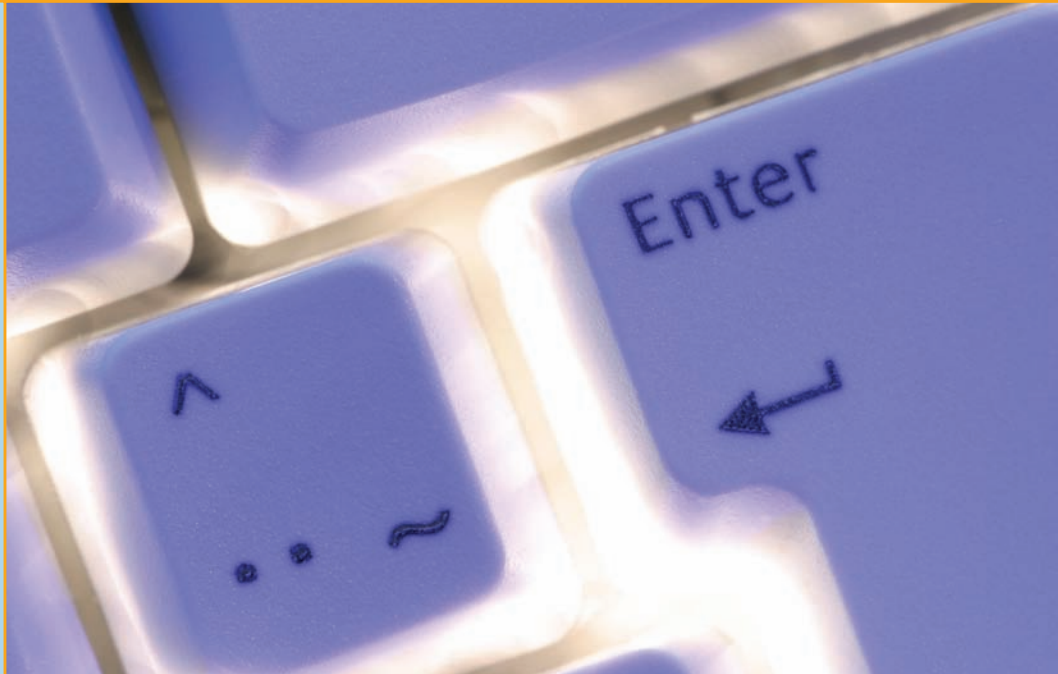


chapter  
**18**

# Working with the Command-Line Interface

*“You wanted to know who I am, Zero Cool? Well, let me explain the New World Order. Governments and corporations need people like you and me. We are Samurai...the Keyboard Cowboys...and all those other people who have no idea what’s going on are the cattle.... Moooo.”*

—THE PLAGUE, HACKERS (1995)



## In this chapter, you will learn how to

- Explain the operation of the command-line interface
- Execute fundamental commands from the command line
- Manipulate files from the command line

Whenever I teach a class of new techs and we get to the section on working with the command line, I’m invariably met with a chorus of moans and a barrage of questions and statements. “Why do we need to learn this old stuff?” “We’re running Windows 7, not Windows 3.1!” “Is this ritualistic hazing appropriate in an IT class?”

For techs who master the interface, the command line provides a powerful, quick, and elegant tool for working on a PC. Learning that interface and understanding how to make it work is not only useful, but also necessary for all techs who want to go beyond baby-tech status. You simply cannot work on all PCs without knowing the command line! I’m not the only one who thinks this way. The CompTIA A+ 220-802 certification exam tests you on a variety of command-line commands for doing everything from renaming a file to rebuilding a system file.

If you're interested in moving beyond Windows and into other operating systems such as Linux, you'll find that pretty much all of the serious work is done at a command prompt. Even the Apple Mac operating system (OS), for years a purely graphical operating system, now supports a command prompt. Why is the command prompt so popular? Well, for three reasons: First, if you know what you're doing, you can do most jobs more quickly by typing a text command than by clicking through a graphical user interface (GUI). Second, a command-line interface doesn't take much operating system firepower, so it's the natural choice for jobs where you don't need or don't want (or can't get to, in the case of Linux) a full-blown GUI for your OS. Third, text commands take very little bandwidth when sent across the network to another system.

So, are you sold on the idea of the command prompt? Good! This chapter gives you a tour of the Windows command-line interface, explaining how it works and what's happening behind the scenes. You'll learn the concepts and master essential commands, and then you'll work with files and folders throughout your drives. The chapter wraps up with a brief section on encryption and file compression in the "Beyond A+" section. A good tactic for absorbing the material in this chapter is to try out each command or bit of information as it is presented. If you have some experience working with a command prompt, many of these commands should be familiar to you. If the command line is completely new to you, please take the red pill and join me as we step into the matrix.

## Historical/Conceptual

Operating systems existed long before PCs were invented. Ancient, massive computers called *mainframes* and *minicomputers* employed sophisticated operating systems. It wasn't until the late 1970s that IBM went looking for an OS for a new *microcomputer*—the official name for the PC—the company was developing, called the IBM Personal Computer. After being rebuffed by a company called Digital Research, IBM went to a tiny company that had written a popular new version of the programming language called BASIC. IBM asked the company president if he could create an OS for the IBM PC. Although his company had never actually written an OS, he brazenly said "Sure!" That man was Bill Gates, and the tiny company was Microsoft.

After shaking hands with IBM representatives, Bill Gates hurriedly began to search for an OS based on the Intel 8086 processor. He found a primitive OS called *Quick-and-Dirty Operating System (QDOS)*, which was written by a one-man shop, and he purchased it for a few thousand dollars. After several minor changes, Microsoft released it as MS-DOS (Microsoft Disk Operating System) version 1.1. Although primitive by today's standards, MS-DOS 1.1 could provide all of the functions an OS needed. Over the years, MS-DOS went through version after version until the last Microsoft version, MS-DOS 6.22, was released in 1994. Microsoft licensed MS-DOS to PC makers so they could add their own changes and then rename the program. IBM called its version PC-DOS.

DOS used a **command-line interface**. You typed a command at a prompt, and DOS responded to that command. When Microsoft introduced Windows 95 and Windows NT, many computer users and techs thought that the command-line interface would go away, but techs not only continued to use the command line, they also *needed it* to troubleshoot and fix problems. With Windows 2000, it seemed once again that the command line would die, but again, that just didn't turn out to be the case.

Finally recognizing the importance of the command-line interface, Microsoft beefed it up in Windows XP and then again in Windows Vista. (The command-line interface in Windows 7 is essentially the same as in Windows Vista, but Microsoft added a whole new interface called PowerShell. See the "Beyond A+" section of this chapter for more details.) The command line in Windows XP and in Vista/7 offers commands and options for those commands that go well beyond anything seen in previous Microsoft operating systems. This chapter starts with some essential concepts of the command line and then turns to more specific commands.

## 802

### ■ Deciphering the Command-Line Interface

So how does a command-line interface work? It's a little like having an Instant Message conversation with your computer. The computer tells you it's ready to receive commands by displaying a specific set of characters called a **prompt**.

```
Computer: Want to play a game?  
Mike: _
```

You type a command and press ENTER to send it.

```
Mike: What kind of game?  
Computer: _
```

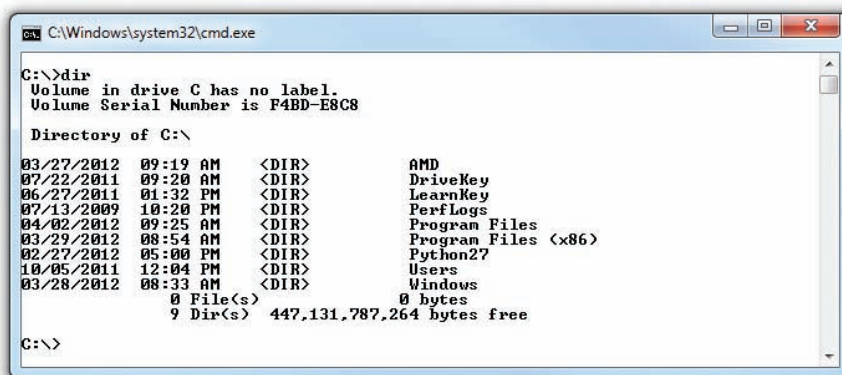
The PC goes off and executes the command, and when it's finished, it displays a new prompt, often along with some information about what it did.

```
Computer: A very fun game...  
Mike: _
```

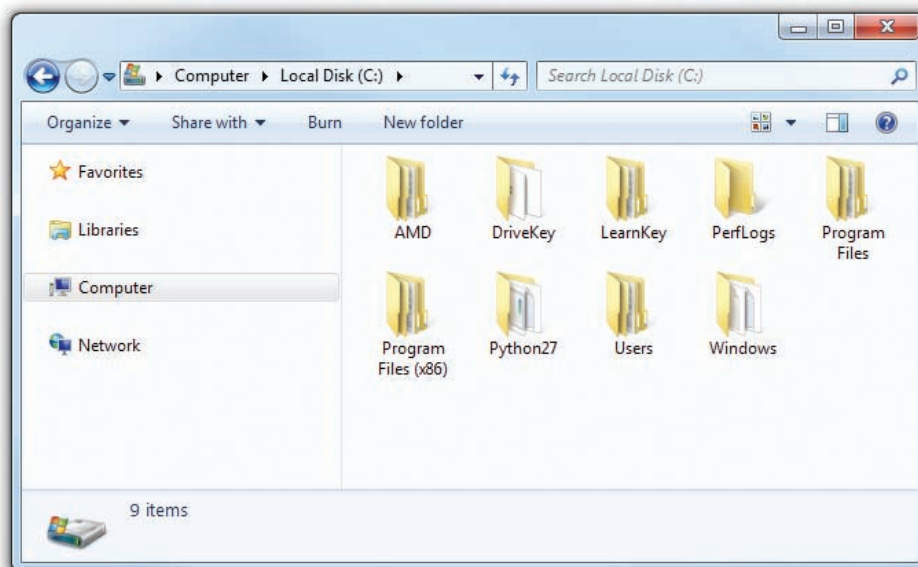
Once you get a new prompt, it means the computer is ready for your next instruction. You can give the computer commands in the GUI of Windows as well, just in a different way, by clicking buttons and menu options with your mouse instead of typing on the keyboard. The results are basically the same: you tell the computer to do something and it responds.

