

Learning About Commands

In Chapter 2, “The Basics,” you started learning about some basic system commands. You covered a lot, but even so, much was left out. The `ls` command is an incredibly rich, powerful tool, with far more options than were provided in Chapter 2. So how can you learn more about that command or others that pique your interest? And how can you discover commands if you don’t even know their names? That’s where this chapter comes in. Here you will find out how to learn more about the commands you already know, those you know you don’t know, and even those that you don’t know that you don’t know!

Let’s start with the two 800-pound gorillas—`man` and `info`—and move from there to some smaller, more precise commands that actually use much of the data collected by `man`. By the time you’re finished, you’ll be ready to start learning about the huge variety of tools available to you in your shell environment.

Find Out About Commands with `man`

`man 1s`

Want to find out about a Linux command? Why, it's easy! Let's say you want to find out more about the `1s` command. Enter `man 1s`, and the `man` (short for *manual*) page appears, chock full of info about the various facets of `1s`. Try the same thing for some of the other commands you've examined in this book. You'll find `man` pages for (almost) all of them.

Still, as useful as `man` pages are, they still have problems. You have to know the name of a command to really use them (although there are ways around that particular issue), and they're sometimes out of date and missing the latest features of a command. They don't always exist for every command, which can be annoying. But worst of all, even if you find one that describes a command in which you're interested and it's up to date, you might still have a big problem: It might be next to useless.

The same developers who write the programs usually (but not always) write the `man` pages. Most of the developers who write applications included with a Linux distribution are excellent programmers, but not always effective writers or explainers of their own work. They know how things work, but they too often forget that users don't know the things that the developers find obvious and intuitive.

With all of those problems, however, `man` pages are still a good resource for Linux users at all levels of experience. If you're going to use Linux on the command line, you need to learn how to use and read `man` pages.

As stated before, using this command isn't hard. Just enter **man**, followed by the command about which you want to learn more.

```
$ man ls
LS(1)                User Commands                LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILES (the current
  directory by default).
  Sort entries alphabetically if none of -cftuSUX
  nor --sort.
  Mandatory arguments to long options are mandatory
  for short options too.
  -a, --all
    do not hide entries starting with .
  -A, --almost-all
    do not list implied . and ..
[Listing condensed due to length]
```

The list of data man provides in this case is quite extensive—over 200 lines worth, in fact. Of course, not all commands provide that much, and some provide far more. Your job is to read the various sections provided in a man page, which usually (but not always) consist of the following:

- **NAME**—The name of the command and a brief description
- **SYNOPSIS**—The basic format of the command
- **DESCRIPTION**—A longer overview of the command's purpose

- **OPTIONS**—The real meat and potatoes of man: The various options for the command, along with short explanations of each one
- **FILES**—Other files used by the command
- **AUTHOR**—Who wrote the command, along with contact information
- **BUGS**—Known bugs and how to report new ones
- **COPYRIGHT**—This one's obvious: Information about copyright
- **SEE ALSO**—Other, related commands

Moving around in a man page isn't that hard. To move down a line at a time, use the down arrow; to move up a line at a time, use the up arrow. To jump down a page, press the spacebar or `f` (for *forward*); to jump up a page, press `b` (for *backward*). When you reach the end of a man page, `man` might quit itself, depositing you back on the shell; it might, however, simply stop at the end without quitting, in which case you should press `q` to quit the program. In fact, you can press `q` at any time to exit `man` if you're not finding the information you want.

It can be hard to find a particular item in a man page, so sometimes you need to do a little searching. To search on a man page after it's open, type `/`, followed by your search term, and then press Enter. If your term exists, you'll jump to it; to jump to the next occurrence of the term, press Enter again (or `n`), and keep pressing Enter (or `n`) to jump down the screen to each occurrence; to go backward, press `Shift+n`.

