7

The Linux Utilities

In This Chapter

Objectives

After reading this chapter you should be able to:

- Use basic utilities to list files and display text files
- Copy, move, and remove files
- Display the beginning or end of a file
- Search, sort, print, categorize, and compare text files
- Compress, decompress, and archive files
- Count the number of letters, words, and lines in a file
- Locate utilities on the system
- Change the modification time of a file
- Display information about users and the system
- Record a session in a file
- Edit text files

When Linus Torvalds introduced Linux and for a long time thereafter, Linux did not have a GUI (graphical user interface): It ran on character-based terminals only, using a CLI (command-line interface), also referred to as a textual interface. All the tools ran from a command line. Today the Linux GUI is important, but many people—especially system administrators—run many command-line utilities. Command-line utilities are often faster, more powerful, or more complete than their GUI counterparts. Sometimes there is no GUI counterpart to a textual utility; some people just prefer the hands-on feeling of the command line. This chapter describes a number of frequently used utilities and concludes with tutorials on the vim and nano text editors. See "Working from the Command Line" on page 125 for an introduction to the command line.

When you work with a CLI, you are working with a shell (Chapters 5, 9, and 28). When you are working on the command line it is important to quote special characters as explained on page 150.

More utilities are covered throughout the book

tip This chapter introduces a few important utilities that are good to know as you start using Linux. More utilities are covered throughout the book. See the inside of the front and back covers for a complete list.

Basic Utilities

One of the advantages of Linux is that it comes with thousands of utilities that perform myriad functions. You will use utilities whenever you work with Linux, whether you use them directly by name from the command line or indirectly from a graphical menu or icon. The following sections discuss some of the most basic and important utilities that are available from a CLI. Some of the more important utilities are also available from a GUI; others are available only from a GUI.

Run these utilities from a command line

tip This chapter describes command-line, or textual, utilities. You can experiment with these utilities from a terminal, a terminal emulator within a GUI (page 126), or a virtual console (page 127).

LPI cat: Joins and Displays Files

The cat utility joins and copies files to standard output. The name of the cat utility is derived from *catenate,* which means to join together, one after the other. You can also use cat to display virtual system files in the **/proc** directory hierarchy as explained on page 490.

Arguments

The arguments are the pathnames of one or more files that cat processes. You can use cat to display the contents of one or more text files on the screen. More precisely, cat copies the files to standard output, which by default is attached to the screen.

```
$ cat practice
This is a small file that I created
using:  a text editor!

End.
```

If you do not specify an argument or if you specify a hyphen (–) in place of a filename, cat reads from standard input. The following example shows cat working without an argument; the **<** symbol causes the shell to redirect standard input to cat to come from **practice**. The shell passes no arguments to cat.

```
$ cat < practice
This is a small file that I created
using:  a text editor!

End.
```

You can use cat to create short, simple files. See "The Keyboard and Screen as Standard Input and Standard Output" on page 160 and "Redirection" on page 161 for more examples of redirecting the standard input and standard output of cat.

Options

Number The **–n** (**––number**) option numbers all lines as they are written to standard output while the **–b** (**––number-nonblank**) numbers only non-blank lines.

```
$ cat -n practice
     1  This is a small file that I created
     2  using:  a text editor!
     3
     4  End.
```

Tabs The **–T** (**––show-tabs**) option displays TABs as **^I**.

```
$ cat -T practice
This is a small file that I created
using:^Ia text editor!

End.
```

You can combine options like this

```
$ cat -bT practice
```

Or, using long options, like this

```
$ cat --number-nonblank --show-tabs practice
```

Nonprinting The **–v** (**––show-nonprinting**) option displays CONTROL characters using the caret notation (e.g., **^M** means CONTROL-M) and displays characters that have the high bit set (META characters) using the **M-** notation (e.g., **M-M** means META-M). This option does not convert TABs and LINEFEEDs. Use **–T** (**––show-tabs**) if you want to display TABs as **^I**. LINEFEEDs cannot be displayed as anything but themselves; otherwise, the line could be too long.