When you type in a command from the command line, you cause the computer to respond. As an example, suppose you want to find out the contents of a particular folder. From the command line, you'd type a command (in this case **dir**, but more on that in a minute), and the computer would respond by displaying a screen like the one in Figure 18.1.

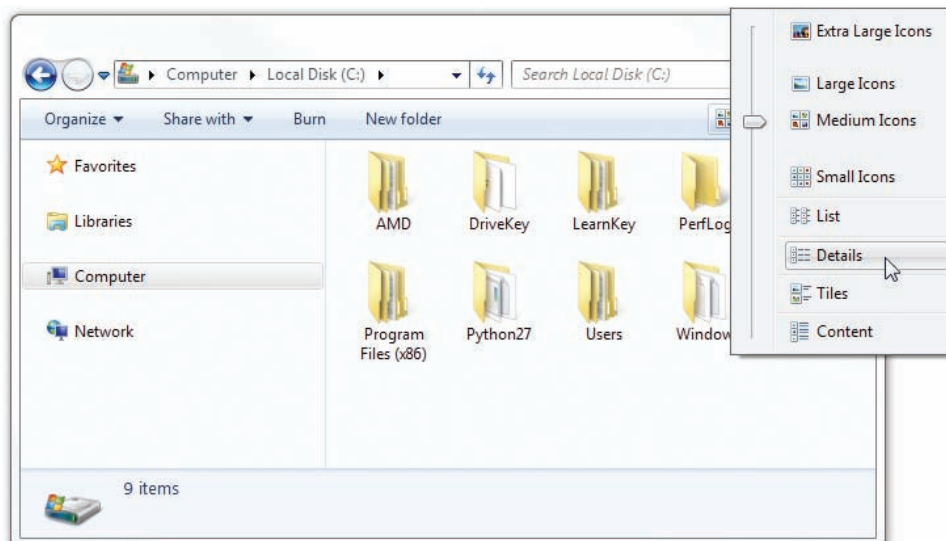
In the Windows GUI, you would open My Computer or Computer and click the C: drive icon to see the contents of that directory. The results might look like Figure 18.2, which at first glance isn't much like the command-line screen; however, simply by choosing a different view (see Figure 18.3), you can make the results look quite a bit like the command-line version, albeit much prettier (see Figure 18.4). The point here is that whichever interface you use, the information available to you is essentially the same.



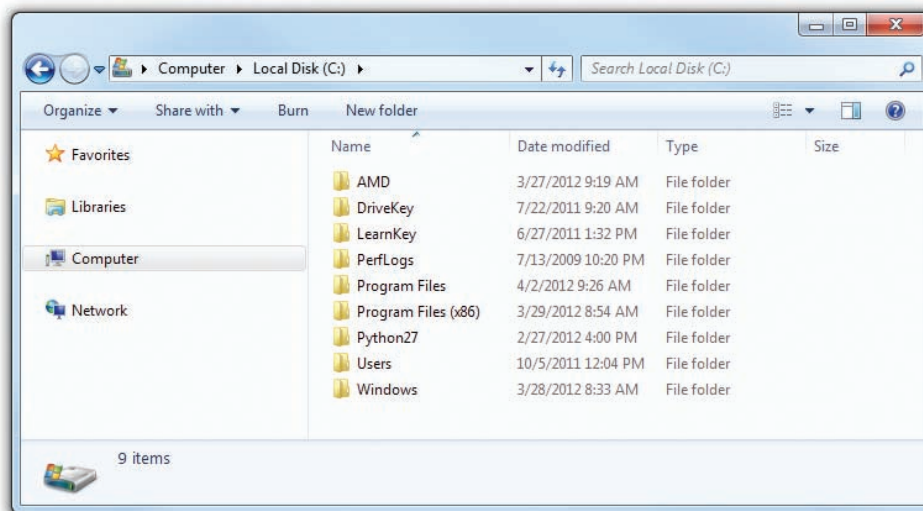
• **Figure 18.1** Contents of C: directory from the command line



• **Figure 18.2** Contents of C: in Computer—Icon view



• **Figure 18.3** Selecting Details view in Computer



• **Figure 18.4** Contents of C: in Computer—Details view



### Try This!

#### Opening GUI Programs from the Command Prompt

Keep in mind as you go through this chapter that the command line is just another tool for communicating with the operating system. Windows responds whether you're clicking or typing and sometimes does both, so try this! At a command prompt: type **notepad** and press the ENTER key. What happens?

The graphical program Notepad opens up, just as if you'd double-clicked its icon!

Here's another: type **explorer** and press ENTER. Voilà! Internet Explorer loads. Windows just responds.



## Accessing the Command Line

Before you can use the command-line interface, you have to open it. You can use various methods to do this, depending on the flavor of Windows you are using. Some methods are simpler than others; just make sure that you know at least one, or you'll never get off the starting line!

One easy way to access the command-line interface in Windows XP is by using the **Run dialog box**. Click the Start button, and then select Run. Type **cmd** or **command** and press the `ENTER` key (see Figure 18.5). If you are using Windows Vista or Windows 7, you access the command-line interface through the Start menu Search bar with the same two commands. A window pops up on your screen with a black background and white text—this is the command-line interface. Alternatively, buried in the Start menu of *most* computers, under Programs/All Programs | Accessories, is a link to the command-line interface; it's called *Command Prompt*. These links, just like the Run dialog box, pull up a nice command line–interface window (see Figure 18.6, which shows the Windows XP version). If you are displaying the command-line interface in Windows Vista or Windows 7, notice that you'll see a newer version number and copyright date. Also notice that the default user profile folder in the later editions is `C:\Users\User name`, as shown in Figure 18.7, rather than `C:\Documents and Settings\User name` as in Windows XP. To close the command line–interface window, you can either click the Close box in the upper-right corner, as on any other window, or simply type **exit** at any command prompt and press `ENTER`.

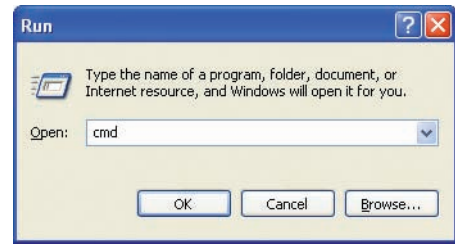
If you attempt to enter at the Windows Vista/7 command prompt a command that requires elevated or administrative privileges, you receive a UAC “Windows needs your permission to continue” dialog box (you learned about UAC in Chapter 16). You can also “manually” run a command with elevated privileges by right-clicking a command-prompt shortcut and then selecting *Run as administrator*. If you are prompted for administrator password or credentials, enter them as needed.



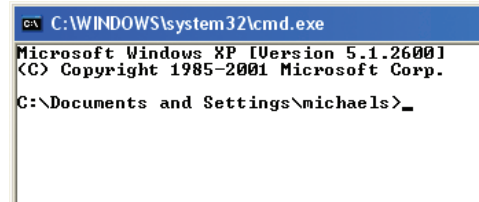
### Try This!

#### Accessing the Command Line

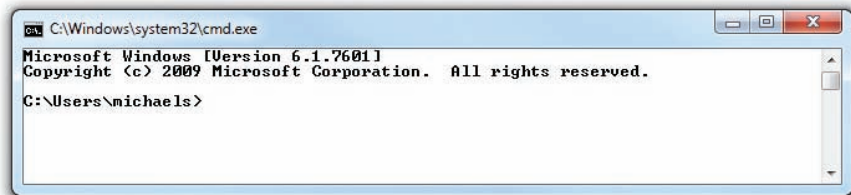
This chapter will be much more fun if you follow along with your own command line, so Try This! Using one of the methods outlined in this section, access a command prompt in Windows. Just remember that everything you do at the prompt can affect the functioning of your PC. So don't delete stuff if you don't know what it's for.



- **Figure 18.5** Type **cmd** in the Run dialog box to open a command line–interface window in Windows XP.



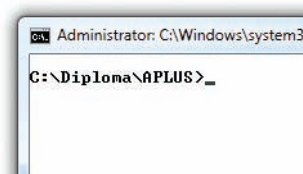
- **Figure 18.6** The command line–interface window with a `C:\` prompt



- **Figure 18.7** The Windows 7 command line–interface window



You can also create an administrator shortcut to the Windows Vista/7 command prompt by right-clicking on the desktop and selecting *New | Shortcut*. Then for the location of the item, type **cmd** and click *Next*. Type **cmd** to name the shortcut, and click *Finish*. Your shortcut appears on the desktop. Next, right-click the shortcut and select the *Advanced* button. In the *Advanced Properties* dialog box, check the *Run as administrator* box and click *OK*. You have now created a Windows Vista/7 command-prompt shortcut that will always run with administrative privileges.



• **Figure 18.8** Command prompt indicating focus on the C:\Diploma\APLUS\ folder



You can hold down the F5 or F8 key during boot-up to access the Windows Advanced Boot Options menu. This has an option to boot to Safe Mode with Command Prompt, which loads the GUI into Safe Mode and then overlays that with a command-line interface for rapid access to a prompt. This saves you the step of going to Start | Run and typing cmd.

## The Command Prompt

The command prompt is always *focused* on a specific folder. This is important because any commands you issue are performed *on the files in the folder* on which the prompt is focused. For example, if you see a prompt that looks like the following line, you know that the focus is on the root directory of the C: drive:

```
C:\>
```

If you see a prompt that looks like Figure 18.8, you know that the focus is on the C:\Diploma\APLUS\ folder of the C: drive. The trick to using a command line is first to focus the prompt on the drive and folder where you want to work.

