Ubuntu.png |
| -rw-r--r-- | 1 | root | root | 32059 | 2007-04-03 | 11:05 | oo-cd-cover.odf |
| -rw-r--r-- | 1 | root | root | 159744 | 2007-04-03 | 11:05 | oo-derivatives.doc |
| -rw-r--r-- | 1 | root | root | 27837 | 2007-04-03 | 11:05 | oo-maxwell.odt |
| -rw-r--r-- | 1 | root | root | 98816 | 2007-04-03 | 11:05 | oo-trig.xls |
| -rw-r--r-- | 1 | root | root | 453764 | 2007-04-03 | 11:05 | oo-welcome.odt |
| -rw-r--r-- | 1 | root | root | 358374 | 2007-04-03 | 11:05 | ubuntu Sax.ogg |

Let's look at the different fields from one of the files and examine their meanings:

*Table 3-2: ls Long Listing Fields*

**Field Meaning**

-rw-r--r-- Access rights to the file. The first character indicates the

type of file. Among the different types, a leading dash means a regular file, while a “d” indicates a directory. The next three characters are the access rights for the file's owner, the next three are for members of the file's group, and the final three are for everyone else. The full meaning of this is discussed in Chapter 9 – Permissions.

1 File's number of hard links. See the discussion of links later in this chapter.

root The username of the file's owner.

root The name of the group which owns the file.

32059 Size of the file in bytes.

2007-04-03 11:05 Date and time of the file's last modification.

oo-cd-cover.odf Name of the file.

Determining A File's Type With file

### Determining A File's Type With file

As we explore the system it will be useful to know what files contain. To do this we will use the file command to determine a file's type. As we discussed earlier, filenames in Linux are not required to reflect a file's contents. While a filename like “picture.jpg” would normally be expected to contain a JPEG compressed image, it is not required to in Linux. We can invoke the file command this way:

**file *filename***

When invoked, the file command will print a brief description of the file's contents. For example:

[me@linuxbox ~]$ **file picture.jpg**

picture.jpg: JPEG image data, JFIF standard 1.01

There are many kinds of files. In fact, one of the common ideas in Unix-like operating systems such as Linux is that “everything is a file.” As we proceed with our lessons, we will see just how true that statement is.

While many of the files on your system are familiar, for example MP3 and JPEG, there are many kinds that are a little less obvious and a few that are quite strange.

### Viewing File Contents With less

The less command is a program to view text files. Throughout our Linux system, there are many files that contain human-readable text. The less program provides a conve- nient way to examine them.

**What Is “Text”?**

There are many ways to represent information on a computer. All methods in- volve defining a relationship between the information and some numbers that will be used to represent it. Computers, after all, only understand numbers and all data is converted to numeric representation.

Some of these representation systems are very complex (such as compressed video files), while others are rather simple. One of the earliest and simplest is called *ASCII text*. ASCII (pronounced "As-Key") is short for American Standard

Code for Information Interchange. This is a simple encoding scheme that was first used on Teletype machines to map keyboard characters to numbers.

Text is a simple one-to-one mapping of characters to numbers. It is very compact. Fifty characters of text translates to fifty bytes of data. It is important to under- stand that text only contains a simple mapping of characters to numbers. It is not the same as a word processor document such as one created by Microsoft Word or OpenOffice.org Writer. Those files, in contrast to simple ASCII text, contain many non-text elements that are used to describe its structure and formatting. Plain ASCII text files contain only the characters themselves and a few rudimen- tary control codes like tabs, carriage returns and line feeds.

Throughout a Linux system, many files are stored in text format and there are many Linux tools that work with text files. Even Windows recognizes the impor- tance of this format. The well-known NOTEPAD.EXE program is an editor for plain ASCII text files.

Why would we want to examine text files? Because many of the files that contain system

settings (called *configuration files*) are stored in this format, and being able to read them gives us insight about how the system works. In addition, many of the actual programs that the system uses (called *scripts*) are stored in this format. In later chapters, we will learn how to edit text files in order to modify systems settings and write our own scripts, but for now we will just look at their contents.

