CHAPTER 7

Welcome to the Command Line

- Starting Up the Terminal
- Getting Started
- Building Pipelines
- Running Commands as Superuser
- Finding Help
- Moving around the Filesystem
- Manipulating Files and Folders
- System Information Commands
- Searching and Editing Text Files
- Dealing with Users and Groups
- Getting Help on the Command Line
- Searching for Man Files
- Using Wildcards
- Executing Multiple Commands
- Moving to More Advanced Uses of the Command Line

ONE OF THE MOST powerful parts of any Ubuntu system is the command line. It can also be one of the most daunting to dive into. It seems there is often little help, and that the commands are not easy to find or figure out. If you are willing to learn, the power of the command line will speed up your work and will be a great education that serves you for years by increasing your ability to do exactly what you want to do with your computer with greater efficiency.

While the command line is a nice addition to a desktop user's life, it is completely invaluable if you run a server. The Ubuntu server installs without any GUI, so the tools explained in this chapter and other books are absolutely critical to success. And hey, remember to have fun!

Starting Up the Terminal

The terminal can be found by clicking the tooltip or Dash Home icon (Figure 7-1)—it has the Ubuntu symbol on it and should be the first icon found at the top of your Launcher. This opens the Dash, and in the dash, type *terminal* to find Terminal (Figure 7-2). When it first launches, you will see something similar to what Figure 7-3 shows.

You will see a blinking cursor immediately preceded by some letters, and perhaps numbers and symbols, ending with a `$`. The first word in that string of characters is your username, followed by the `@` symbol. After the `@`, the hostname of your computer is listed, followed by a colon and the name of the directory you are currently in (you always start in your home directory, which is represented by a ~ symbol).

There are many dozens of commands. This chapter presents just a few useful ones in a narrative style to get you started, then lists some more with just a basic description and broken down by category.