## Filenames and File Formats

Windows manifests each program and piece of data as an individual file. Each file has a name, which is stored with the file on the drive. Windows inherits the idea of files from older operating systems—namely DOS—so a quick review of the old-style DOS filenames helps in understanding how Windows filenames work. Names are broken down into two parts: the filename and the extension. In true DOS, the filename could be no longer than eight characters, so you'll often see oddly named files on older systems. The extension, which is optional, could be up to three characters long in true DOS, and most computer programs and users continue to honor that old limit, even though it does not apply to modern PCs. No spaces or other illegal characters (/ \ [ ] | ÷ + = ; , \* ?) could be used in the filename or extension. The filename and extension are separated by a period, or *dot*. This naming system was known as the **8.3 (eight-dot-three) naming system**.

Here are some examples of acceptable true DOS filenames:

```
fred.exe          system.ini        file1.doc  
driver3.sys       janet             code33.h
```

Here are some unacceptable true DOS filenames:

```
4charext.exec    waytoolong.fil   bad÷char.bat     .no
```

I mention the true DOS limitations for a simple reason: *backward compatibility*. Starting with Windows 9x, Windows versions did not suffer from the 8.3 filename limitation. Instead they supported filenames of up to 255 characters (but still with the three-character extension) by using a trick called long filenames (LFN). Windows systems using LFN retained complete backward compatibility by automatically creating two names for every file, an 8.3 filename and a long filename. Modern Windows systems similarly support backward compatibility in file naming.

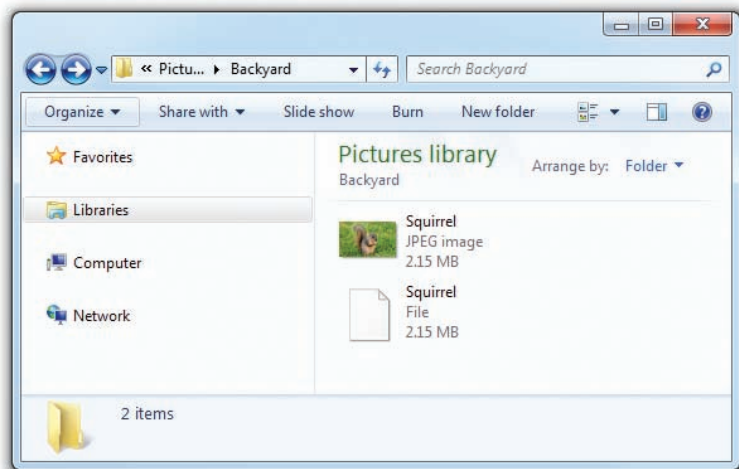
Whether you're running an ancient DOS system or the latest edition of Windows 7, the extension is very important, because the extension part of the filename tells the computer the type or function of the file. Program files use the extension .exe (for executable) or .com (for command). Anything that is not a program is some form of data to support a program. Different programs use different types of data files. The extension usually indicates which program uses that particular data file. For example, Microsoft Word

uses the extension .docx (.doc for Microsoft Office versions before 2007), while WordPerfect uses .wpd and PowerPoint uses .pptx (.ppt for Microsoft Office PowerPoint versions prior to 2007). Graphics file extensions, in contrast, often reflect the graphics standard used to render the image, such as .gif for CompuServe's Graphics Interchange Format or .jpg for the JPEG (Joint Photographic Experts Group) format.

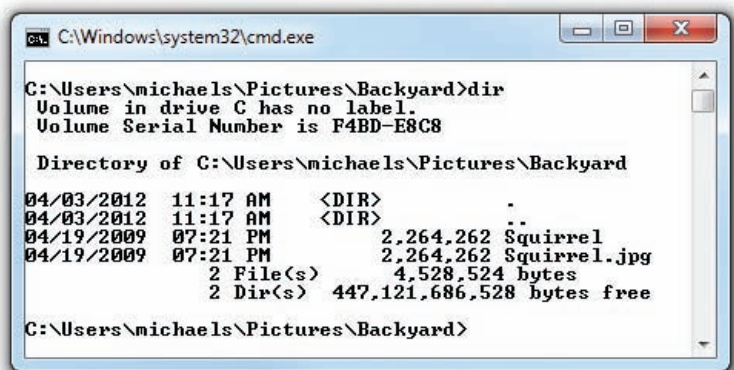
Changing the extension of a data file does not affect its contents, but without the proper extension, Windows won't know which program uses it. You can see this clearly in My Computer. Figure 18.9 shows a folder with two identical image files. The one on top shows a thumbnail because Windows recognizes this as a JPEG image; the one on the bottom shows a generic icon because I deleted the extension. Windows' GUI doesn't show file extensions by default. Figure 18.10 shows the contents of that same folder from the command line.

All files are stored on the hard drive in binary format, but every program has its own way of reading and writing this binary data. Each unique method of binary organization is called a file *format*. One program cannot read another program's files unless it can convert the other program's format into its format. In the early days of DOS, no programs were capable of performing this type of conversion, yet people wanted to exchange files. They wanted some type of common format that any program could read. The answer was a special format called **American Standard Code for Information Interchange (ASCII)**.

The ASCII standard defines 256 eight-bit characters. These characters include all of the letters of the alphabet (uppercase and lowercase), numbers, punctuation, many foreign characters (such as accented letters for French and Spanish—é, ñ, ô—and other typical non-English characters), box-drawing characters, and a series of special characters for commands such as a carriage return, bell, and end of file (see Figure 18.11). ASCII files, more commonly known as *text files*, store all data in ASCII format. The ASCII standard, however, is for more than just files. For example, when you press a key, the keyboard



• **Figure 18.9** What kind of file is the one on the lower right?



• **Figure 18.10** One file has no extension.

|     |       |     |    |     |   |     |   |     |   |     |   |     |   |     |   |
|-----|-------|-----|----|-----|---|-----|---|-----|---|-----|---|-----|---|-----|---|
| 000 | <nul> | 032 | sp | 064 | @ | 096 | ` | 128 | Ç | 160 | á | 192 | L | 224 | α |
| 001 | @     | 033 | !  | 065 | A | 097 | a | 129 | ü | 161 | í | 193 | l | 225 | β |
| 002 | (     | 034 | "  | 066 | B | 098 | b | 130 | é | 162 | ó | 194 | l | 226 | Γ |
| 003 | >     | 035 | #  | 067 | C | 099 | c | 131 | à | 163 | ú | 195 | l | 227 | Σ |
| 004 | ^     | 036 | \$ | 068 | D | 100 | d | 132 | â | 164 | ñ | 196 | l | 228 | Π |
| 005 | *     | 037 | %  | 069 | E | 101 | e | 133 | ã | 165 | ñ | 197 | l | 229 | Σ |
| 006 | *     | 038 | &  | 070 | F | 102 | f | 134 | ä | 166 | ê | 198 | l | 230 | μ |
| 007 | <     | 039 | '  | 071 | G | 103 | g | 135 | å | 167 | ë | 199 | l | 231 | θ |
| 008 | [     | 040 | <  | 072 | H | 104 | h | 136 | æ | 168 | ì | 200 | l | 232 | ε |
| 009 | <     | 041 | >  | 073 | I | 105 | i | 137 | è | 169 | í | 201 | l | 233 | θ |
| 010 | <     | 042 | *  | 074 | J | 106 | j | 138 | é | 170 | î | 202 | l | 234 | θ |
| 011 | ~     | 043 | +  | 075 | K | 107 | k | 139 | í | 171 | ï | 203 | l | 235 | ó |
| 012 | ~     | 044 | ,  | 076 | L | 108 | l | 140 | ì | 172 | ï | 204 | l | 236 | ω |
| 013 | ~     | 045 | -  | 077 | M | 109 | m | 141 | í | 173 | ï | 205 | l | 237 | ω |
| 014 | ~     | 046 | .  | 078 | N | 110 | n | 142 | î | 174 | « | 206 | l | 238 | ε |
| 015 | *     | 047 | /  | 079 | O | 111 | o | 143 | ë | 175 | » | 207 | l | 239 | ε |
| 016 | ~     | 048 | 0  | 080 | P | 112 | p | 144 | è | 176 | » | 208 | l | 240 | ε |
| 017 | ~     | 049 | 1  | 081 | Q | 113 | q | 145 | æ | 177 | » | 209 | l | 241 | ± |
| 018 | ~     | 050 | 2  | 082 | R | 114 | r | 146 | æ | 178 | » | 210 | l | 242 | ± |
| 019 | ~     | 051 | 3  | 083 | S | 115 | s | 147 | ø | 179 | » | 211 | l | 243 | ± |
| 020 | ~     | 052 | 4  | 084 | T | 116 | t | 148 | ø | 180 | » | 212 | l | 244 | ± |
| 021 | ~     | 053 | 5  | 085 | U | 117 | u | 149 | ø | 181 | » | 213 | l | 245 | ± |
| 022 | ~     | 054 | 6  | 086 | V | 118 | v | 150 | ø | 182 | » | 214 | l | 246 | ± |
| 023 | ~     | 055 | 7  | 087 | W | 119 | w | 151 | ø | 183 | » | 215 | l | 247 | ± |
| 024 | ~     | 056 | 8  | 088 | X | 120 | x | 152 | ø | 184 | » | 216 | l | 248 | ± |
| 025 | ~     | 057 | 9  | 089 | Y | 121 | y | 153 | ø | 185 | » | 217 | l | 249 | ± |
| 026 | ~     | 058 | :  | 090 | Z | 122 | z | 154 | ø | 186 | » | 218 | l | 250 | ± |
| 027 | ~     | 059 | ;  | 091 | [ | 123 | [ | 155 | ø | 187 | » | 219 | l | 251 | ± |
| 028 | ~     | 060 | <  | 092 | \ | 124 | \ | 156 | ø | 188 | » | 220 | l | 252 | ± |
| 029 | ~     | 061 | >  | 093 | ^ | 125 | > | 157 | ø | 189 | » | 221 | l | 253 | ± |
| 030 | ~     | 062 | ?  | 094 | ^ | 126 | ~ | 158 | ø | 190 | » | 222 | l | 254 | ± |
| 031 | ~     | 063 | ?  | 095 | - | 127 | ~ | 159 | ø | 191 | » | 223 | l | 255 | ± |

• **Figure 18.11** ASCII characters



sends the letter of that key to the PC in ASCII code. Even the monitor outputs in ASCII when you are running DOS.