The less command is used like this:

**less *filename***

Once started, the less program allows you to scroll forward and backward through a text file. For example, to examine the file that defines all the system's user accounts, enter the following command:

[me@linuxbox ~]$ **less /etc/passwd**

Once the less program starts, we can view the contents of the file. If the file is longer than one page, we can scroll up and down. To exit less, press the “q” key.

The table below lists the most common keyboard commands used by less.

Viewing File Contents With less

*Table 3-3: less Commands*

**Command Action**

Page Up or b Scroll back one page Page Down or space Scroll forward one page Up Arrow Scroll up one line

Down Arrow Scroll down one line

G Move to the end of the text file

1G or g Move to the beginning of the text file

/*characters* Search forward to the next occurrence of *characters*

n Search for the next occurrence of the previous search

h Display help screen

q Quit less

**Less Is More**

The less program was designed as an improved replacement of an earlier Unix program called more. The name “less” is a play on the phrase “less is more” — a motto of modernist architects and designers.

less falls into the class of programs called “pagers,” programs that allow the easy viewing of long text documents in a page by page manner. Whereas the more program could only page forward, the less program allows paging both forward and backward and has many other features as well.

### A Guided Tour

The file system layout on your Linux system is much like that found on other Unix-like systems. The design is actually specified in a published standard called the *Linux Filesys- tem Hierarchy Standard*. Not all Linux distributions conform to the standard exactly but most come pretty close.

Next, we are going to wander around the file system ourselves to see what makes our Linux system tick. This will give you a chance to practice your navigation skills. One of the things we will discover is that many of the interesting files are in plain human-read - able text. As we go about our tour, try the following:

1. cd into a given directory
2. List the directory contents with ls -l
3. If you see an interesting file, determine its contents with file
4. If it looks like it might be text, try viewing it with less

**Remember the copy and paste trick!** If you are using a mouse, you can double click on a filename to copy it and middle click to paste it into commands.

As we wander around, don't be afraid to look at stuff. Regular users are largely prohibited from messing things up. That's the system administrators job! If a command complains about something, just move on to something else. Spend some time looking around. The system is ours to explore. Remember, in Linux, there are no secrets!

Table 3-4 lists just a few of the directories we can explore. Feel free to try more!

*Table 3-4: Directories Found On Linux Systems*

**Directory Comments**

/ The root directory. Where everything begins.

/bin Contains binaries (programs) that must be present for the system to boot and run.

/boot Contains the Linux kernel, initial RAM disk image (for drivers needed at boot time), and the boot loader.

Interesting files:

* /boot/grub/grub.conf or menu.lst, which are used to configure the boot loader.
* /boot/vmlinuz, the Linux kernel

/dev This is a special directory which contains *device nodes*. “Everything is a file” also applies to devices. Here is where the kernel maintains a list of all the devices it understands.

**Directory Comments**

/etc The /etc directory contains all of the system-wide configuration files. It also contains a collection of shell scripts which start each of the system services at boot time. Everything in this directory should be readable text.

Interesting files: While everything in /etc is interesting, here are some of my all-time favorites:

* + /etc/crontab, a file that defines when automated jobs will run.
  + /etc/fstab, a table of storage devices and their associated mount points.
  + /etc/passwd, a list of the user accounts.

/home In normal configurations, each user is given a directory in

/home. Ordinary users can only write files in their home directories. This limitation protects the system from errant user activity.

/lib Contains shared library files used by the core system programs. These are similar to DLLs in Windows.

/lost+found Each formatted partition or device using a Linux file system, such as ext3, will have this directory. It is used in the case of a partial recovery from a file system corruption event.

Unless something really bad has happened to your system, this directory will remain empty.

/media On modern Linux systems the /media directory will contain the mount points for removable media such as USB drives, CD-ROMs, etc. that are mounted automatically at insertion.

/mnt On older Linux systems, the /mnt directory contains mount points for removable devices that have been mounted manually.

/opt The /opt directory is used to install “optional” software. This is mainly used to hold commercial software products that may be installed on your system.

**Directory Comments**

/proc The /proc directory is special. It's not a real file system in the sense of files stored on your hard drive. Rather, it is a virtual file system maintained by the Linux kernel. The “files” it contains are peepholes into the kernel itself. The files are readable and will give you a picture of how the kernel sees your computer.

/root This is the home directory for the root account.

/sbin This directory contains “system” binaries. These are programs that perform vital system tasks that are generally reserved for the superuser.