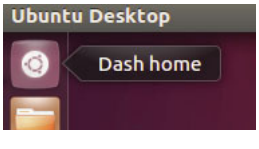

Figure 7-1 The Dash Home icon

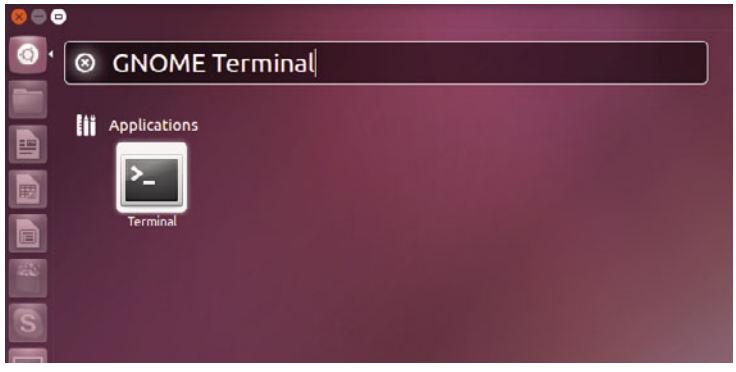

Figure 7-2 The Dash with the Terminal search

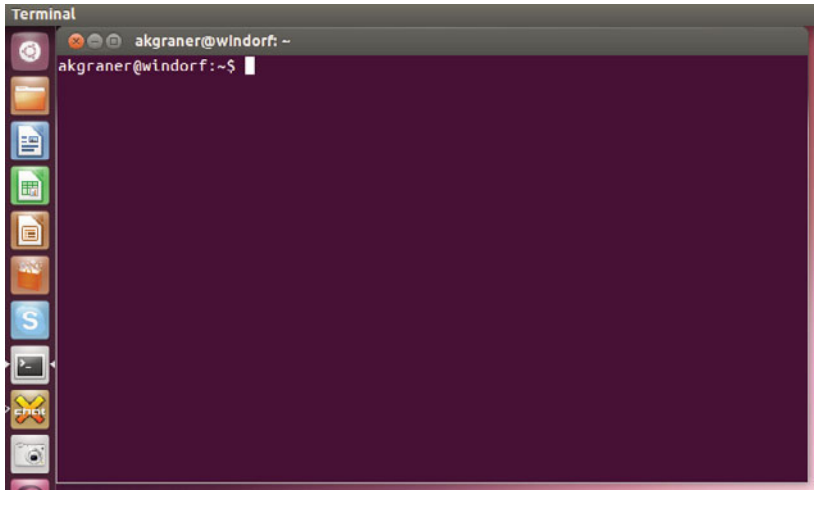

Figure 7-3 The Terminal window

Getting Started

First have a look at the files in your home folder by running the following command:

```
username@computer:~$ ls
```

The `ls` command lists the files in your current folder. The default command just displays a collection of items that are in your current directory,

or location in the filesystem. To make `ls` more useful, you can type it with options:

```
username@computer:~$ ls -al
```

The `-al` parts are options that can be passed to the command. In this example, two options, `a` (list all files) and `l` (use a long display format to display file permissions, dates, sizes, and more), are used with `ls` to display all of the files (including hidden files) and their details.

TIP **To Dash or Not to Dash?**
In many command-line tools, options are added after a dash (-). Some tools, however, don't need the dash. It isn't particularly consistent, so you must pay attention as you learn new commands.

Now move to a different directory:

```
username@computer:~$ cd Desktop
```

The `cd` command changes the directory to the place you specify after the command (in this case, the desktop directory). A nice shortcut that you can use when typing files and folders is to type the first few letters and then press the Tab key to fill in the remainder of the file/folder name. As an example, in the previous command, you could type `cd Des` and press the Tab key to fill in the rest of `Desktop` (with the / added automatically because it is a directory).

When inside a directory, you may want to have a quick look at the contents of a text file. To do this, use the `cat` command:

```
username@computer:~$ cat myfile.txt
```

This command prints the contents of the file on the screen (a more correct way to say this in computer geek jargon would be "outputs to the screen").

Building Pipelines

The power of the command line really comes into its own when you start to pass the output of one command so that it goes to the input of the next, combining commands by using pipelines. A pipeline uses the pipe symbol

(|) to string together a number of commands to perform a specific task. As an example, if you use the `cat` command to display the contents of a file to the screen, but the file scrolls past you, create a pipeline and use the `less` command so you can browse the file:

```
username@computer:~$ cat foo.txt | less
```

To see how this works, break the command into parts, each separated by the pipe. The output of the part on the left (`cat`'ing the file) is fed into the `less` command on the right, which allows you to browse the file with the arrow keys.

Pipelines can be useful for finding specific information on the system. As an example, if you want to find out how many particular processes are running, you could run a command like this:

```
username@computer:~$ ps ax | grep getty | wc -l
```

Here you count how many `getty` processes are running (`getty` is the software that runs a console session). The `ps ax` command on the left lists the processes on the system, and then the `grep` command searches through the process list and returns only the lines that contain the text "getty." Finally, these lines are fed into `wc`, which is a small tool that counts the number of words or lines. The `-l` option specifies that the number of lines should be counted. Cool, huh?

Running Commands as Superuser

When you log in to your computer, the account you use is a normal user account. This account is restricted from performing various system administration tasks. The security model behind Ubuntu has you run as a normal user all the time and dip into the system administrator account only when you need to. This prevents accidental changes or malicious installation of unwanted programs and similar things.

To jump to this superuser account when using the terminal, put the `sudo` command before the command you want to run. As an example, if you want to restart the networking system from the command line, run:

```
username@computer:~$ sudo apt-get install byobu
```

The command to the right of `sudo` is the command that should be run as the administrator, but `sudo` lets you run the command as the current user. When you run the above command, you are asked for the administrator password. This is the same password as the one you established for the first user you added when you installed Ubuntu on the computer. If you are using that user's account, just enter your normal password. In this instance, you are installing byobu, which you can use to make your life in the terminal easier.