Search for a Command Based on What It Does

```
man -k
```

With just a little training, you find that you can zoom around man pages and find exactly what you need...assuming you know which man page to read. What if you know a little about what a command does, but you don't know the actual name of the command? Try using the `-k` option (or `--apropos`) and search for a word or phrase that describes the kind of command you want to discover. You'll get back a list of all commands whose name or synopsis matches your search term.

```
$ man list
```

```
No manual entry for list
```

```
$ man -k list
```

```
last (1) - show listing of last logged in users
```

```
ls (1) - list directory contents
```

```
lshal (1) - List devices and their properties
```

```
lshw (1) - list hardware
```

```
lsof (8) - list open files
```

```
[Listing condensed due to length]
```

Be careful with the `-k` option, as it can produce a long list of results, and you might miss what you were looking for. Don't be afraid to try a different search term if you think it might help you find the command you need.

TIP: The `-k` option, also represented by `--apropos`, is exactly the same as the `apropos` command.

Quickly Find Out What a Command Does Based on Its Name

```
man -f
```

If you know a command's name but don't know what it does, there's a quick and dirty way to find out without requiring you to actually open the man page for that command. Use the `-f` option (or `--what-is`), and the command's synopsis appears.

```
$ man -f ls
```

```
ls (1) - list directory contents
```

TIP: Yes, the `-f` option, also known as `--what-is`, is the spitting image of the `what-is` command, which is examined in more detail later in this chapter.

Rebuild man's Database of Commands

```
man -u
```

Occasionally you'll try to use `man` to find out information about a command and `man` reports that there is no page for that command. Before giving up, try again with the `-u` option (or `--update`), which forces `man` to rebuild the database of commands and man pages it uses. It's often a good first step if you think things aren't quite as they should be.

```
$ man ls
No manual entry for ls
$ man -u ls
LS(1)                User Commands                LS(1)
NAME
    ls - list directory contents
SYNOPSIS
    ls [OPTION]... [FILE]...
[Listing condensed due to length]
```

Read a Command's Specific Man Page

man [1-8]

You might notice in the previous listing that the first line of man's page on `ls` references `LS(1)`, while earlier, when you used the `-k` option, all the names of commands were also followed by numbers in parentheses. Most of them are 1, but one, `lsuf`, is 8. So what's up with all of these numbers?

The answer is that man pages are categorized into various sections, numbered from 1 to 8, which break down as follows (and don't worry if you don't recognize some of the example, as many of them are pretty arcane and specialized):

1. General commands. Examples are `cd`, `chmod`, `lp`, `mkdir`, and `passwd`.
2. Low-level system calls provided by the kernel. Examples are `intro` and `chmod`.
3. C library functions. Examples are `beep`, `HTML::Parser`, and `Mail::Internet`.

4. Special files, such as devices found in `/dev`. Examples are `console`, `lp`, and `mouse`.
5. File formats and conventions. Examples are `apt.conf`, `dpkg.cfg`, `hosts`, and `passwd`.
6. Games. Examples are `atlantik`, `bouncingcow`, `kmahjongg`, and `rubik`.
7. Miscellanea, including macro packages. Examples are `ascii`, `samba`, and `utf-8`.
8. System administration commands used by `root`. Examples are `mount` and `shutdown`.

Almost every command we've looked at so far in this book falls into section 1, which isn't surprising because we're focused on general use of your Linux system. But notice how some commands fall into more than one section: `chmod`, for instance, is in both 1 and 2, while `passwd` can be found in 1 and 5. By default, if you enter `man passwd` in your shell, `man` defaults to the lower number, so you'll get the section 1 man page for `passwd`, which isn't very helpful if you want to learn about the file `passwd`. To see the man page for the file `passwd`, follow `man` with the section number for the data you want to examine.

```
$ man passwd
PASSWD(1)                                PASSWD(1)
NAME
  passwd - change user password
SYNOPSIS
  passwd [-f|-s] [name]
  passwd [-g] [-r|-R] group
  passwd [-x max] [-n min] [-w warn] [-i inact]
login
  passwd {-l|-u|-d|-S|-e} login
```

DESCRIPTION

passwd changes passwords for user and group
→accounts. A normal user...