ASCII was the first universal file format. Virtually every type of program—word processors, spreadsheets, databases, presentation programs—can read and write text files. However, text files have severe limitations. A text file can't store important information such as shapes, colors, margins, or text attributes (bold, underline, font, and so on). Therefore, even though text files are fairly universal, they are also limited to the 256 ASCII characters.

Even in the most basic text, you need to perform a number of actions beyond just printing simple characters. For example, how does the program reading the text file know when to start a new line? This is where the first 32 ASCII characters come into play. These first 32 characters are special commands (actually, some of them are

both commands and characters). For example, the ASCII value 7 can be either a large dot or a command to play a note (bell) on the PC speaker. ASCII value 9 is a Tab. ASCII value 27 is an Escape.

ASCII worked well for years, but as computers became used worldwide, the industry began to run into a problem: there are a lot more than 256 characters used all over the world! Nobody could use Arabic, Greek, Hebrew, or even Braille. In 1991, the Unicode Consortium, an international standards group, introduced Unicode. Basic **Unicode** is a 16-bit code that covers every character for the most common languages, plus a few thousand symbols. With Unicode you can make just about any

character or symbol you might imagine—plus a few thousand more you'd never even think of. The first 256 Unicode characters are the same as ASCII characters, making for easy backward compatibility.

## Drives and Folders

When working from the command line, you need to be able to focus the prompt at the specific drive and folder that contains the files or programs with which you want to work. This can be a little more complicated than it seems.

At boot, Windows assigns a drive letter (or name) to each hard drive partition and to each floppy or other disk drive. The first floppy drive, if installed, is called A:, and the second, if installed, is called B:. Hard drives usually start with the letter C: and can continue to Z: if necessary. Optical drives by default get the next available drive letter after the last hard drive. Windows enables you to change the default lettering for drives, so you're



### Try This!

#### Make Some Unicode!

A lot of e-mail programs can use Unicode characters, as can Internet message boards such as my Tech Forums. You can use Unicode characters to accent your writing or simply to spell a person's name correctly—Martin *Acuña*—when you address him. Working with Unicode is fun, so Try This!

1. Open a text editing program such as Notepad in the Windows GUI.
2. Hold down the ALT key on your keyboard and, referring to Figure 18.11, press numbers on your keyboard's number pad to enter special characters. For example, pressing ALT-164 should display an ñ, whereas ALT-168 shows a ç.
3. If you have access to the Internet, surf over to the Tech Forums ([www.totalsem.com/techforum/index.php](http://www.totalsem.com/techforum/index.php)) and say howdy. Include some Unicode in your post, of course!

likely to see all sorts of lettering schemes. On top of that, you can mount a hard drive as a volume in another drive, as you'll recall from Chapter 12.

Whatever the names of the drives, Windows uses a hierarchical directory tree to organize the contents of these drives. All files are put into groups Windows calls *folders*, although you'll often hear techs use the term *directory* rather than *folder*, a holdover from the true DOS days. Any file not in a folder *within* the tree—that is, any file in the folder at the root of the directory tree—is said to be in the **root directory**. A folder inside another folder is called a *subfolder*. Any folder can have multiple subfolders. Two or more files with the same name can exist in different folders on a PC, but two files in the same folder cannot have the same name. In the same way, no two subfolders under the same folder can have the same name, but two subfolders under different folders can have the same name.

When describing a drive, you use its letter and a colon. For example, the hard drive would be represented by C:. To describe the root directory, put a backslash (\) after the C:, as in C:\. To describe a particular directory, add the name of the directory. For example, if a PC has a directory in the root directory called Test, it is C:\Test. Subdirectories in a directory are displayed by adding backslashes and names. If the Test directory has a subdirectory called System, it is shown like this: C:\Test\System. This naming convention provides for a complete description of the location and name of any file. If the C:\Test\System directory includes a file called test2.txt, it is C:\Test\System\test2.txt.

The exact location of a file is called its **path**. The path for the test2.txt file is C:\Test\System. Here are some examples of possible paths:

```
C:\Program Files
C:\WINNT\system32\1025
F:\FRUSCH3\CLEAR
A:\Reports
D:\
```

Here are a few items to remember about folder names and filenames:

- Folders and files may have spaces in their names.
- The only disallowed characters are the following eleven: \* " / \ [ ] : ; | = ,
- Files aren't required to have extensions, but Windows won't know the file type without an extension.
- Folder names may have extensions—but they are not commonly used.

## ■ Mastering Fundamental Commands

It's time to try using the command line, but before you begin, a note of warning is in order: the command-line interface is picky and unforgiving. It will do what you *say*, not what you *mean*, so it always pays to double-check that those are one and the same before you press ENTER and commit the command. One careless keystroke can result in the loss of crucial data, with



### Tech Tip

#### Directory Trees

*It helps to visualize a directory tree as upside down, because in geekspeak, the trunk, or root directory, is described as "above" the folders that divide it, and those subfolders "below" root are spoken of as being "above" the other subfolders inside them. For example, "The file is in the Adobe folder under Program Files."*

no warning and no going back. In this section, you'll explore the structure of commands and then play with four commands built into all versions of Microsoft's command-line interface: `dir`, `cd`, `md`, and `rd`.

## Structure: Syntax and Switches

All commands in the Windows command-line interface use a similar structure and execute in the same way. You type the name of the command, followed by the target of that command and any modifications of that command that you want to apply. You can call up a modification by using an extra letter or number, called a **switch** or *option*, which may follow either the command or the target, depending on the command. The proper way to write a command is called its **syntax**. The key with commands is that you can't spell anything incorrectly or use a `\` when the syntax calls for a `/`. The command line is completely inflexible, so you have to learn the correct syntax for each command.

```
[command] [target (if any)] [switches]
```

or

```
[command] [switches] [target (if any)]
```

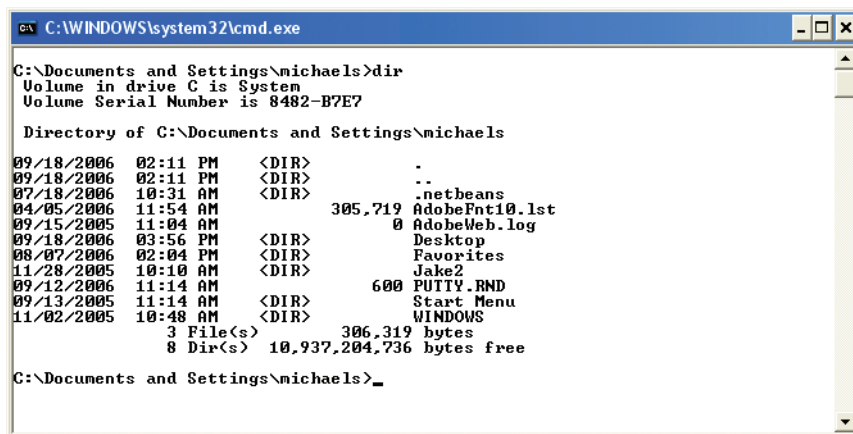
How do you know what switches are allowed? How do you know whether the switches come before or after the target? If you want to find out the syntax and switches used by a particular command, always type the command followed by `/?` to get help:

```
[command name] /?
```

## Viewing Directory Contents: The `dir` Command

The **dir command** shows you the contents of the directory where the prompt is focused. If you're like most techs, you'll use `dir` more often than any other

command at the command prompt. When you open a command-line window in Windows, it opens focused on your user folder. You will know this because the prompt in Windows XP will look like this: `C:\Documents and Settings\username>` and the prompt in Windows Vista/7 looks like `C:\Users\User name>`. By typing `dir` and then pressing the ENTER key (remember that you must always press ENTER to execute a command from the command line), you will see something like Figure 18.12 in Windows XP.



• **Figure 18.12** Results for `dir` in a user's folder

If you are following along on a PC, remember that different computers contain different files and programs, so you will absolutely see something different from what's shown in Figure 18.12! If a lot of text scrolls quickly down the screen, try typing `dir /p` (pause). Don't forget to press `ENTER`. The `dir /p` command is a lifesaver when you're looking for something in a large directory. Just press `SPACEBAR` to display the next screen.

When you type a simple `dir` command, you will see that some of the entries look like this:

```
09/04/2012 05:51 PM 63,664 bambi.jpg
```

All of these entries are files. The `dir` command lists the creation date, creation time, file size in bytes, filename, and extension. Any entries that look like this are folders:

```
12/31/2012 10:18 AM <DIR> WINDOWS
```

The `dir` command lists the creation date, creation time, `<DIR>` to tell you it is a folder, and the folder name. If you ever see a listing with `<JUNCTION>` instead of `<DIR>`, you're looking at a hard drive partition that's been mounted as a folder instead of a drive letter:

```
08/06/2012 02:28 PM <JUNCTION> Other Drive
```

Now type the `dir /w` command. Note that the `dir /w` command shows only the filenames, but they are arranged in five columns across your screen. Finally, type `dir /?` to see the screen shown in Figure 18.13, which lists all possible switches for the command.



### Tech Tip

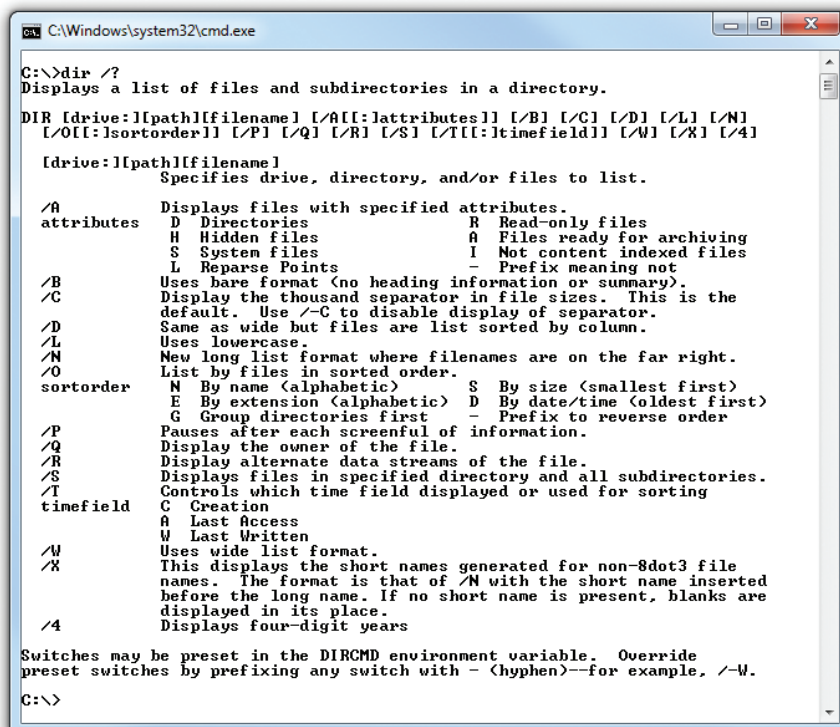
#### Do You Need Spaces?

*Some commands give you the same result whether you include spaces or not. Typing `dir/p` or `dir /p`, for example, provides the same output. Some commands, however, require spaces between the command and switches. In general, get into the habit of putting spaces between your command and switches and you won't run into problems.*



#### Extra text typed after

a command to modify its operation, such as `/w` or `/p` after `dir`, is called a *switch*. Almost all switches can be used simultaneously to modify a command. For example, try typing `dir /w /p`.



• **Figure 18.13** Typing `dir /?` lists all possible switches for the `dir` command.





## Tech Tip

### Errors Are Good!

Consider errors in general for a moment—not just command-prompt errors such as “Invalid directory,” but any error, including Windows errors. Many new computer users freeze in horror when they see an error message. Do not fear error messages. Error messages are good! Love them. Worship them. They will save you.

Seriously, think how confusing it would be if the computer didn't tell you when you messed up. Error messages tell you what you did wrong so you can fix it. You absolutely cannot hurt your PC in any way by typing the `dir` or `cd` command incorrectly. Take advantage of this knowledge and experiment. Intentionally make mistakes to familiarize yourself with the error messages. Have fun and learn from errors!

Typing any command followed by `/?` brings up a help screen for that particular command. Although these help screens can sometimes seem a little cryptic, they're useful when you're not familiar with a command or you can't figure out how to get a command to do what you need. Even though I have almost every command memorized, I still refer to these help screens; you should use them as well. If you're really lost, type **help** at the command prompt for a list of commands you may type. Once you find one, type **help** and then the name of the command. For example, if you type **help dir**, you'll see the screen shown in Figure 18.13.

## Changing Directory Focus: The `cd` Command

You can use the **cd (or chdir) command** to change the focus of the command prompt to a different directory. To use the `cd` command, type `cd\` followed by the name of the directory on which you want the prompt to focus. For example, to go to the `C:\Obiwan` directory, you type `cd\obiwan` and then press `ENTER`. If the system has an Obiwan directory, the prompt changes focus to that directory and appears as `C:\Obiwan>`. If no Obiwan directory exists or if you accidentally type something like `obiwam`, you get the error “The system cannot find the path specified.” If only I had a dollar for every time I've seen those errors! I usually get them because I've typed too fast. If you get this error, check what you typed and try again.

To return to the root directory, type `cd\` and press `ENTER`. You can use the `cd` command to point the prompt to any directory. For example, you could type `cd\fred\backup\test` from a `C:\` prompt, and the prompt would change to `C:\Fred\Backup\Test\>`—assuming, of course, that your system has a directory called `C:\Fred\Backup\Test`.

Once the prompt has changed, type `dir` again. You should see a different list of files and directories. Every directory holds different files and subdirectories, so when you point the prompt to different directories, the `dir` command shows you different contents.

The `cd` command allows you to use a space instead of a backslash, a convenient shortcut. For example, you could go to the `C:\Windows` directory from the root directory simply by typing `cd windows` at the `C:\` prompt. You can use the `cd [space]` command to move one level at a time, like this:

```
C:\>cd fred
C:\Fred>cd backup
C:\Fred\Backup>cd test
```

Or, you can jump multiple directory levels in one step, like this:

```
C:\>cd fred\backup\test
C:\Fred\Backup\Test>
```

A final trick: If you want to go *up* a single directory level, you can type `cd` followed immediately by two periods. So, for example, if you're in the `C:\Fred\Backup` directory and you want to move up to the `C:\Fred` directory, you can simply type `cd..` and you'll be there:

```
C:\Fred\Backup>cd..
C:\Fred>
```

Take some time to move the prompt focus around the directories of your PC, using the `cd` and `dir` commands. Use `dir` to find a directory, and then use `cd` to move the focus to that directory. Remember, `cd \` always gets you back to the root directory.

## Moving Between Drives

The `cd` command is *not* used to move between drives. To get the prompt to point to another drive (“point” is command-line geekspeak for “switch its focus”), just type the drive letter and a colon. If the prompt points at the `C:\Sierra` directory and you want to see what is on the USB thumb drive (`E:`), just type `e:` and the prompt will point to the USB drive. You’ll see the following on the screen:

```
C:\Sierra>e:  
E:\>
```

To return to the `C:` drive, just type `c:` and you’ll see the following:

```
E:\>c:  
C:\Sierra>
```

Note that you return to the same directory you left. Just for fun, try typing in a drive letter that you know doesn’t exist. For example, I know that my system doesn’t have a `W:` drive. If I type in a nonexistent drive on a Windows system, I get the following error:

```
The system cannot find the drive specified.
```

Try inserting a CD or DVD and use the `cd` command to point to its drive. Type `dir` to see the contents of the optical disc. Type `cd` to move the focus to any folders on the optical disc. Now return focus to the `C:` drive.

Using the `dir`, `cd`, and drive letter commands, you can access any folder on any storage device on your system. Make sure you can use these commands comfortably to navigate inside your computer.

## Making Directories: The `md` Command

Now that you have learned how to navigate in a command-prompt world, it’s time to start making stuff, beginning with a new directory.

To make a directory, use the **`md` (or `mkdir`) command**. To create a directory called `Steam` under the root directory `C:`, for example, first type `cd \` to ensure that you are in the root directory. You should see the prompt

```
C:\>
```

Now that the prompt points to the root directory, type `md Steam` to create the directory:

```
C:\>md Steam
```

Once you press `ENTER`, Windows executes the command, but it won’t volunteer any information about what it did. You must use the `dir` command to see that you have, in fact, created a new directory. Note that the `Steam` directory in this example is not listed last, as you might expect.

```

C:\>dir
Volume in Drive C is
Volume Serial Number is 1734-3234
Directory of C:\

09/01/2012  8:11 AM    <DIR>          ATI
06/10/2012  08:11 AM    <DIR>          Intel
10/04/2012  10:22 PM    <DIR>          Steam
09/11/2012  11:32 AM    <DIR>          NVIDIA
08/06/2012  02:28 PM    <JUNCTION>    Other Drive
09/14/2012  11:15 AM    <DIR>          PerfLogs
09/14/2012  11:11 AM    <DIR>          Program Files
09/14/2012  11:11 AM    <DIR>          Program Files (x86)
09/12/2012  08:32 PM                21 statusclient.log
07/31/2012  10:40 PM                153 systemscandata.txt
03/13/2012  09:54 AM                1,111,040 t3h0
04/21/2012  04:19 PM    <DIR>          temp
09/24/2012  12:11 AM    <DIR>          Users
07/12/2012  10:18 AM    <DIR>          Windows
                3 File(s)          1,111,214 bytes
                10 Dir(s)          2,294,182,881,834 bytes free

```

What about uppercase and lowercase? Windows supports both in file and folder names but rarely makes any distinction with commands. Use the `md` command to make a folder called `steam` (note the lowercase) and see what happens. This also happens in the graphical Windows. Go to your desktop and try to make two folders, one called `STEAM` and the other called `steam`, and see what Windows tells you.

To create a Files subdirectory in the Steam directory, first use the `cd\` command to point the prompt to the Steam directory:

```
cd\steam
```

Then run the `md` command to make the Files directory:

```
md Files
```

Make sure that the prompt points to the directory in which you want to make the new subdirectory before you execute the `md` command. When you're finished, type `dir` to see the new Files subdirectory. Just for fun, try the process again and add a Games directory under the Steam directory. Type `dir` to verify success.

## Removing Directories: The `rd` command

Removing subdirectories works exactly like making them. First, get to the directory that contains the subdirectory you want to delete, and then execute the **rd (or rmdir) command**. In this example, let's delete the Files subdirectory in the `C:\Steam` directory. First, get to where the Files directory is located—`C:\Steam`—by typing `cd\steam`. Then type `rd files`. If you received no response from Windows, you probably did it right! Type `dir` to check that the Files subdirectory is gone.

The plain `rd` command will not delete a directory in Windows if the directory contains files or subdirectories. If you want to delete a directory that contains files or subdirectories, you must first empty that directory by

using the `del` (for files) or `rd` (for subdirectories) command. You can use the `rd` command followed by the `/s` switch to delete a directory as well as all files and subdirectories. The `rd` command followed by the `/s` switch is handy but dangerous, because it's easy to delete more than you want. When deleting, always follow the maxim "Check twice and delete once."

Let's delete the Steam and Games directories with `rd` followed by the `/s` switch. Because the Steam directory is in the root directory, point to the root directory with `cd \`. Now execute the command `rd c:\steam /s`. In a rare display of mercy, Windows responds with the following:

```
C:\>rd steam /s
steam, Are you sure (Y/N)?
```

Press the `y` key and both `C:\Steam` and `C:\Steam\Games` are eliminated.



Make sure you know how to use `md`, `rd`, and `cd` for the CompTIA A+ 220-802 exam.



## Try This!

### Working with Directories

PC techs should be comfortable creating and deleting directories. To get some practice, Try This!

1. Create a new directory in the root directory by using the make directory command (`md`). Type `cd \` to return to the root directory. At the command prompt, make a directory called Jedi:

```
C:\>md Jedi
```

2. As usual, the prompt tells you nothing; it just presents a fresh prompt. Do a `dir` (that is, type the `dir` command) to see your new directory. Windows creates the new directory wherever it is pointing when you issue the command, whether or not that's where you meant to put it. To demonstrate, point the prompt to your new directory by using the `cd` command:

```
C:\>cd jedi
```

3. Now use the make directory command again to create a directory called Yoda:

```
C:\Jedi>md Yoda
```

Do a `dir` again, and you should see that your Jedi directory now contains a Yoda directory.

4. Type `cd \` to return to the root directory so you can delete your new directories by using the remove directory command (`rd`):

```
C:\>rd /s jedi
```

In another rare display of mercy, Windows responds with the following:

```
jedi, Are you sure <Y/N>?
```

5. Press `y` to eliminate both `C:\Jedi` and `C:\Jedi\Yoda`.



## Running a Program

To run a program from the command line, simply change the prompt focus to the folder where the program is located, type the name of the program, and then press the ENTER key on your keyboard. Try this safe example. Go to the C:\Windows\System32 folder—the exact name of this folder varies by system. Type `dir /p` to see the files one page at a time. You should see a file called `mem.exe` if you're running a 32-bit version of Windows (see Figure 18.14).

As mentioned earlier, all files with extensions `.exe` and `.com` are programs, so `mem.exe` is a program.

To run the `mem.exe` program, just type the filename, in this case `mem`, and press ENTER (see Figure 18.15). Note that you do not have to type the `.exe` extension, although you can. Congratulations! You have just run your first program from the command line.

## Working with Files

This section deals with basic file manipulation. You will learn how to look at, copy, move, rename, and delete files. The examples in this section are based on a C: root directory with the following files and directories:

```
Administrator: C:\Windows\system32\cmd.exe - dir /p
04/10/2009 11:27 PM          950,272 mblctr.exe
01/19/2008 12:33 AM        275,968 mcbuilder.exe
11/02/2006 07:35 AM          73,376 mciaivi.drv
11/02/2006 04:46 AM        82,944 mciaivi32.dll
11/02/2006 04:46 AM        38,912 mcicda.dll
11/02/2006 04:46 AM        36,352 mcigtz32.dll
11/02/2006 04:46 AM        23,552 mciseq.dll
11/02/2006 07:35 AM        25,264 mciseq.drv
11/02/2006 04:46 AM        23,040 mcivave.dll
11/02/2006 07:35 AM        28,160 mcivave.drv
04/10/2009 11:32 PM       438,744 mcupdate_GenuineIntel.dll
01/19/2008 12:34 AM       129,024 McxDrv.dll
03/16/2007 11:54 PM          6,656 md5c.exe
04/09/2007 01:23 PM        28,040 mdmon.dll
01/19/2008 12:34 AM       205,312 mdminst.dll
11/02/2006 04:45 AM          89,064 MdRes.exe
01/19/2008 12:33 AM       128,512 MdSched.exe
04/10/2009 11:28 PM       356,864 MediaMetadataHandler.dll
11/02/2006 02:09 AM          39,274 mem.exe
04/10/2009 11:28 PM     2,868,224 mf.dll
11/02/2006 04:46 AM        41,984 mf3216.dll
11/02/2006 04:46 AM       924,944 mfc40.dll
11/02/2006 04:46 AM       924,944 mfc40u.dll
04/10/2009 11:28 PM     1,135,104 mfc42.dll
06/17/1998 06:08 PM          53,248 MFC42EMU.DLL
04/10/2009 11:28 PM     1,160,704 mfc42u.dll
03/18/2003 09:20 PM          1,060,864 mfc71.dll
03/18/2003 09:12 PM          1,047,552 mfc71u.dll
Press any key to continue . . .
```

• **Figure 18.14** The `mem.exe` program displayed in the System32 folder

```
Administrator: C:\Windows\system32\cmd.exe
C:\Windows\System32>mem
655360 bytes total conventional memory
655360 bytes available to MS-DOS
580912 largest executable program size

1048576 bytes total contiguous extended memory
0 bytes available contiguous extended memory
941056 bytes available XMS memory
MS-DOS resident in High Memory Area

C:\Windows\System32>
```

• **Figure 18.15** Running `mem` in Windows Vista 32-bit



Windows includes a lot of command-line tools for specific jobs such as starting and stopping services, viewing computers on a network, converting hard drive file systems, and more. This book discusses these task-specific tools in the chapters that reflect their task. Chapter 22 goes into detail on the versatile and powerful `net` command, for example. You'll read about the `convert` command in Chapter 29. I couldn't resist throwing in two of the more interesting tools, `compact` and `cipher`, in the "Beyond A+" section of this chapter.

```
C:\>dir
Volume in drive C has no label.
Volume Serial Number is 4C62-1572

Directory of C:\

05/26/2009 11:37 PM          0 AILog.txt
05/29/2009 05:33 PM         5,776 aoedoppl.txt
05/29/2009 05:33 PM         2,238 aoeWVlog.txt
07/12/2009 10:38 AM          <DIR>      books
07/15/2009 02:45 PM         1,708 CtDrvStp.log
07/12/2008 04:46 AM          <DIR>      Documents and Settings
06/04/2009 10:22 PM          <DIR>      Impressions Games
09/11/2008 11:32 AM          <DIR>      NVIDIA
08/06/2009 02:28 PM          <JUNCTION> Other Drive
01/03/2009 01:12 PM          <DIR>      pers-drv
09/14/2008 11:11 AM          <DIR>      Program Files
09/12/2009 08:32 PM           21 statusclient.log
07/31/2009 10:40 PM          153 systemscandata.txt
03/13/2009 09:54 AM       1,111,040 t3h0
04/21/2009 04:19 PM          <DIR>      temp
```

```

01/10/2008  07:07 PM    <DIR>          WebCam
12/31/2007  10:18 AM    <DIR>          WINDOWS
09/14/2008  12:48 PM    <DIR>          WINNT
01/03/2008  09:06 AM    <DIR>          WUTemp
              7 File(s)        1,120,936 bytes
              12 Dir(s)  94,630,002,688 bytes free

```

Because you probably don't have a PC with these files and directories, follow the examples but use what's on your drive. In other words, create your own folders and copy files to them from various folders currently on your system.

## Attributes

Remember way back in Chapter 4 when you had to make changes to the folder options in My Computer/Computer to see hidden and system files? You were actually seeing files with special attributes.

All files have four special values, or **attributes**, that determine how programs (such as My Computer in Windows XP or Computer in Windows Vista/7) treat the file in special situations. The first attribute is the hidden attribute. If a file is hidden, it is not displayed when you issue the `dir` command. Next is the read-only attribute. A file with a **read-only attribute** cannot be modified or deleted. Third is the system attribute, which is used only for Windows XP system files such as `ntldr` and `boot.ini`. In reality, it does nothing more than provide an easy identifier for these files. Fourth is the archive attribute, which is used by backup software to identify files that have been changed since their last backup.

Up to this point in the chapter, you've used commands wrapped up in a single executable file called `cmd.exe`. The `cmd.exe` program all by itself, in other words, enables you to access the command-line interface and use built-in or *internal* commands such as `dir`, `cd`, `md`, and so forth. Several common command-line commands, in contrast, use their own executable. We call these *external* commands.

The external command-line program **attrib.exe** enables you to inspect and change file attributes. To inspect a file's attributes, type the **attrib** command followed by the name of the file. To see the attributes of the file `AILog.txt`, for example, type **attrib ailog.txt**. The result is

```
A      AILog.txt
```

The letter *A* stands for archive, the only attribute of `AILog.txt`.

Go to the `C:\` directory and type **attrib** by itself. You'll see a result similar to the following:

```

C:\>attrib
A      C:\AILog.txt
A      C:\aoedoppl.txt
A      C:\aoeWVlog.txt
A H    C:\AUTOEXEC.BAT
A SH   C:\boot.ini
A H    C:\CONFIG.SYS
A      C:\CtDrvStp.log
A SH   C:\hiberfil.sys
A SHR  C:\IO.SYS
A SHR  C:\MSDOS.SYS

```

```
A SHR      C:\NTDETECT.COM
A SHR      C:\ntldr
A SH       C:\pagefile.sys
A          C:\statusclient.log
A          C:\systemscandata.txt
A          C:\t3h0
```

The letter *R* means read-only, *H* is hidden, and *S* is system. Hey! There are some new files there. That's right, some were hidden. Don't panic if you see a number of files different from those just listed. No two C:\ directories are ever the same. In most cases, you'll see many more files than just these. Notice that important files such as `ntldr` and `ntdetect.com` in Windows XP have the system, hidden, and read-only attributes set. Microsoft does this to protect them from accidental deletion.

You also use the `attrib` command to change a file's attributes. To add an attribute to a file, type the attribute letter preceded by a plus sign (+) as an option, and then type the filename. To delete an attribute, use a minus sign (-). For example, to add the read-only attribute to the file `AllLog.txt`, type this:

```
attrib +r ailog.txt
```

To remove the archive attribute, type this:

```
attrib -a ailog.txt
```

You can add or remove multiple attributes in one command. Here's an example of removing three attributes from the `ntdetect.com` file:

```
attrib -r -s -h ntdetect.com
```

You can also automatically apply `attrib` to matching files in subdirectories by using the `/s` switch at the end of the statement. For example, if you



## Try This!

### Working with Attributes

It's important for you to know that everything you do at the command line affects the same files at the GUI level, so Try This!

1. Go to My Computer or Computer and create a folder called Test in the root directory of your C: drive.
2. Copy a couple of files into that folder and then right-click one to see its properties.
3. Open a command-line window and navigate to the C:\Test folder. Type `dir` to see that the contents match what you see in My Computer/Computer.
4. From the command line, change the attributes of one or both files. Make one a hidden file, for example, and the other read-only.
5. Now go back to My Computer/Computer and access the properties of each file. Any changes?

have lots of files in your My Music folder that you want to hide, but they are neatly organized in many subdirectories, you could readily use `attrib` to change all of them with a simple command. Change directories from the prompt until you're at the My Music folder and then type the following:

```
attrib +h *.mp3 /s
```

When you press the `ENTER` key, all your music files in My Music and any My Music subdirectories will become hidden files.