Related Utilities

tac The tac (cat spelled backwards) utility works the same way as cat works except it reverses the order of the lines in each file.

rev The rev (reverse) utility works the same way as cat works except it reverses the order of the characters in each line.

LPI date: Displays the System Time and Date

Without any arguments, the date utility displays the date and time known to the local system. If you set up a locale database (page 374), date uses that database to substitute in its output terms appropriate to the locale for your account. Use timedatectl (page 579) to set the system clock.

```
$ date
Mon Sep 29 18:02:26 PDT 2014
```

The hardware clock and the system clock

tip The hardware clock is the time kept in the BIOS (page 32). This clock is battery powered and keeps time even when the computer is off. The system clock runs only while the system is running and is what Linux uses when it needs to know the time. On some systems the system clock is adjusted periodically by NTP (Network Time Protocol; page 321) so it stays accurate.

Arguments

The following example formats and specifies the contents of the output of date. See the date man page for a complete list of format sequences.

```
$ date +"%A %B %d"
Monday September 29
```

Options

Date The **–d** ***datestring*** (**––date=*datestring***) option displays the date specified by ***datestring***, not the date known to the system. According to the date man page, "the ***datestring*** is a mostly free-format date string" such as **2pm next thursday**. See **DATE STRING** in the date man page for details about the syntax of ***datestring***. This option does not change the system clock.

UTC The **–u** (**––utc** *or* **––universal**) option displays or sets the time and date using UTC (Universal Coordinated Time; page 1280). UTC is also called GMT (Greenwich Mean Time).

```
$ date -u
Tue Sep 30 00:50:18 UTC 2014
```

Related Utilities

timedatectl The timedatectl utility displays more information about the system clock. You can also use it to set the system clock. See page 579.

cal The cal utility displays a calendar for the month and year you specify. Without any arguments it displays a calendar for the current month.

LE+ echo: Displays Arguments

The echo utility copies its arguments, followed by a NEWLINE, to standard output. The Bourne Again Shell has an echo builtin that works similarly to the echo utility.

The echo builtin/utility is a good tool for learning about the shell and other Linux utilities. Some examples on page 174 use echo to illustrate how special characters, such as the asterisk, work. Throughout Chapters 5, 9, and 28, echo helps explain how shell variables work and how you can send messages from shell scripts to the screen. You can even use echo to create a short file by redirecting its output:

```
$ echo "This is a short file." > short
$ cat short
This is a short file.
```

Arguments

The arguments can include quoted strings, ambiguous file references, and shell variables. The shell recognizes and expands unquoted special characters in arguments.

The following example shows echo copying its arguments to standard output. The second command includes an unquoted asterisk (*; page 174) that the shell expands into a list of files in the working directory before passing that list as arguments to echo; echo copies this list to standard output.

```
$ echo This is a sentence.
This is a sentence.
$ echo star: *
star: memo memo.0714 practice
$ ls
memo  memo.0714  practice
```

Option

NEWLINE The **–n** option suppresses the NEWLINE that normally terminates the output of echo. This option is useful in shell scripts when you want to prompt a user and have the response appear on the same line as the prompt. See page 1063 for an example of a shell script that uses this feature.

LPI hostname: Displays the System Name

The hostname utility displays the name of the system you are working on. Use this utility if you are not sure that you are logged in on the correct machine. See also **/etc/hostname** on page 485.

```
$ hostname
guava
```

Option

The following option works with hostname.

FQDN The **–f** (**––fqdn**) option displays the *FQDN* (page 1248) of the local system.

Related Utility

hostnamectl The hostnamectl utility displays more information about the local system. You can also use it to change the hostname.

LE less Is more: Display a Text File One Screen at a Time

You can use the less or more utilities, called *pagers,* to view a long text file one screen at a time. Each of these utilities pauses after displaying a screen of text. You can then press the SPACE bar to display the next screen of text.