[Listing condensed due to length]

\$ man 5 passwd

PASSWD(5)

PASSWD(5)

NAME

passwd - The password file

DESCRIPTION

passwd contains various pieces of information for
→each user account.

[Listing condensed due to length]

Print Man Pages

```
man -t
```

As easy as it is to view man pages using a terminal program, sometimes it's necessary to print out a man page for easier reading and cogitation. Printing a man page isn't a one-step process, however, and the commands that round out this particular section are actually using principles that will be covered extensively later. But if you want to print out a man page, this is the way to do it. Just have faith that you'll understand what these commands mean more fully after reading upcoming chapters.

Let's say you have a printer that you have identified with the name `hp_laserjet` connected to your system. You want to print a man page about the `ls` command directly to that printer, so use the `-t` option (or `--troff`) and then pipe the output to the `lpr` command, identifying your printer with the `-P` option.

```
$ man -t ls | lpr -P hp_laserjet
```

NOTE: You'll learn about the pipe symbol (`|`) in Chapter 4, "Building Blocks," and `lpr` in Chapter 6, "Printing and Managing Print Jobs."

In just a moment or two, depending upon the speed of your computer and your printer, `hp_laserjet` should begin spitting out the man pages for `ls`. Maybe you don't want to actually print the pages, however. Maybe just creating PDFs of the man page for the `ls` command would be enough. Once again, the commands to do so might seem a bit arcane now, but they will be far more comprehensible very soon.

Again, you use the `-t` option, but this time you send the output to a PostScript file named for the `ls` command. If that process completes successfully, you convert the PostScript file to a PDF using the `ps2pdf` command, and then after that finishes correctly, you delete the original PostScript file because it's no longer needed.

```
$ man -t ls > ls.ps && ps2pdf ls.ps && rm ls.ps
```

NOTE: You'll learn about the `>` and `&&` symbols in Chapter 4 and `ps2pdf` in Chapter 6.

If you want to make a printed library of man pages covering your favorite commands, or if you want that library in PDF format (which can then be printed to hard copies, if necessary), now you know how to do it. For something so simple, `man` is powerful and flexible, which makes it even more useful than it already is.

Learn About Commands with `info`

`info`

The `man` command, and the resulting `man` pages, are simple to work with, even if the content isn't always as user friendly as you might like. In response to that and other perceived shortcomings with `man` pages, the GNU Project, which is responsible in many ways for many of the commands you're reading about in this book, created its own format: Info pages, which use the `info` command for viewing.

Info pages tend to be better written, more comprehensive, and more user friendly in terms of content than `man` pages, but `man` pages are far easier to use. A `man` page is just one page, while Info pages almost always organize their contents into multiple sections, called *nodes*, which might also contain subsections, called *subnodes*. The trick is learning how to navigate not just around an individual page, but also between nodes and subnodes. It can be a bit overwhelming at first to just move around and find what you're seeking in Info pages, which is rather ironic: Something that was supposed to be nicer for newbies than `man` is actually far harder for the novice to learn and use.

There are many facets to `info`, and one of the best commands to use `info` against is, in fact, `info`. To learn how to use `info`, as well as read about `info`, enter the following command:

```
$ info info
```

This opens the Info page for the `info` command. Now you need to learn how to get around in this new Info world.

Navigate Within `info`

Within a particular section's screen, to move down, or forward, one line at a time, use the down arrow; to move up, or back, one line at a time, use the up arrow. When you reach the bottom, or end, of a particular section, your cursor stops and you are not able to proceed.

If you instead want to jump down a screen at a time, use your keyboard's PageDown button; to jump up a screen at a time, use the PageUp key instead. You are not able to leave the particular section you're in, however.

If you reach the end of the section and you want to jump back to the top, just press `b`, which stands for *beginning*. Likewise, `e` takes you to the "end."

