8

In this chapter:
- *Documentation*
- *Shell Aliases and Functions*
- *Programming*
- *Using Unix on Non-Unix Systems*

Where to Go from Here

Now that you're almost to the end of this guide, let's look at some ways to continue learning about Unix. Documentation is an obvious choice, but it isn't always in obvious places. You can save time by taking advantage of other shell features—aliases, functions, and scripts—that let you shorten a repetitive job and "let the computer do the dirty work."

We'll close by seeing how you can use Unix commands on non-Unix systems.

Documentation

You might want to know the options to the programs we've introduced—and get more information about them and the many other Unix programs. You're now ready to consult your system's documentation and other resources.

The man Command

Different versions of Unix have adapted Unix documentation in different ways. Almost all Unix systems have documentation derived from a manual originally called the *Unix Programmer's Manual*. The manual has numbered sections; each section is a collection of manual pages, often called "manpages"; each program has its own manpage. Section 1 has manpages for general Unix programs such as **who** and **ls**.

Many Unix installations have individual manual pages stored on the computer; users can read them online. If your system has online manpages, and you want to know the correct syntax for entering a command or the particular features of a program, enter the command **man** and the name of the command. The syntax is:

man *command*

For example, if you want to find information about the program **mail**, which allows you to send messages to other users, enter:

```
$ man mail
         .
         .
$
```

The output of **man** may be filtered through a pager like **less** automatically. If it isn't, just pipe the output of **man** to **less** (or **more** or **pg**).

After you enter the command, the screen fills with text. Press $\boxed{\text{SPACE}}$ or $\boxed{\text{RETURN}}$ to read more, and **q** to quit.

Some systems also have a command called **apropos** or **man -k** to help you locate a command if you have an idea of what it does but are not sure of its correct name. Enter **apropos** followed by a descriptive word; you'll get a list of commands that might help.

Problem checklist

man *says there is no manual entry for the command.*

Some commands—**cd** and **jobs**, for example—aren't separate Unix programs; they're part of the shell. On some Unix systems, you'll find documentation for those commands in the manual page for the shell. (To find the shell's name, see the section "The Unix Shell" in Chapter 1.)

If the program isn't a standard part of your Unix system—that is, your system staff added the program to your system—there may not be a manual page, or you may have to configure the **man** program to find the local manpage files.

The info Command

Linux systems, as well as some others, have a program called **info**. It serves the same purpose as **man**: to document system programs. The **info** output is in a different format, though. The syntax to start **info** is:

info *command*

For example, if you want to find information about the program **find**, which searches for files, enter **info find**. After you enter the command, press $\boxed{\text{SPACE}}$ to read more or "q" to quit.

Documentation via the Internet

The Internet changes so quickly that any list of online Unix documentation we'd give you would soon be out of date. Still, the Internet is a great place to find out about Unix systems. Remember that there are many different versions of Unix—so some documentation you find may not be completely right for you. Also, some information you'll find may be far too technical for your needs (many computer professionals use and discuss Unix). But don't be discouraged! Once you've found a site with the general kind of information you need, you can probably come back later for more.

Many Unix command names are plain English words, which can make searching hard. If you're looking for collections of Unix information, try searching for the Unix program named **grep**. As this book went to press, one especially Unix-friendly search engine was Google, at *http://www.google.com.*

Here are some other places to try:

- **Magazines**, both in print and online-only, have Unix tutorials and links to more information. Many are written for beginners.
- **Publishers**, like O'Reilly & Associates, Inc. (*http://www.oreilly.com*), have areas of their websites that feature Unix and have articles written by their books' authors. They may also have books online (such as the O'Reilly Safari service) available for a small monthly fee—which is a good way to learn a lot quickly without needing to buy a paper copy of a huge book, most of which you may not need.
- **Vendors' sites** like Red Hat (*http://www.redhat.com*), and Unix-related organizations like the Free Software Foundation (*http://www.fsf.org*), usually have documentation and support files online, where you can search for what you need.
- **Universities** often use Unix-like systems and will have online documentation. You'll probably have better luck at the Computer Services division (which services the whole campus) than at the Computer Science department (which may be more technical).

Books

Bookstores, both traditional and online, are full of computer books. The books are written for a wide variety of needs and backgrounds. Unfortunately, many books are rushed to press, written by authors with minimal Unix experience, full of errors. Before you buy a book, read through parts of it. Does the style (brief or lots of detail, chatty and friendly or organized as a reference) fit your needs? Search the Internet for reviews; online bookstores may have readers' comments on file.

Shell Aliases and Functions

If you type command names that are hard for you to remember, or command lines that seem too long, you'll want to learn about *shell aliases* and *shell functions*. These shell features let you abbreviate commands, command lines, and long series of commands. In most cases, you can replace them with a single word or a word and a few arguments. For example, one of the long pipelines the section "Pipes and Filters" (Chapter 5) could be replaced by an alias or function named (for instance) "aug." When you type **aug** at a shell prompt, the shell would list files modified in August, sorted by size.

Making an alias or function is almost as simple as typing in the command line or lines that you want to run. References in the section "Documentation," earlier in this chapter, have more information. Shell aliases and functions are actually a simple case of shell programming.

Programming

We mention earlier that the shell is the system's command interpreter. It reads each command line you enter at your terminal and performs the operation that you call for. Your shell is chosen when your account is set up.

The shell is just an ordinary program that can be called by a Unix command. However, it contains some features (such as variables, control structures, and so on) that make it similar to a programming language. You can save a series of shell commands in a file, called a *shell script*, to accomplish specialized functions.

Programming the shell should be attempted only when you are reasonably confident of your ability to use Unix commands. Unix is quite a powerful tool and its capabilities become more apparent when you try your hand at shell programming.

Take time to learn the basics. Then, when you're faced with a new task, take time to browse through references to find programs or options that will help you get the job done more easily. Once you've done that, learn how to build shell scripts so that you never have to type a complicated command sequence more than once.

You might also want to learn Perl. Like the shell, Perl interprets script files full of commands. But Perl has a steeper learning curve than the shell. Also, since you've already learned a fair amount about the shell and Unix commands by reading this book, you're almost ready to start writing shell scripts now; on the other hand, Perl will take longer to learn. But if you have sophisticated needs, learning Perl is another way to use even more of the power of your Unix system.

Using Unix on Non-Unix Systems

Once you get comfortable working quickly at the Unix command line, you may miss that power and flexibility when you use another system like Microsoft Windows. You can get programs—both commercial and freely available—that let you use a Unix-like shell prompt and Unix utilities (**grep**, **sort**, and so on) from within other operating systems. You'll also find that an increasing number of systems are built on top of the stable Unix or a Unix-like operating system. Two of the latest examples are Mac OS X on the Macintosh and a variety of machines with Linux embedded inside.

Unix, Microsoft Windows, and the Macintosh all use different conventions for the way that they mark the end of a line of text. If you transfer text files between these systems, you'll probably need to convert them. (The command-line FTP client does this automatically if you set its ascii transfer mode.) And if you have an executable program file that runs on one system, it won't run on the others—unless it's written in Java or it's a *script* program from a language such as the shell or Perl.

Under Microsoft Windows

Cygwin, from *http://www.cygwin.com*, is a package of Unix-like software development utilities that runs under Microsoft Windows NT, 98, and 95 (and probably others, as Microsoft Windows evolves). Although it's aimed at software developers, it also has a lot of the standard Unix utilities. You can use Cygwin from its **bash** shell (a Unix-like shell) or from the standard Windows command shell.

The *MKS Toolkit*, from *http://www.mks.com*, is a commercial package of Unix-like utilities that runs under Microsoft Windows. MKS Toolkit has been on the market, and been updated constantly, since the time of MS-DOS in the 1980s.

With a little hunting, you'll find versions of other Unix programs for Windows systems. Three of these are the Pine email program, the Lynx browser, and **vim**, a version of the **vi** text editor.

Mac OS X

The latest version of the Macintosh operating system (as of this writing) is Mac OS X, a Unix-based system. The OS X window system, Aqua, interacts with the operating system much as the X Window System you've seen in this book. (In fact, you now can use X on the Mac!)

If you want to use a Unix-like terminal under OS X, you can open Terminal. It's a regular double-clickable application found in */Applications/Utilities*. Navigate to it using the Finder, launch it, and you'll get a terminal window like the ones shown in this book.

Once you open Terminal, you can use standard Unix utilities on your Mac files, on files you create with those utilities, or on files you transfer over a network. File pathnames are separated by slashes (/), just as on Unix, but be sure to put quotes around Mac filenames that don't follow our file naming rules (see the section "File and Directory Names" in Chapter 4). Unlike Unix and Windows, some Macintosh files have two *forks*: the resource and data forks. If you copy a Mac file, watch out—the **cp** utility won't copy both forks! Instead, you'll need to install and run **CpMac** (from the Developer Tools CD that comes with OS X; then you can simply run **/Developer/Tools/CpMac**).