TIP **Byobu**
Byobu is a Japanese term for decorative, multipanel screens that serve as folding room dividers. As an open source project, byobu is an elegant enhancement of the otherwise functional, plain, practical GNU screen. Byobu includes an enhanced profile and configuration utilities for the GNU screen window manager, such as toggle-able system status notifications. More about byobu can be found at http://manpages.ubuntu.com/manpages/precise/en/man1/byobu.1.html.

When you have authenticated yourself to `sudo` using the terminal, you will not be asked for the password again for another 15 minutes.

Finding Help

Each command on your computer includes a manual page—or man page—that contains a list of the options available. Man pages are traditionally rather terse and intended only for referencing the different ways the command should be used. For a friendlier introduction to using commands, we recommend a Google search.

To view a man page (such as the man page for `ls`), run:

```
username@computer:~$ man ls
```

The man page command itself has a number of options (run `man man` to see them), and one of the most useful is `-k`. This option allows you to search the man pages for a particular word. This is useful when you don't remember the command. As an example, you could find all commands related to processes by running:

```
username@computer:~$ man -k processes
```

TIP **Ubuntu Manpage Repository**
This site contains hundreds of thousands of dynamically generated manuals from every package of every supported version of Ubuntu, and it's updated on a daily basis. Manpages as described earlier are traditionally viewed from the command line, but thanks to Dustin Kirkland, Ubuntu users now have all the manuals included in Ubuntu in an HTML, Web-browsable format. This repository can be found at http://manpages.ubuntu.com.

The remainder of this chapter gives a brief introduction to some of the more common and useful commands you will encounter and want to learn, organized in categories based on how they are used. We end with a short list of some other resources for further research.

Moving around the Filesystem

Commands for navigating in the filesystem include the following.

- **pwd:** The `pwd` command allows you to know the directory in which you're located (`pwd` stands for "print working directory"). For example, `pwd` in the desktop directory will show ~/Desktop. Note that the GNOME terminal also displays this information in the title bar of its window.
- **cd:** The `cd` command allows you to change directories. When you open a terminal, you will be in your home directory. To move around the filesystem, use `cd`.
  - Use `cd ~/Desktop` to navigate to your desktop directory.
  - Use `cd /` to navigate into the root directory.
  - Use `cd` to navigate to your home directory.
  - Use `cd ..` to navigate up one directory level.
  - Use `cd -` to navigate to the previous directory (or back).
  - If you want to go directly to a specific, known directory location at once, use `cd /directory/otherdirectory`. For example, `cd /var/www` will take you directly to the /www subdirectory of /var.

Manipulating Files and Folders

You can manipulate files and folders with the following commands.

- **cp:** The `cp` command makes a copy of a file for you. For example, `cp` *file* `foo` makes an exact copy of the file whose name you entered and names the copy foo, but the first file will still exist with its original name.
- **mv:** The `mv` command moves a file to a different location or renames a file. Examples are as follows: `mv` *file* `foo` renames the original file to foo. `mv foo ~/Desktop` moves the file foo to your desktop directory but does not rename it. You must specify a new filename to rename a file. After you use `mv`, the original file no longer exists, but after you use `cp`, as above, that file stays and a new copy is made.
- To save on typing, you can substitute `~` in place of the home directory, so `/home/username/pictures` is the same as `~/pictures`.

NOTE If you are using `mv` with `sudo`, which is often necessary outside of your home directory, you will not be able to use the ~ shortcut. Instead, you will have to use the full pathnames to your files.

- **rm:** Use this command to remove or delete a file in your directory, as in `rm file.txt`. It does not work on directories that contain files, which must first be emptied and may then be deleted using the `rmdir` command. There are some advanced cases where you may use `rm` to remove directories, but discussing those are beyond the intent of this chapter.
- **ls:** The `ls` command shows you the files in your current directory. Used with certain options, it lets you see file sizes, when files where created, and file permissions. For example, `ls ~` shows you the files that are in your home directory.
- **mkdir:** The `mkdir` command allows you to create directories. For example, `mkdir music` creates a music directory.
- **chmod:** The `chmod` command changes the permissions on the files listed. Permissions are based on a fairly simple model. You can set permissions for user, group, and world, and you can set whether each

can read, write, and/or execute the file. For example, if a file had permission to allow everybody to read but only the user could write, the permissions would read `rwxr–r–`. To add or remove a permission, you append a `+` or a `-` in front of the specific permission. For example, to add the capability for the group to edit in the previous example, you could type `chmod g+w file`.