If, at any time, as you're jumping around from place to place, you notice that things look a little strange, such as the letters or words being distorted, press `Ctrl+l` to redraw the screen, and all should be well.

Now that you know how to navigate within a particular section or node, let's move on to navigating between nodes. If you don't want to use PageDown and PageUp to move forward and backward within a section, you can instead use the spacebar to move down and the Backspace or Delete keys to move up. These keys offer one big advantage over PageDown and PageUp, besides being easier to reach: When you hit the end of a node, you automatically proceed onward to the next node, through subnodes if they exist. Likewise, going up moves you back to the previous node, through any

subnodes. Using the spacebar, or the Backspace or Delete buttons, you can quickly run through an entire set of Info pages about a particular command.

If you want to engage in fewer key presses, you can use *n* (for *next*) to move on to the next node at the same level. If you are reading a node that had subnodes and you press *n*, you skip those subnodes and move to the next node that's a peer of the one you're currently reading. If you reading a subnode and press *n*, however, you jump to the next subnode. If *n* moves you to the next node at the current level, then *p* moves you to the previous one (*p* for *previous*, get it?), again at the same level.

If you instead want to move forward to a node or subnode, use the *]*, or right square brace, key. If you're reading a node and you press *]*, you'll jump to that node's first subnode, if one exists. Otherwise, you'll move to that node's next peer node. To move backward in the same fashion, use the *[*, or left square brace, key.

If you want to move up a node, to the parent of the node you're currently reading, use the *u* (for *up*) key. Be careful, though—it's easy to jump up past the home page of the command you're reading about in Info to what is termed the Directory node, the root node that leads to all other Info nodes (another way to reach the Directory node is to type *d*, for *directory*, at any time).

The Directory node is a particularly large example of a type of page you find all throughout Info: a Menu page, which lists all subnodes or nodes. If you ever find yourself on a Menu page, you can quickly navigate to one of the subnodes listed in that menu in one of two ways. First, type an *m* (for *menu*) and then start typing the name of the subnode to which you want to jump. For instance, here's the first page you see when you enter **info info** on the command line:

```
File: info.info, Node: Top, Next: Getting Started,
↳Up: (dir)
```

```
Info: An Introduction
```

```
*****
```

```
The GNU Project distributes most of its on-line
manuals in the "Info format", which you read using
an "Info reader". You are probably using an Info
reader to read this now.
```

```
[content condensed due to length]
```

```
* Menu:
```

```
* Getting Started:: Getting started using an Info
↳reader.
```

```
* Expert Info:: Info commands for experts.
```

```
* Creating an Info File:: How to make your own Info
↳file.
```

```
* Index:: An index of topics, commands, and
↳variables.
```

To jump to Expert Info, you could type **m**, followed by **Exp**. At this point, you could finish typing **ert Info**, or you could just press the Tab key and Info fills in the rest of the menu name that matches the characters you've already entered. If Info complains, you have entered a typo, or more than one menu choice matches the characters you've typed. Fix your typo, or enter more characters until it is obvious to Info which menu choice is the one in which you're interested. If you realize that you don't want to go to a Menu choice at this time, press **Ctrl+g** to cancel your command, and go back to reading the node on which you find yourself.

Alternatively, you could just use your up or down arrow key to position the cursor over the menu choice you want, and then press **Enter**. Either method works.

If you don't want to navigate Info pages and instead want to search, you can do that as well, in two ways: By searching just the titles of all nodes for the Info pages about a particular command, or by searching in the actual text of all nodes associated with a particular command. To search the titles, enter **i** (for *index* because this search uses the node index created by Info), followed by your search term, and press Enter. If your term exists somewhere in a node's title, you'll jump to it. If you want to repeat your search and go to the next result, press the comma key.

If you want to search text instead of titles, enter **s** (for *search*), followed by your search term or phrase, and then press Enter. To repeat that search, enter **s**, followed immediately by Enter. That's not as easy as just pressing the comma key like you do when you're searching titles, but it does work.