Although less and more are very similar, they have subtle differences. The less utility, for example, allows you to move backward while viewing a file in some situations where more does not. Whereas more must read an entire file before displaying anything, less does not have to and so starts more quickly than more when displaying a large file. At the end of the file less displays an **END** message and waits for you to press **q** before returning you to the shell. In contrast, more returns you directly to the shell. While using both utilities you can press **h** to display a Help screen that lists commands you can use while paging through a file. Give the command **less /etc/services** to experiment with paging through a file.

Arguments

Both less and more take the names of files you want to view as arguments. If you do not specify an argument or if you specify a hyphen (**–**; less only) in place of a filename, less and more read from standard input.

Options

The following options work with less.

Clear screen The **–c** (**––clear-screen**) option paints each new screen from the top down; by default less scrolls the display.

EOF The **–e** (**––quit-at-eof**) option causes less to exit when it reaches the second EOF (when you press SPACE while less is displaying **END** at the end of a file) so you do not need to type **q** to quit.

Truncate The **–S** (**––chop-long-lines**) option truncates lines wider than the screen. By default less wraps long lines.

No initialization The **–X** (**––no-init**) option prevents less from sending terminal initialization and deinitialization strings to the terminal. By default, less clears the screen when it finishes; this option causes the last page of what you were viewing to remain on the screen.

optional You can set the **LESS** environment variable (page 1054) in your **~/.bash_profile** file (page 336) to set options for less each time you call it and when it is called from another program such as man. For example, the following line in the **.bash_profile** file in your home directory will cause less to always run with the **–X** option.

```
export LESS='-X'
```

Related Utility

most The most utility (part of the **most** package; see page 512 for installation instructions) is newer and more capable than less and more. See the most man page for details.

LPI ls: Displays Information About Files

The ls utility displays information about one or more files. It lists the information alphabetically by filename unless you use an option to change the order.

When you do not provide an argument, ls displays the names of the visible files (those with filenames that do not begin with a period; page 188) in the working directory.

Arguments

The arguments are one or more pathnames of any ordinary, directory, or device files. The shell expands ambiguous file references (page 173) in the arguments.

When you specify an ordinary file as an argument, ls displays information about that one file. When the argument is a directory, ls displays the contents of the directory. It displays the name of the directory only when needed to avoid ambiguity, such as when the listing includes more than one directory.

```
$ ls memos.zach/130715.1
memos.zach/130715.1

$ ls memos.zach
130712.1  130714.2  130714.3  130715.1

$ ls memos.*
memos.max:
130619.1  130621.2  130622.1

memos.sam:
130811.2  130811.3  130812.1

memos.zach:
130712.1  130714.2  130714.3  130715.1
```

LE Options

Options determine the type of information ls displays, and the manner and order in which it displays the information. When you do not use an option, ls displays a short list that contains just the names of files, in alphabetical order. See page 153 for examples of how options affect ls output.

All The **–a** (**––all**) option includes hidden filenames (those filenames that begin with a period; page 188) in the listing. Without this option ls does not list information about files with hidden filenames unless you specify the name of a hidden file as an argument. The * ambiguous file reference does not match a leading period in a filename, so you must use this option or explicitly specify a filename (ambiguous or not) that begins with a period to display information about files with hidden filenames.

LE Directory The **–d** (**––directory**) option displays directories without displaying their contents.

```
$ ls -ld /
dr-xr-xr-x. 27 root root 4096 07-16 18:37 /
```

This option is useful when you want to find out, for example, the name of the user who owns a directory in a large directory such as **/etc**. Instead of scrolling through the output of **ls –l /etc** looking for a directory, you can give the command **ls –ld /etc/*filename*** (e.g., **ls –ld /etc/cron.d**).

Human readable The **–h** (**––human-readable**) option, when specified with the **–l** option, displays sizes in K (kilobyte), M (megabyte), and G (gigabyte) blocks, as appropriate. This option works with the **–l** and **–s** options only. It displays powers of 1,024. Use **––si** to display powers of 1,000.

```
$ ls -lh /bin
...
-rwxr-xr-x 1 root root  15K 06-03 13:54 tailf
-rwxr-xr-x 1 root root 346K 2014-02-04  tar
-rwxr-xr-x 1 root root  11K 2014-08-27  tempfile
-rwxr-xr-x 1 root root  59K 2014-03-24  touch
-rwxr-xr-x 1 root root  27K 2014-03-24  true
...
```

Long The **–l** (lowercase "el"; **––format=long**) option lists more information about each file. See page 199 for an explanation of this output. If standard output for a directory listing is sent to the screen, this option displays the number of blocks used by all files in the listing on a line before the listing.