- **chown:** The `chown` command allows the user to change the user and group ownerships of a file. For example, `sudo chown jim` *file* changes the ownership of the file to Jim.

System Information Commands

System information commands include the following.

- **df:** The `df` command displays filesystem disk space usage for all partitions. The command `df-h` is probably the most useful. It uses megabytes (`M`) and gigabytes (`G`) instead of blocks to report. (`-h` means "human-readable.")
- **free:** The `free` command displays the amount of free and used memory in the system. For example, `free -m` gives the information using megabytes, which is probably most useful for current computers.
- **top:** The `top` command displays information on your Linux system, running processes, and system resources, including the CPU, RAM, swap usage, and total number of tasks being run. To exit `top`, press Q.
- **uname -a:** The `uname` command with the `-a` option prints all system information, including machine name, kernel name, version, and a few other details. This command is most useful for checking which kernel you're using.
- **lsb_release -a:** The `lsb_release` command with the `-a` option prints version information for the Linux release you're running. For example:

```
username@computer:~$ lsb_release -a

No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 12.04
Release:        12.04
Codename:       precise
```

- **ifconfig:** This reports on your system's network interfaces.
- **iwconfig:** The `iwconfig` command shows you any wireless network adapters and the wireless-specific information from them, such as speed and network connected.
- **ps:** The `ps` command allows you to view all the processes running on the machine.

The following commands list the hardware on your computer, either of a specific type or with a specific method. They are most useful for debugging when a piece of hardware does not function correctly.

- **lspci:** The `lspci` command lists all PCI buses and devices connected to them. This commonly includes network cards and sound cards.
- **lsusb:** The `lsusb` command lists all USB buses and any connected USB devices, such as printers and thumb drives.
- **lshal:** The `lshal` command lists all devices the hardware abstraction layer (HAL) knows about, which should be most hardware on your system.

Searching and Editing Text Files

Search and edit text files by using the following commands.

- **grep:** The `grep` command allows you to search inside a number of files for a particular search pattern and then print matching lines. For example, `grep blah` *file* will search for the text "blah" in the file and then print any matching lines.
- **sed:** The `sed` (or Stream EDitor) command allows search and replace of a particular string in a file. For example, if you want to find the string "cat" and replace it with "dog" in a file named pets, type `sed s/cat/dog/g pets`.

Both `grep` and `sed` are extremely powerful programs. There are many excellent tutorials available on using them, but here are a couple of good Web sites to get you started:

- https://help.ubuntu.com/community/grep
- http://manpages.ubuntu.com/manpages/precise/man1/9base-sed.1.html

Five other commands are useful for dealing with text.

- **cat:** The `cat` command, short for concatenate, is useful for viewing and adding to text files. The simple command `cat` *FILENAME* displays the contents of the file. Using `cat` *FILENAME* `file` adds the contents of the first file to the second and displays both on the screen, one after the other. You could also use `cat file1 >> file2` to append the contents of file1 to the end of file2.
- **nano:** Nano is a simple text editor for the command line. To open a file, use `nano` *filename*. Commands listed at the bottom of the screen are accessed via pressing Ctrl followed by the letter. To close and save a file you are working on, use Ctrl-X.
- **less:** The `less` command is used for viewing text files as well as standard output. A common usage is to pipe another command through `less` to be able to see all the output, such as `ls | less`.
- **head:** The `head` command is used for viewing the first 10 lines of text files as well as standard output. A common usage is to pipe another command through `head` to be able to see varying lines of output, such as `ls | head`.
- **tail:** The tail command is used for viewing the last 10 lines of text files as well as standard output. A common usage is to pipe another command through `tail` to be able to see varying lines of output, such as `ls | tail`.