If at any time you get lost inside Info and need help, just press the **?** key and the bottom half of your window displays all of the various options for Info. Move up and down in that section using the keys you've already learned. When you want to get out of help, press **l**.

Finally, and perhaps most importantly, to get out of Info altogether, just press **q**, for *quit*, which dumps you back in your shell. Whew!

Locate the Paths for a Command's Executable, Source Files, and Man Pages

whereis

The `whereis` command performs an incredibly useful function: It tells you the paths for a command's

executable program, its source files (if they exist), and its man pages. For instance, here's what you might get for KWord, the word processor in the KOffice set of programs (assuming, of course, that the binary, source, and man files are all installed):

```
$ whereis kword
```

```
kword: /usr/src/koffice-1.4.1/kword /usr/bin/kword  
↳/usr/bin/X11/kword usr/share/man/man1/kword.1.gz
```

The `whereis` command first reports where the source files are: `/usr/src/koffice-1.4.1/kword`. Then it informs you as to the location of any binary executables: `/usr/bin/kword` and `/usr/bin/X11/kword`. KWord is found in two places on this machine, which is a bit unusual but not bizarre. Finally, you find out where the man pages are: `/usr/share/man/man1/kword.1.gz`. Armed with this information, you can now verify that the program is in fact installed on this computer, and you know now how to run it.

If you want to search only for binaries, use the `-b` option.

```
$ whereis -b kword
```

```
kword: /usr/bin/kword /usr/bin/X11/kword
```

If you want to search only for man pages, the `-m` option is your ticket.

```
$ whereis -m kword
```

```
kword: /usr/share/man/man1/kword.1.gz
```

Finally, if you want to limit your search only to sources, try the `-s` option.

```
$ whereis -s kword
```

```
kword: /usr/src/koffice-1.4.1/kword
```

The `whereis` command is a good, quick way to find vital information about programs on the computer you're using. You'll find yourself using it more than you think.

Read Descriptions of Commands

`whatis`

Earlier in this chapter you found out about the `-f` option for `man`, which prints onscreen the description of a command found in a man page. If you can remember that `man -f` presents that information, bully for you. It might be easier, however, to remember the `whatis` command, which does exactly the same thing: Displays the man page description for a command.

```
$ man -f ls
ls (1) - list directory contents
$ whatis ls
ls (1) - list directory contents
```

The `whatis` command also supports regular expressions and wildcards. To search the man database using wildcards, use the `-w` option (or `--wildcard`).

```
$ whatis -w ls*
ls (1) - list directory contents
lsb (8) - Linux Standard Base support for Debian
lshad (1) - List devices and their properties
lshw (1) - list hardware
lskat (6) - Lieutenant Skat card game for KDE
[Listing condensed due to length]
```

Using wildcards might result in a slightly slower search than `whatis` without options, but it's pretty negligible

on today's fast machines, so you probably don't have to worry about it.

Regular expressions can be used with the `-r` (or `--regex`) option.

```
$ whatis -r ^rm.*
```

```
rm (1) - remove files or directories
```

```
rmail (8) - handle remote mail received via uucp
```

```
rmdir (1) - remove empty directories
```

```
rmt (8) - remote magtape protocol module
```

TIP: There's not enough room in this book to cover regular expressions, but you can read more in *Sams Teach Yourself Regular Expressions in 10 Minutes* (ISBN: 0672325667) by Ben Forta.

Again, regular expressions are supposed to slow down the response you get with `whatis`, but you'll probably never notice it.

The `whatis` command is easy to remember (easier than `man -f`, some would say) and it quickly returns some important information, so memorize it.

Find a Command Based on What It Does

`apropos`

The `whatis` command is similar to `man -f`, and the `apropos` command is likewise similar to `man -k`. Both commands search `man` pages for the names and descriptions of commands, helping you if you know what a command does but can't remember its name.

NOTE: The word *apropos* isn't exactly in common usage, but it is a real word, and it means, essentially, "relevant" or "pertinent." The word *appropriate* has a somewhat similar meaning, but it's based on a different Latin root than *apropos*. Check the words out at www.answers.com for yourself if you're a fellow linguophile.

Using *apropos* is easy: Just follow the command with a word or phrase describing the feature of the command in which you're interested.

```
$ man list
```

```
No manual entry for list
```

```
$ man -k list
```

```
last (1) - show listing of last logged in users
```

```
ls (1) - list directory contents
```

```
lshw (1) - list hardware
```

```
lsuf (8) - list open files
```

```
[Listing condensed due to length]
```

```
$ apropos list
```

```
last (1) - show listing of last logged in users
```

```
ls (1) - list directory contents
```

```
lshw (1) - list hardware
```

```
lsuf (8) - list open files
```

```
[Listing condensed due to length]
```

Just like *whatis*, you can use the *-w* (or *--wildcard*) or the *-r* (or *--regex*) options for searches. More interestingly, though, you can use the *-e* option (or *--exact*) when you want to tightly focus on a word or phrase, without any exception. For instance, in the previous listing, searching for *list* turned up the *last* command because it had the word *listing* in its description. Let's try that same search, but with the *-e* option.

```
$ apropos -e list
```

```
ls (1) - list directory contents
```

```
lshw (1) - list hardware
```

```
lsdf (8) - list open files
```

```
[Listing condensed due to length]
```

This time, `last` doesn't show up because you wanted only results with the exact word *list*, not *listing*. In fact, the list of results for *list* went from 80 results without the `-e` option to 55 results with the `-e` option, which makes it far easier to precisely target your command searches and find exactly the command you want.

Find Out Which Version of a Command Will Run

```
which
```

Think back to the `whereis` command and what happened when you ran it against `KWord` using the `-b` option, for *show binaries only*.

```
$ whereis -b kword
```

```
kword: /usr/bin/kword /usr/bin/X11/kword
```

The executable for `KWord` is in two places. But which one will run first? You can tell that by running the `which` command.

```
$ which kword
```

```
/usr/bin/kword
```

The `which` command tells you which version of a command will run if you just type its name. In other words, if you type in `kword` and then press Enter, your

shell executes the one found inside `/usr/bin`. If you want to run the version found in `/usr/bin/X11`, you have to change directories using the `cd` command and then enter `./keyword`, or use the absolute path for the command and type out `/usr/bin/X11/keyword`.

The `which` command is also a speedy way to tell if a command is on your system. If the command is on your system and in your `PATH`, you'll be told where to find it; if the command doesn't exist, you're back on the command line with nothing.

```
$ which arglebargle
$
```

If you want to find all the locations of a command (just like you would if you used `whereis -b`), try the `-a` (for *all*) option.

```
$ which -a keyword
/usr/bin/keyword
/usr/bin/X11/keyword
```

Conclusion

The title of this chapter is “Learning About Commands,” and that’s what we’ve covered. By now, you’ve seen that there are a variety of ways to find out more about your options on the command line. The two big dogs are `man` and `info`, with their volumes of data and descriptions about virtually all the commands found on your Linux computer. Remember that `whereis`, `whatis`, `apropos`, and `which` all have their places as well, especially if your goal is to avoid having to wade through the sometimes overwhelming verbiage

of `man` and `info`, an understandable but often impossible goal. Sometimes you just have to roll up your sleeves and start reading a man page. Think of it like broccoli: You might not enjoy it, but it's good for you.

It's true that many of the commands in this chapter overlap to a degree. For instance, `man -k` is the same as `apropos` and `man -f` is just like `whatis`, while `whereis -b` is functionally equivalent to `which -a`. The choice of which one to use in a given situation is up to you. It's still a good idea, however, to know the various similarities, so you'll be able to read shell scripts or instructions given by others and understand exactly what's happening. Linux is all about variety and choice, even in such seemingly small matters as commands run in the shell.