LE Recursive The **–R** (**––recursive**) option recursively lists directory hierarchies.

Reverse The **–r** (**––reverse**) option displays the list of filenames in reverse sorted order.

LE Size The **–s** (**––size**) option displays the number of 1,024-byte blocks allocated to the file. The size precedes the filename. With the **–l** option, this option displays the size in column 1 and shifts other items one column to the right. If standard output for a directory listing is sent to the screen, this option displays the number of blocks used by all files in the listing on a line before the listing. You can include the **–h** option to make the file sizes easier to read. The following example recursively lists the **memos** directory hierarchy in reverse size order.

```
$ ls -lRrs memos
memos:
4 drwxrwxr-x. 2 sam sam 4096 04-29 14:11 memos.zach
4 drwxrwxr-x. 2 sam sam 4096 04-29 14:32 memos.sam
4 drwxrwxr-x. 2 sam sam 4096 04-30 14:15 memos.max

memos/memos.zach:
4 -rw-rw-r--. 1 sam sam 4064 04-29 14:11 130715.1
4 -rw-rw-r--. 1 sam sam 3933 04-29 14:11 130714.3
4 -rw-rw-r--. 1 sam sam 3317 04-29 14:11 130714.2
...
```

LE+ rm: Removes a File (Deletes a Link)

The rm utility removes hard and/or symbolic links to one or more files. Frequently this action results in the file being deleted. See page 211 for information on links.

Arguments

The arguments are the pathnames of the files whose links rm will remove. Removing the only (last) hard link to a file deletes the file (page 216). Removing a symbolic link deletes the symbolic link only.

Be careful when you use rm with ambiguous file references

caution Because this utility enables you to remove a large number of files with a single command, use rm cautiously, especially when you are working with ambiguous file references. *Never use* rm *with ambiguous file references while you are working with* **root** *privileges.* If you have any doubts about the effect of an rm command with an ambiguous file reference, first use echo with the same file reference and evaluate the list of files the reference generates. Alternately, you can use the rm **–i** (**––interactive**) option.

Options

Force The **–f** (**––force**) option, without asking for your consent, removes files for which you do not have write access permission. This option suppresses informative messages if a file does not exist.

Interactive The **–i** (**––interactive**) option prompts you before removing each file. If you use **–r** (**––recursive**) with this option, rm also prompts you before examining each directory.

```
$ rm -ri memos
rm: descend into directory 'memos'? y
rm: descend into directory 'memos/memos.max'? y
rm: remove regular file 'memos/memos.max/130621.2'? y
rm: remove regular file 'memos/memos.max/130619.1'? n
...
```

You can create an alias (page 398) for **rm –i** and put it in a startup file (page 188) so rm always runs in interactive mode.

Recursive The **–r** (**––recursive**) option deletes the contents of the specified directory, including all its subdirectories, and the directory itself. Use this option with caution; do not use it with wildcards (* and ?).

Verbose The **–v** (**––verbose**) option displays the name of each file as it is removed.

Related Utilities

testdisk The testdisk utility (**testdisk** package) scans, repairs, and sometimes can recover disk partitions. In some cases it can undelete files. See the testdisk man page for details.

Working with Files

This section describes utilities that copy, move, print, search through, display, sort, compare, and identify files.

Filename completion

tip After you enter one or more letters of a filename (as an argument) on a command line, press TAB, and the shell will complete as much of the filename as it can. When only one filename starts with the characters you entered, the shell completes the filename and places a SPACE after it. You can keep typing or you can press RETURN to execute the command at this point. When the characters you entered do not uniquely identify a filename, the shell completes what it can and waits for more input. If pressing TAB does not change the display, press TAB again to display a list of possible completions. For more information refer to "Pathname Completion" on page 395.