Dealing with Users and Groups

You can use the following commands to administer users and groups.

- **adduser:** The `adduser` command creates a new user. To create a new user, simply type `sudo adduser` *loginname*. This creates the user's home directory and default group. It prompts for a user password and then further details about the user.

- **passwd:** The `passwd` command changes the user's password. If run by a regular user, it will change his or her password. If run using `sudo`, it can change any user's password. For example, `sudo passwd joe` changes Joe's password.
- **who:** The `who` command tells you who is currently logged into the machine.
- **addgroup:** The `addgroup` command adds a new group. To create a new group, type `sudo addgroup` *groupname*.
- **deluser:** The `deluser` command removes a user from the system. To remove the user's files and home directory, you need to add the `-remove-home` option.
- **delgroup:** The `delgroup` command removes a group from the system. You cannot remove a group that is the primary group of any users.
- **chgrp:** The `chgrp` command changes group ownership of files and directories.

Getting Help on the Command Line

This section provides you with some tips for getting help on the command line. The commands `–help` and `man` are the two most important tools at the command line.

Virtually all commands understand the `-h` (or `–help`) option, which produces a short usage description of the command and its options, then exits back to the command prompt. Try `man -h` or `man –help` to see this in action.

Every command and nearly every application in Linux has a man (manual) file, so finding such a file is as simple as typing `man` *command* to bring up a longer manual entry for the specified command. For example, `man mv` brings up the `mv` (move) manual.

Some helpful tips for using the `man` command include the following.

- **Arrow keys:** Move up and down the man file by using the arrow keys.
- **q:** Quit back to the command prompt by typing `q`.

- **man man:** `man man` brings up the manual entry for the `man` command, which is a good place to start!
- **man intro:** `man intro` is especially useful. It displays the Introduction to User Commands, which is a well-written, fairly brief introduction to the Linux command line.

There are also info pages, which are generally more in-depth than man pages. Try `info info` for the introduction to info pages.

Searching for Man Files

If you aren't sure which command or application you need to use, you can try searching the man files.

- **man -k foo:** This searches the man files for *foo*. Try `man -k nautilus` to see how this works.

NOTE `man -k foo` is the same as the `apropos` command.

- **man -f foo:** This searches only the titles of your system's man files. Try `man -f unity`, for example.

NOTE `man -f foo` is the same as the `whatis` command.

Using Wildcards

Sometimes you need to look at or use multiple files at the same time. For instance, you might want to delete all .rar files or move all .odt files to another directory. Thankfully, you can use a series of wildcards to accomplish such tasks.

- `*` matches any number of characters. For example, `*.rar` matches any file with the ending .rar.
- **?** matches any single character. For example, `?.rar` matches a.rar but not ab.rar.

- **[*characters*]** matches any of the characters within the brackets. For example, `[ab].rar` matches a.rar and b.rar but not c.rar.
- ***[!*characters*]** matches any characters that are not listed. For example, `*[!ab].rar` matches c.rar but not a.rar or b.rar.

Executing Multiple Commands

Often you may want to execute several commands together, either by running one after another or by passing output from one to another.

Running Sequentially

If you need to execute multiple commands in sequence but don't need to pass output between them, there are two options based on whether or not you want the subsequent commands to run only if the previous commands succeed or not. If you want the commands to run one after the other regardless of whether or not preceding commands succeed, place a `;` between the commands. For example, if you want to get information about your hardware, you could run `lspci ; lsusb`, which would output information on your PCI buses and USB devices in sequence.

However, if you need to conditionally run the commands based on whether the previous command has succeeded, insert `&&` between commands. An example of this is building a program from source, which is traditionally done with `./configure`, `make`, and `make install`. The commands `make` and `make install` require that the previous commands have completed successfully, so you would use `./configure && make && make install`.

Using Byobu to Manage Your Terminal

One of the challenges of using the terminal is the difficulty of managing multiple screens. If you are in a desktop environment, you can launch another terminal window or use GNOME terminal's tabs, but if you are on a server or another machine that doesn't have a desktop environment installed, that doesn't work.