/tmp The /tmp directory is intended for storage of temporary, transient files created by various programs. Some configurations cause this directory to be emptied each time the system is rebooted.

/usr The /usr directory tree is likely the largest one on a Linux system. It contains all the programs and support files used by regular users.

/usr/bin /usr/bin contains the executable programs installed by

your Linux distribution. It is not uncommon for this directory to hold thousands of programs.

/usr/lib The shared libraries for the programs in /usr/bin.

/usr/local The /usr/local tree is where programs that are not

included with your distribution but are intended for system- wide use are installed. Programs compiled from source code are normally installed in /usr/local/bin. On a newly installed Linux system, this tree exists, but it will be empty until the system administrator puts something in it.

/usr/sbin Contains more system administration programs.

/usr/share /usr/share contains all the shared data used by

programs in /usr/bin. This includes things like default configuration files, icons, screen backgrounds, sound files, etc.

/usr/share/doc Most packages installed on the system will include some

kind of documentation. In /usr/share/doc, we will find documentation files organized by package.

**Directory Comments**

/var With the exception of /tmp and /home, the directories we have looked at so far remain relatively static, that is, their contents don't change. The /var directory tree is where data that is likely to change is stored. Various databases, spool files, user mail, etc. are located here.

/var/log /var/log contains *log files*, records of various system

activity. These are very important and should be monitored from time to time. The most useful one is

/var/log/messages. Note that for security reasons on some systems, you must be the superuser to view log files .

### Symbolic Links

As we look around, we are likely to see a directory listing with an entry like this:

lrwxrwxrwx 1 root root 11 2007-08-11 07:34 libc.so.6 -> libc-2.6.so

Notice how the first letter of the listing is “l” and the entry seems to have two filenames? This is a special kind of a file called a *symbolic link* (also known as a *soft link* or *sym- link*.) In most Unix-like systems it is possible to have a file referenced by multiple names. While the value of this may not be obvious, it is really a useful feature.

Picture this scenario: A program requires the use of a shared resource of some kind con- tained in a file named “foo,” but “foo” has frequent version changes. It would be good to include the version number in the filename so the administrator or other interested party could see what version of “foo” is installed. This presents a problem. If we change the name of the shared resource, we have to track down every program that might use it and change it to look for a new resource name every time a new version of the resource is in- stalled. That doesn't sound like fun at all.

Here is where symbolic links save the day. Let's say we install version 2.6 of “foo,” which has the filename “foo-2.6” and then create a symbolic link simply called “foo” that points to “foo-2.6.” This means that when a program opens the file “foo”, it is actually opening the file “foo-2.6”. Now everybody is happy. The programs that rely on “foo” can find it and we can still see what actual version is installed. When it is time to upgrade to “foo-2.7,” we just add the file to our system, delete the symbolic link “foo” and create a new one that points to the new version. Not only does this solve the problem of the ver- sion upgrade, but it also allows us to keep both versions on our machine. Imagine that “foo-2.7” has a bug (damn those developers!) and we need to revert to the old version. Again, we just delete the symbolic link pointing to the new version and create a new

symbolic link pointing to the old version.

The directory listing above (from the /lib directory of a Fedora system) shows a sym- bolic link called “libc.so.6” that points to a shared library file called “libc-2.6.so.” This means that programs looking for “libc.so.6” will actually get the file “libc-2.6.so.” We will learn how to create symbolic links in the next chapter.

### Hard Links

While we are on the subject of links, we need to mention that there is a second type of link called a *hard link*. Hard links also allow files to have multiple names, but they do it in a different way. We’ll talk more about the differences between symbolic and hard links in the next chapter.

### Summing Up

With our tour behind us, we have learned a lot about our system. We've seen various files and directories and their contents. One thing you should take away from this is how open the system is. In Linux there are many important files that are plain human-readable text. Unlike many proprietary systems, Linux makes everything available for examination and study.

### Further Reading

* The full version of the *Linux Filesystem Hierarchy Standard* can be found here: <http://www.pathname.com/fhs/>
* An article about the directory structure of Unix and Unix-like systems: <http://en.wikipedia.org/wiki/Unix_directory_structure>
* A detailed description of the ASCII text format:

Outputs:

