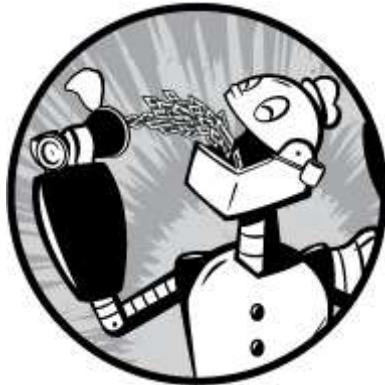


# 3

## 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 attributes. As we have seen, we can simply enter `ls` to get 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 include lib local sbin share src
```

---

We can even specify multiple directories. In the following example, we list both the user's home directory (symbolized by the ~ character) and the `/usr` directory:

---

```
[me@linuxbox ~]$ ls ~ /usr  
/home/me:  
Desktop Documents Music Pictures Public Templates Videos  
/usr:  
bin games include lib local sbin share src
```

---

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 2017-10-26 17:20 Desktop  
drwxrwxr-x 2 me me 4096 2017-10-26 17:20 Documents  
drwxrwxr-x 2 me me 4096 2017-10-26 17:20 Music  
drwxrwxr-x 2 me me 4096 2017-10-26 17:20 Pictures  
drwxrwxr-x 2 me me 4096 2017-10-26 17:20 Public  
drwxrwxr-x 2 me me 4096 2017-10-26 17:20 Templates  
drwxrwxr-x 2 me me 4096 2017-10-26 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, which consist of a single character preceded by a dash, for example, `-l`. Many commands, however, 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 the following example, the `ls` command is given two options, which are 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
```

---

We'll add the long option `--reverse` to reverse the order of the sort.

---

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

---

**NOTE**

*Command options, like filenames in Linux, are case sensitive.*

---

The `ls` command has a large number of possible options, the most common of which are listed in [Table 3-1](#).

**Table 3-1:** Common `ls` Options

Option	Long Option	Description
<code>-a</code>	<code>--all</code>	List all files, even those with names that begin with a period, which are normally not listed (that is, hidden).
<code>-A</code>	<code>--almost-all</code>	Like the <code>-a</code> option except it does not list <code>.</code> (current directory) and <code>..</code> (parent directory).

Option	Long option	Description
-d	--direc-tory	Ordinarily, if a directory is specified, <code>ls</code> will list the contents of the directory, not the directory itself. Use this option in conjunction with the <code>-l</code> option to see details about the directory rather than its contents.
-F	--clas-sify	This option will append an indicator character to the end of each listed name. For example, it will append a forward slash (/) if the name is a directory.
-h	--hu-man-read-able	In long format listings, display file sizes in human-readable format rather than in bytes.
-l		Display results in long format.
-r	--re-verse	Display the results in reverse order. Normally, <code>ls</code> 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 earlier, the `-l` option causes `ls` to display its results in long format. This format contains a great deal of useful information. Here is the `Examples` directory from an Ubuntu system:

---

```
-rw-r--r-- 1 root root 3576296 2017-04-03 11:05 Experience ubuntu.ogg
-rw-r--r-- 1 root root 1186219 2017-04-03 11:05 kubuntu-leaflet.png
-rw-r--r-- 1 root root 47584 2017-04-03 11:05 logo-Edubuntu.png
-rw-r--r-- 1 root root 44355 2017-04-03 11:05 logo-Kubuntu.png
-rw-r--r-- 1 root root 34391 2017-04-03 11:05 logo-Ubuntu.png
-rw-r--r-- 1 root root 32059 2017-04-03 11:05 oo-cd-cover.odf
-rw-r--r-- 1 root root 159744 2017-04-03 11:05 oo-derivatives.doc
-rw-r--r-- 1 root root 27837 2017-04-03 11:05 oo-maxwell.odt
-rw-r--r-- 1 root root 98816 2017-04-03 11:05 oo-trig.xls
-rw-r--r-- 1 root root 453764 2017-04-03 11:05 oo-welcome.odt
-rw-r--r-- 1 root root 358374 2017-04-03 11:05 ubuntu Sax.ogg
```

---

**Table 3-2** provides us with a look at the different fields from one of the files and their meanings.

**Table 3-2:** `ls` Long Listing Fields

Field	Meaning
<code>-rw-r--r-</code>	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 <code>d</code> 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. <a href="#">Chapter 9</a> discusses the full meaning of this in more detail.
<code>1</code>	File's number of hard links. See "Symbolic Links" on <a href="#">page 21</a> and "Hard Links" on <a href="#">page 22</a> .
<code>root</code>	The username of the file's owner.
<code>root</code>	The name of the group that owns the file.
<code>32059</code>	Size of the file in bytes.

Field	Meaning
2017-04-03 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`

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 our system are familiar, for example, MP3 and JPEG files, 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 convenient way to examine them.

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, some 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 system 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`

---

### WHAT IS “TEXT”?

There are many ways to represent information on a computer. All methods involve defining a relationship between the information and some numbers that will be used to represent it. Computers, after all, understand only 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 understand 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 LibreOffice 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 rudimentary control codes such as 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 importance of this format. The well-known NOTEPAD.EXE program is an editor for plain ASCII text files.

---

Once started, the `less` program allows us 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 `q`.

**Table 3-3** lists the most common keyboard commands used by `less`.

**Table 3-3:** `less` Commands

Command	Action
PAGE UP or <code>b</code>	Scroll back one page
PAGE DOWN or space	Scroll forward one page
Up arrow	Scroll up one line
Down arrow	Scroll down one line

Command	Action
<code>G</code>	Move to the end of the text file
<code>1G</code> or <code>g</code>	Move to the beginning of the text file
<code>/characters</code>	Search forward to the next occurrence of <code>characters</code>
<code>n</code>	Search for the next occurrence of the previous search
<code>h</code>	Display help screen
<code>q</code>	Quit <code>less</code>

---

## 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 *pgers*, 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.

---

## Taking a Guided Tour

The file system layout on a Linux system is much like that found on other Unix-like systems. The design is actually specified in a published standard

called the *Linux Filesystem Hierarchy Standard*. Not all Linux distributions conform to the standard exactly, but most come pretty close.

---

### REMEMBER THE COPY-AND-PASTE TRICK!

If you are using a mouse, you can double-click a filename to copy it and middle-click to paste it into commands.

---

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

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

If we accidentally attempt to view a non-text file and it scrambles the terminal window, we can recover by entering the `reset` command.

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 administrator's 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. There may be some slight differences depending on our Linux distribution. Don't be afraid to look around and 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 include <i>/boot/grub/grub.conf</i> , or <i>menu.lst</i> , which is used to configure the boot loader, and <i>/boot/vmlinuz</i> (or something similar), the Linux kernel.
/dev	This is a special directory that contains <i>device nodes</i> . “Everything is a file” also applies to devices. Here is where the kernel maintains a list of all the devices it understands.
/etc	The /etc directory contains all the system-wide configuration files. It also contains a collection of shell scripts that start each of the system services at boot time. Everything in this directory should be readable text. While everything in /etc is interesting, here are some all-time favorites: <i>/etc/crontab</i> , a file that defines when automated jobs will run; <i>/etc/fstab</i> , a table of storage devices and their associated mount points; and <i>/etc/passwd</i> , a list of the user accounts.
/home	In normal configurations, each user is given a directory in /home. Ordinary users can write files only in their home directories. This limitation protects the system from errant user activity.

Directory	Comments
<i>/lib</i>	Contains shared library files used by the core system programs. These are similar to dynamic link libraries (DLLs) in Windows.
<i>/lost+found</i>	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.
<i>/media</i>	On modern Linux systems, the <i>/media</i> directory will contain the mount points for removable media such as USB drives, CD-ROMs, and so on, that are mounted automatically at insertion.
<i>/mnt</i>	On older Linux systems, the <i>/mnt</i> directory contains mount points for removable devices that have been mounted manually.
<i>/opt</i>	The <i>/opt</i> directory is used to install “optional” software. This is mainly used to hold commercial software products that might be installed on the system.
<i>/proc</i>	The <i>/proc</i> 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.
<i>/root</i>	This is the home directory for the root account.

Directory	Comments
<i>/sbin</i>	This directory contains “system” binaries. These are programs that perform vital system tasks that are generally reserved for the superuser.
<i>/tmp</i>	The <i>/tmp</i> directory is intended for the storage of temporary, transient files created by various programs. Some configurations cause this directory to be emptied each time the system is rebooted.
<i>/usr</i>	The <i>/usr</i> directory tree is likely the largest one on a Linux system. It contains all the programs and support files used by regular users.
<i>/usr/bin</i>	<i>/usr/bin</i> contains the executable programs installed by your Linux distribution. It is not uncommon for this directory to hold thousands of programs.
<i>/usr/lib</i>	The shared libraries for the programs in <i>/usr/bin</i> .
<i>/usr/local</i>	The <i>/usr/local</i> 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 <i>/usr/local/bin</i> . On a newly installed Linux system, this tree exists, but it will be empty until the system administrator puts something in it.
<i>/usr/sbin</i>	Contains more system administration programs.
<i>/usr/share</i>	<i>/usr/share</i> contains all the shared data used by programs in <i>/usr/bin</i> . This includes things such as default configuration files, icons, screen backgrounds, sound files, and so on.

Directory	Comments
/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.
/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, and so forth, are located here.
/var/log	/var/log contains <i>log files</i> , records of various system activity. These are important and should be monitored from time to time. The most useful ones are /var/log/messages and /var/log/syslog. 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 (for example, /lib) with an entry like this:

---

```
lrwxrwxrwx 1 root root 11 2018-08-11 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 *symlink*). In most Unix-like systems, it is possible to have a file referenced by multiple names. While the value of this might not be obvious, it is really a useful feature.

Picture this scenario: a program requires the use of a shared resource of some kind contained in a file named “foo,” but “foo” has frequent ver-

sion 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 installed. That doesn’t sound like fun at all.

Here is where symbolic links save the day. Suppose 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 version upgrade, 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 at the beginning of this section (from the `/lib` directory of a Fedora system) shows a symbolic 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 *hard links*. 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.

[Support](#)    [Sign Out](#)

©2022 O'REILLY MEDIA, INC. [TERMS OF SERVICE](#)    [PRIVACY POLICY](#)