Thankfully, such a tool to help you does exist: byobu. Japanese for *screen*, byobu is a set of default configurations for the GNU screen command. Essentially, screen is a window manager for the command line. To install byobu, type `sudo apt-get install byobu`.

After it is started, you will notice you are back at a terminal prompt, but with a few differences. At the bottom are two lines of information. From the left to the right, the bottom line shows you the version of Ubuntu you are currently running, number of packages to update (if there are none, this won't appear), how long the system has been running, the system load, the CPU speed, the current memory usage, and the current date and time. The upper bar shows the list of open windows, the current logged in user, the system name, and the menu option (Figure 7-4).

You can now use your terminal exactly as you normally would, just with a few added pieces of information. Advanced use of byobu (and screen) is a topic too large for this chapter, but following are a few commands to get you started:

F2: Open new terminal window.

F3/F4: Move backward/forward through the list of windows.

F6: Detach from current byobu session. To reattach, use byobu -x.

F7: Scroll back through the output. Hit Esc to exit this mode and return to the command prompt.

F8: Set the window title to a custom title.

F9: Launch the menu.

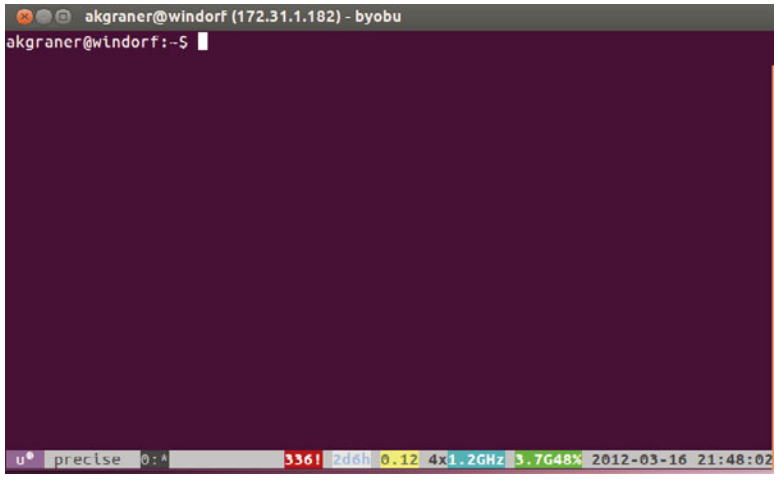

Figure 7-4 Byobu in GNOME terminal

Using Byobu by Default in GNOME Terminal

To have GNOME terminal launch byobu by default when it starts, you need to edit the preferences, which can be found at Edit > Profile Preferences under the Title and Command tab. Tick the box Run a Custom Command Instead of My Shell, and enter byobu in the line below. Now when you launch GNOME terminal, byobu will launch with it, and closing byobu will close GNOME terminal as well.

Moving to More Advanced Uses of the Command Line

There are a great number of good books out there for working the command line. In addition, because most of the command line has not changed in many years, a large body of information is available on the Internet. If you need help with something, often simply searching for the command will turn up what you need.

As you can imagine, there are hundreds and hundreds of different commands available on the system, and we don't have the space to cover them here. A number of superb Web sites and books can help you find out about the many different commands.

To get you started, here are some recommendations.

- ***The Official Ubuntu Server Book*** by Kyle Rankin and Benjamin Mako Hill (Prentice Hall, 2009) is an excellent resource for learning all things server related, including the effective use of the command line to accomplish administration tasks.
- ***A Practical Guide to Linux® Commands, Editors, and Shell Programming, Second Edition***, by Mark G. Sobell (Prentice Hall, 2009) is a good book for any user of the shell in Linux to have on his or her bookshelf.
- **LinuxCommand.org**, found at http://linuxcommand.org, is an excellent Web site designed to help people new to using the command line.
- **The Linux Documentation Project**, found at http://www.tdlp.org, is another excellent and free resource.