You should know how to use `attrib` for the CompTIA A+ 220-802 exam.

## Wildcards

Visualize having 273 files in one directory. A few of these files have the extension `.docx`, but most do not. You are looking only for files with the `.docx` extension. Wouldn't it be nice to be able to type the `dir` command in such a way that only the `.docx` files come up? You can do this by using wildcards.

A **wildcard** is one of two special characters—asterisk (\*) and question mark (?)—that you can use in place of all or part of a filename, often so that a command-line command will act on more than one file at a time. Wildcards work with all command-line commands that take filenames. A great example is the `dir` command. When you execute a plain `dir` command, it finds and displays all of the files and folders in the specified directory; however, you can also narrow its search by adding a filename. For example, if you type the command `dir ailog.txt` while in your root (`C:\`) directory, you get the following result:

```
C:\>dir ailog.txt
Volume in drive C has no label.
Volume Serial Number is 4C62-1572
Directory of C:\
05/26/2009  11:37 PM                0 AILog.txt
              1 File(s)                0 bytes
              0 Dir(s)  94,630,195,200 bytes free
```

If you just want to confirm the presence of a particular file in a particular place, this is very convenient. But suppose you want to see all files with the extension `.txt`. In that case, you use the \* wildcard, like this: `dir *.txt`. A good way to think of the \* wildcard is "*I don't care.*" Replace the part of the filename that you don't care about with an asterisk (\*). The result of `dir *.txt` would look like this:

```
Volume in drive C has no label.
Volume Serial Number is 4C62-1572

Directory of C:\

05/26/2009  11:37 PM                0 AILog.txt
05/29/2009  05:33 PM            5,776 aoedoppl.txt
05/29/2009  05:33 PM            2,238 aoeWVlog.txt
07/31/2008  10:40 PM            153 systemsdata.txt
              4 File(s)                8,167 bytes
              0 Dir(s)  94,630,002,688 bytes free
```



Wildcards also substitute for parts of filenames. This dir command will find every file that starts with the letter *a*:

```
C:\>dir a*.*
Volume in drive C has no label.
Volume Serial Number is 4C62-1572

Directory of C:\

05/26/2009  11:37 PM                0 AILog.txt
05/29/2009  05:33 PM                5,776 aoedoppl.txt
05/29/2009  05:33 PM                2,238 aoeWVlog.txt
           3 File(s)                8,014 bytes
           0 Dir(s)  94,629,675,008 bytes free
```

We've used wildcards only with the dir command, but virtually every command that deals with files will take wildcards. Let's examine the ren and del commands and see how they use wildcards.

## Renaming Files

To rename files, you use the **ren (or rename) command**, which seems pretty straightforward. To rename the file img033.jpg to park.jpg, type the following and then press the ENTER key:

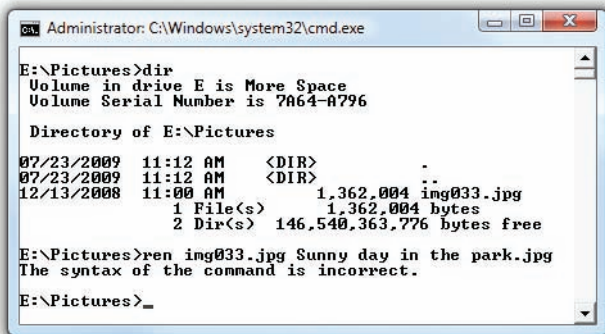
```
ren img033.jpg park.jpg
```

"That's great," you might be thinking, "but what about using a more complex and descriptive filename, such as Sunny day in the park.jpg?" Type what should work, like this:

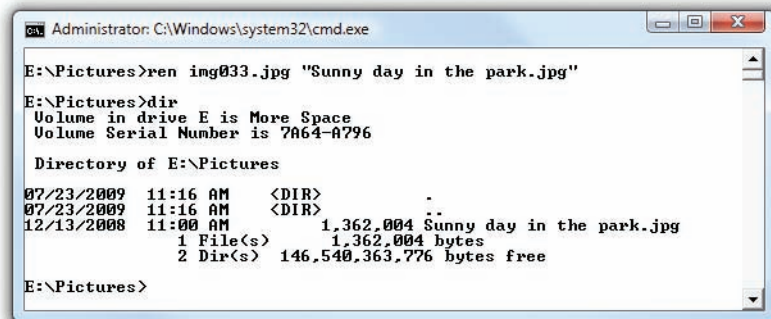
```
ren img033.jpg Sunny day in the park.jpg
```

But you'll get an error message (see Figure 18.16). Even the tried-and-true method of seeking help by typing the command followed by /? doesn't give you the answer.

You can use more complicated names by putting them in quotation marks. Figure 18.17 shows the same command that failed but now succeeds because of the quotation marks.



• **Figure 18.16** Rename failed me



• **Figure 18.17** Success at last.

## Deleting Files

To delete files, you use the **del (or erase) command**. (They're identical commands that you can use interchangeably.) Deleting files is simple—maybe too simple. Windows users enjoy the luxury of retrieving deleted files from the Recycle Bin on those “Oops, I didn't mean to delete that” occasions everyone encounters at one time or another. The command line, however, shows no such mercy to the careless user. It has no function equivalent to the Windows Recycle Bin. Once you have erased a file, you can recover it only by using a special recovery utility such as Norton's UNERASE. Again, the rule here is to *check twice and delete once*.

To delete a single file, type the **del** command followed by the name of the file to delete. To delete the file `ailog.txt`, for example, type this:

```
del ailog.txt
```

Although nothing appears on the screen to confirm it, the file is now gone. To confirm that the `ailog.txt` file is no longer listed, use the `dir` command.

As with the `dir` command, you can use wildcards with the `del` and `erase` commands to delete multiple files. For example, to delete all files with the extension `.txt` in a directory, you would type this:

```
del *.txt
```

To delete all files with the filename `config` in a directory, type **del config.\***. To delete all of the files in a directory, you can use the popular `*.*` wildcard (often pronounced “star-dot-star”), like this:

```
del *.*
```

This is one of the few command-line commands that elicits a response. Upon receiving the `del *.*` command, Windows responds with “Are you sure? (Y/N),” to which you respond with a `Y` or `N`. Pressing `Y` erases every file in the directory, so use `*.*` with care!

Don't confuse deleting *files* with deleting *directories*. Use `del` to delete files; it will not remove directories. Use `rd` to delete directories.



You'll be tested on `del`, `erase`, and `rd` on the CompTIA A+ 220-802 exam.

## Copying and Moving Files

Being able to copy and move files in a command line is crucial to all technicians. Because of its finicky nature and many options, the copy command is also rather painful to learn, especially if you're used to dragging icons in Windows. The following tried-and-true, five-step process makes it easier, but the real secret is to get in front of a `C:\` prompt and just copy and move files around until you're comfortable. Keep in mind that the only difference between copying and moving is whether the original is left behind (copy) or not (move). Once you've learned the **copy command**, you've also learned the **move command**!

### Mike's Five-Step copy/move Process

I've been teaching folks how to copy and move files for years by using this handy process. Keep in mind that hundreds of variations on this process exist. As you become more confident with these commands, try doing a

copy /? or move /? at any handy prompt to see the real power of the commands. But first, follow this process step by step:

1. Point the command prompt to the directory containing the file(s) you want to copy or move.
2. Type **copy** or **move** and a space.
3. Type the *name(s)* of the file(s) to be copied/moved (with or without wildcards) and a space.
4. Type the *path* of the new location for the file(s).
5. Press ENTER.

Let's try an example. The directory C:\Steam contains the file readme.txt. Copy this file to a USB thumb drive (E:).

1. Type **cd \steam** to point the command prompt to the Steam directory.  
C:\>cd\steam
2. Type **copy** and a space.  
C:\Steam>copy
3. Type **readme.txt** and a space.  
C:\Steam>copy readme.txt
4. Type **e:\**.  
C:\Steam>copy readme.txt e:\
5. Press ENTER.

The entire command and response would look like this:

```
C:\Steam>copy readme.txt e:\
1 file(s) copied
```

If you point the command prompt to the E: drive and type **dir**, the readme.txt file will be visible. Let's try another example. Suppose 100 files are in the C:\Docs directory, 30 of which have the .doc extension, and suppose you want to move those files to the C:\Steam directory. Follow these steps:

1. Type **cd \docs** to point the command prompt to the Docs directory.  
C:\>cd\docs
2. Type **move** and a space.  
C:\Docs>move
3. Type **\*.doc** and a space.  
C:\Docs>move \*.doc
4. Type **c:\steam**.  
C:\Docs>move \*.doc c:\steam
5. Press ENTER.  
C:\Docs>move \*.doc c:\steam
30 file(s) copied

The power of the copy/move command makes it rather dangerous. The copy/move command not only lets you put a file in a new location; it also lets you change the name of the file at the same time. Suppose you want to copy a file called autoexec.bat from your C:\ folder to a thumb drive, for example, but you want the name of the copy on the thumb drive to be auto1.bat. You can do both things with one copy command, like this:

```
copy c:\autoexec.bat e:\auto1.bat
```

Not only does the autoexec.bat file get copied to the thumb drive, but the copy also gets the new name auto1.bat.

As another example, move all of the files with the extension .doc from the C:\Docs directory to the C:\Back directory and simultaneously change the .doc extension to .sav. Here is the command:

```
move c:\docs\*.doc c:\back\*.sav
```

This says, “Move all files that have the extension .doc from the directory C:\Docs into the directory C:\Back, and while you’re at it, change their file extensions to .sav.” This is very handy, but very dangerous!