LE+ cp: Copies Files

The cp utility copies one or more files. Use scp (page 723) or rsync (page 724) to copy files from one system to another (or to make local copies). See page 245 for a description of the related mv (move) utility.

Arguments

With two arguments that are pathnames, neither of which specifies a directory, cp copies the file named by the first argument to the file named by the second argument.

```
$ cp memo memo.cp
```

With two or more arguments that are pathnames, the last of which is an existing directory, cp copies the files named by all but the last argument to the directory named by the last argument.

```
$ cp memo1 memo2 memo4 memo.dir
```

cp can destroy a file

caution If the destination file exists *before* you give a cp command, cp overwrites it. Because cp overwrites (and destroys the contents of) an existing destination file without warning, you must take care not to cause cp to overwrite a file that you need. The cp **–i** (interactive) option prompts you before it overwrites a file.

The following example assumes the file named **orange.2** exists before you give the cp command. The user answers **y** to overwrite the file.

```
$ cp -i orange orange.2
cp: overwrite 'orange.2'? y
```

Options

Archive The **–a** (**––archive**) option attempts to preserve the owner, group, permissions, access date, and modification date of source file(s) while copying a directory recursively.

Backup The **–b** (**––backup**) option makes a backup copy of a file that would be removed or overwritten by cp. The backup copy has the same name as the destination file with a tilde (~) appended to it. When you use both **–b** and **–f**, cp makes a backup copy when you try to copy a file over itself. For more backup options, search for **Backup options** in the **coreutils** info page.

Force The **–f** (**––force**) option causes cp to try to remove the destination file (when it exists but cannot be opened for writing) before copying the source file. This option is useful when the user copying a file does not have write permission to an existing file but does have write permission to the directory containing the file. Use this option with **–b** to back up a destination file before removing or overwriting it.

Interactive The **–i** (**––interactive**) option prompts you whenever cp would overwrite a file. If you respond with a string that starts with **y** or **Y**, cp copies the file. If you enter anything else, cp does not copy the file.

```
$ cp -i * ../memos.helen
cp: overwrite '../memos.helen/130619.1'? y
cp: overwrite '../memos.helen/130622.1'? n
```

Preserve The **–p** (**––preserve**[=***attr***]) option creates a destination file with the same owner, group, permissions, access date, modification date, and ACLs as the source file. The **–p** option does not take an argument.

Without ***attr,*** **––preserve** works as described above. The ***attr*** is a comma-separated list that can include **mode** (permissions), **ownership** (owner and group), **timestamps** (access and modification dates), **links** (hard links), and **all** (all attributes).

Recursive The **–R** *or* **–r** (**––recursive**) option recursively copies directory hierarchies including ordinary files. The last argument must be the name of a directory.

Verbose The **–v** (**––verbose**) option displays the name of each file as cp copies it.

```
$ cp -rv memos memos.bak
'memos' -> 'memos.bak'
'memos/memos.max' -> 'memos.bak/memos.max'
'memos/memos.max/130619.1' -> 'memos.bak/memos.max/130619.1'
'memos/memos.max/130622.1' -> 'memos.bak/memos.max/130622.1'
'memos/memos.sam' -> 'memos.bak/memos.sam'
'memos/memos.sam/130811.3' -> 'memos.bak/memos.sam/130811.3'
'memos/memos.sam/130811.2' -> 'memos.bak/memos.sam/130811.2'
...
```

LE+ cut: SELECTS CHARACTERS OR FIELDS FROM INPUT LINES

The cut utility selects characters or fields from lines of input and writes them to standard output. Character and field numbering start with 1. Although limited in functionality, cut is easy to learn and use and is a good choice when columns and fields can be specified without using pattern matching.

ARGUMENTS

Arguments to cut are pathnames of ordinary files. If you do not specify an argument or if you specify a hyphen (–) in place of a pathname, cut reads from standard input.

Options

Characters The **–c** ***clist*** (**––characters=*clist***) option selects the characters given by the column numbers in ***clist***. The value of ***clist*** is one or more comma-separated column numbers or column ranges. A range is specified by two column numbers separated by a hyphen. A range of ***–n*** means columns **1** through ***n***; ***n–*** means columns ***n*** through the end of the line.

The following example displays the permissions of the files in the working directory. The **–c2-10** option selects characters 2 through 10 from each input line.

```
$ ls -l | cut -c2-10
otal 2944
rwxr-xr-x
rw-rw-r--
rw-rw-r--
rw-rw-r--
rw-rw-r--
```

Input delimiter The **–d** ***dchar*** (**––delimiter=*dchar***) option specifies ***dchar*** as the input field delimiter. This option also specifies ***dchar*** as the output field delimiter unless you use the **––output-delimiter** option. The default delimiter is a TAB character. Quote ***dchar*** as necessary to protect it from shell expansion.

Fields The **–f** ***flist*** (**––fields=*flist***) option selects the fields specified by ***flist***. The value of ***flist*** is one or more comma-separated field numbers or field ranges. A range is specified by two field numbers separated by a hyphen. A range of ***–n*** means fields **1** through ***n***; ***n–*** means fields ***n*** through the last field. The field delimiter is a TAB character unless you use the **–d** option to change it.

The following example displays a list of full names as stored in the fifth field (**–f5**) of the **/etc/passwd** file. The **–d** option specifies that the colon character is the field delimiter.

```
$ cut -d: -f5 /etc/passwd
Sam the Great
Sam the Great
Zach Brill
Max Wild
...
```

The next command outputs the size and name of each file in the working directory. The **–f** option selects the fifth and ninth fields from the input lines. The **–d** option tells cut to use SPACEs, not TABs, as delimiters. The tr utility (page 266) with the **–s** option changes sequences of two or more SPACE characters to a single SPACE; otherwise, cut counts the extra SPACE characters as separate fields.

```
$ ls -l | tr -s '  ' ' ' | cut -f5,9 -d' '

259 countout
9453 headers
1474828 memo
1474828 memos_save
```

```
7134 tmp1
4770 tmp2
13580 typescript
```

Output delimiter The **––output-delimiter=*ochar*** option specifies ***ochar*** as the output field delimiter. By default, the output field delimiter is the same as the input field delimiter (the TAB character unless you change it using **–d**). Quote ***ochar*** as necessary to protect it from shell expansion.

diff: DISPLAYS THE DIFFERENCES BETWEEN TWO TEXT FILES

The diff (difference) utility displays line-by-line differences between two text files. By default diff displays the differences as instructions that you can use to edit one of the files to make it the same as the other.

The sdiff utility is similar to diff but its output might be easier to read; its output is the same as that of the diff **–y** (**––side-by-side**) option. Use the diff3 utility to compare three files and cmp to compare nontext (binary) files.

ARGUMENTS

The diff utility commonly takes the pathnames of two ordinary files it is to compare as arguments.

When you call diff without any options, it produces a series of lines containing Add (**a**), Delete (**d**), and Change (**c**) instructions. Each of these lines is followed by the lines from the file you need to add to, delete from, or change, respectively, to make the files the same. A less than symbol (**<**) precedes lines from ***file1***. A greater than symbol (**>**) precedes lines from ***file2***. The diff output appears in the format shown in Table 7-1. A pair of line numbers separated by a comma represents a range of lines; a single line number represents a single line.

Table 7-1 diff output

Instruction	**Meaning (to change file1 to file2)**

`line1 a line2,line3`
`> lines from file2`
	Append *line2* through *line3* from **file2** after *line1* in **file1**

`line1,line2 d line3`
`< lines from file1`
	Delete *line1* through *line2* from **file1**

`line1,line2 c line3,line4`
`< lines from file1`
`---`
`> lines from file 2`
	Change *line1* through *line2* in **file1** to *line3* through *line4* from **file2**

The diff utility assumes you will convert the file named by the first argument to the file named by the second argument. The line numbers to the left of each of the **a**, **c**, or **d** instructions always pertain to the first file; the line numbers to the right of the instructions apply to the second file. To display instructions to convert the files in the opposite order, reverse the arguments.