Let’s say, for example, that I made one tiny typo. Here I typed a semicolon instead of a colon after the second C:

```
move c:\docs\*.doc c;\back\*.sav
```

The command line understands the semicolon to mean “end of command” and therefore ignores both the semicolon and anything I type after it. As far as the command line is concerned, I typed this:

```
move c:\docs\*.doc c
```

This, unfortunately for me, means “take all of the files with the extension .doc in the directory C:\Docs and copy them back into that same directory, but squish them all together into a single file called c.” If I run this command, Windows gives me only one clue that something went wrong:

```
move c:\docs\*.doc c
1 file(s) copied
```

See “1 file(s) copied”? Feeling the chilly hand of fate slide down my spine, I do a dir of the directory, and I now see a single file called c, where there used to be 30 files with the extension .doc. All of my .doc files are gone, completely unrecoverable.

## **xcopy**

The standard copy and move commands can work only in one directory at a time, making them a poor choice for copying or moving files in multiple directories. To help with these multi-directory jobs, Microsoft added the **xcopy command**. (Note that there is no xmove, only xcopy.)

The xcopy command works similarly to copy, but xcopy has extra switches that give it the power to work with multiple directories. Here’s how it works. Let’s say I have a directory on my C: drive called \Data. The \Data directory has three subdirectories: \Jan, \Feb, and \Mar. All of these directories, including the \Data directory, contain about 50 files. If I wanted

to copy all of these files to my D: drive in one command, I would use xcopy in the following manner:

```
xcopy c:\data d:\data /s
```

Because xcopy works on directories, you don't have to use filenames as you would in copy, although xcopy certainly accepts filenames and wildcards. The /s switch, the most commonly used of all of the many switches that come with xcopy, tells xcopy to copy all subdirectories except for empty ones. The /e switch tells xcopy to copy empty subdirectories. When you have a lot of copying to do over many directories, xcopy is the tool to use.

## robocopy

Microsoft introduced the **robocopy command**—short for Robust File Copy—many years ago as an add-on tool for Windows Server to enable techs to manage files and folders more quickly and efficiently than with xcopy or copy. The robocopy command, now included in Windows Vista and Windows 7, is powerful indeed, enabling you to, for example, copy the files and folders from one computer to another across a network, fully replicating the structure on the destination system *and* deleting anything on that system that wasn't part of the copy. It can do this with a simple command.

The robocopy syntax does not resemble that of copy or xcopy, so if you're going to use the tool, you need to unlearn a few things. Here's the basic syntax:

```
robocopy source destination [file [file]...] [options]
```

Here's an example of the command in action. The following command would copy all files and subfolders from a local machine's D:\testserver\newwebsite folder to a shared folder on the remote server \\webhost\companywebsite.

```
robocopy d:\testserver\newwebsite \\webhost\companywebsite /mir
```

The /mir switch, for mirror, tells robocopy to copy everything from the source and make the destination mirror it. That means robocopy will also delete anything in the destination that doesn't match the source folders and files.

If that were it, robocopy would be powerful, but that's not even the tip of the iceberg. The robocopy command can copy encrypted files. It enables an administrator to copy files even if the administrator account is expressly denied access to those files. It will also resume copying after an interruption, and do so at the spot it stopped. For the full syntax, type the following:

### robocopy /?

Their power and utility make the del, copy/move, xcopy, and robocopy commands indispensable for a PC technician, but that same power and utility can cause disaster. Only a trained Jedi, with The Force as his ally...well, wrong book, but the principle remains: Beware of the quick and easy key-stroke, for it may spell your doom. Think twice and execute the command once. The data you save may be yours!



Know copy, xcopy, and robocopy for the CompTIA A+ 220-802 exam.





## Cross Check

### Command Line Versus GUI

Now that you're familiar with the command-line interface, you should be able to switch between the Windows GUI and the command line with relative ease. Check to make sure you can explain how to do the basic tasks covered in this chapter both in the Windows GUI and at the command line.

1. How do you delete a folder in Windows?
2. How do you delete a directory from the command line?
3. What advantage do wildcards give the command line over the GUI when you're working with files?

## And Even More Tools, Utilities, and Commands

As a proficient IT technician in the field, you need to be familiar with a whole slew of command-line tools and other important utilities. The CompTIA A+ 220-802 exam focuses in on several of them, and although many have been discussed in detail in previous chapters, it is extremely important that you understand and practice with `chkdsk`, `format`, `sfc`, and `shutdown`.

### `chkdsk (/f /r)`

The **`chkdsk` (checkdisk) command** scans, detects, and repairs hard drive- and volume-related issues and errors. You can run the `chkdsk` utility from a command prompt with the switches `/f` and `/r`. The `/f` switch attempts to fix volume-related errors, while the `/r` switch attempts to locate and repair bad sectors. To run successfully, `chkdsk` needs direct access to a drive. In other words, the drive needs to be "unlocked." For example, if you run `chkdsk /f /r` and `chkdsk` does not consider your drive unlocked, you will receive a "cannot lock current drive" message, meaning that another process has the drive locked and is preventing `chkdsk` from locking the drive itself. After this, `chkdsk` presents you with the option to run it the next time the system restarts (see Figure 18.18).

```
Administrator: C:\Windows\system32\cmd.exe - chkdsk /f/r c:  
C:\>chkdsk /f/r c:  
The type of the file system is NTFS.  
Cannot lock current drive.  
  
Chkdsk cannot run because the volume is in use by another  
process. Would you like to schedule this volume to be  
checked the next time the system restarts? (Y/N) _
```

• **Figure 18.18** The `chkdsk /f /r` utility and switches on a locked drive



## Cross Check

### Error-checking

You've seen the graphical version of `chkdsk` way back in Chapter 12, but how about refreshing your memory? How do you get to the Error-checking utility in Windows' graphical user interface? Does it have the same functionality as the command-line version's switches?

```

Administrator: C:\Windows\system32\cmd.exe
C:\>format /?
Formats a disk for use with Windows.

FORMAT volume [/FS:file-system] [/U:label] [/Q] [/A:size] [/C] [/X] [/P:passes]
FORMAT volume [/U:label] [/Q] [/F:size] [/P:passes]
FORMAT volume [/U:label] [/Q] [/I:tracks] [/N:sectors] [/P:passes]
FORMAT volume [/U:label] [/Q] [/P:passes]
FORMAT volume [/Q]

volume           Specifies the drive letter (followed by a colon),
                  mount point, or volume name.
/FS:filesystem   Specifies the type of the file system (FAT, FAT32, exFAT, NTFS,
                  or UDF).
/U:label        Specifies the volume label.
/Q             Performs a quick format. Note that this switch overrides /P.
/C            NTFS only: Files created on the new volume will be compressed
                  by default.
/X            Forces the volume to dismount first if necessary. All opened
                  handles to the volume would no longer be valid.
/R:revision     UDF only: Forces the format to a specific UDF version
                  (1.02, 1.50, 2.00, 2.01, 2.50). The default
                  revision is 2.01.
/D            UDF 2.50 only: Metadata will be duplicated.
/A:size        Overrides the default allocation unit size. Default settings
                  are strongly recommended for general use.
                  NTFS supports 512, 1024, 2048, 4096, 8192, 16K, 32K, 64K.
                  FAT supports 512, 1024, 2048, 4096, 8192, 16K, 32K, 64K,
                  (128K, 256K for sector size > 512 bytes).
                  FAT32 supports 512, 1024, 2048, 4096, 8192, 16K, 32K, 64K,
                  (128K, 256K for sector size > 512 bytes).
                  exFAT supports 512, 1024, 2048, 4096, 8192, 16K, 32K, 64K,
                  128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M.

Note that the FAT and FAT32 file systems impose the
following restrictions on the number of clusters on a volume:

FAT: Number of clusters <= 65526
FAT32: 65526 < Number of clusters < 4177918

Format will immediately stop processing if it decides that
the above requirements cannot be met using the specified
cluster size.

NTFS compression is not supported for allocation unit sizes
above 4096.

/F:size        Specifies the size of the floppy disk to format (1.44)
/I:tracks      Specifies the number of tracks per disk side.
/N:sectors     Specifies the number of sectors per track.
/P:passes      Zero every sector on the volume passes times. This switch is
                  not valid with /Q
C:\>

```

• **Figure 18.19** Using format /? at the command prompt

## format

After the previous chapters, you should have an expert-level knowledge of (or, at the very least, a passing familiarity with) formatting and partitioning hard drives. Formatting, you may remember, prepares a partition or volume so it can hold an operating system or data. We have already discussed the various built-in Windows utilities available to provide the formatting of drives, and you no doubt know that many third-party formatting tools are out there. In this chapter, you just need to become familiar with the format command and its switches.

The **format command**, you may have guessed, enables you to format disks from the command line. The very best way to familiarize yourself with the format command and its available switches is simply to enter **format /?** from the command prompt. Your results should be

similar to those displayed in Figure 18.19.

Although the new CompTIA A+ 220-802 exam focuses primarily on GUI operating system formatting utilities and options, you should familiarize yourself with the format command and its switches by practicing them on a test system you are literally not afraid to wipe out. Besides, you never know what skeletons CompTIA may pull out of the closet.

## hostname

The **hostname command** is the most straightforward of all command-line commands. If you type **hostname** at the command prompt, it will display the name of your computer, also known as the hostname. When I type **hostname**, for example, it prints “MikePC.”

## sfc (System File Checker)

The Windows **sfc (System File Checker)**, or simply sfc.exe, scans, detects, and restores important Windows system files, folders, and paths. Techs often use the sfc utility from within a working version of Windows or from a Windows installation disc to restore a corrupt Windows environment. If you run sfc and it finds issues, it attempts to replace corrupted or missing files from cached DLLs located in the %WinDir%\System32\Dllcache\ directory. Without getting very deep into the mad science involved, just know that you can use sfc to correct corruption. To run sfc from a command prompt,

enter `sfc /scannow`. To familiarize yourself with `sfc`'s switches, enter `sfc /?` (see Figure 18.20).

## shutdown

The **shutdown command** enables you to do exactly that to a local or remote computer, namely, shut it down. The cool part of the tool is that you can use a number of switches to control and report the shutdown. A network administrator could use this tool to restart a computer remotely, for example with a few keystrokes, like this:

```
shutdown /r /m \\testserver
```

The `/r` switch tells `shutdown` to have the computer reboot rather than just shut down. If you want to see the full syntax for `shutdown`, type the following:

```
shutdown /?
```

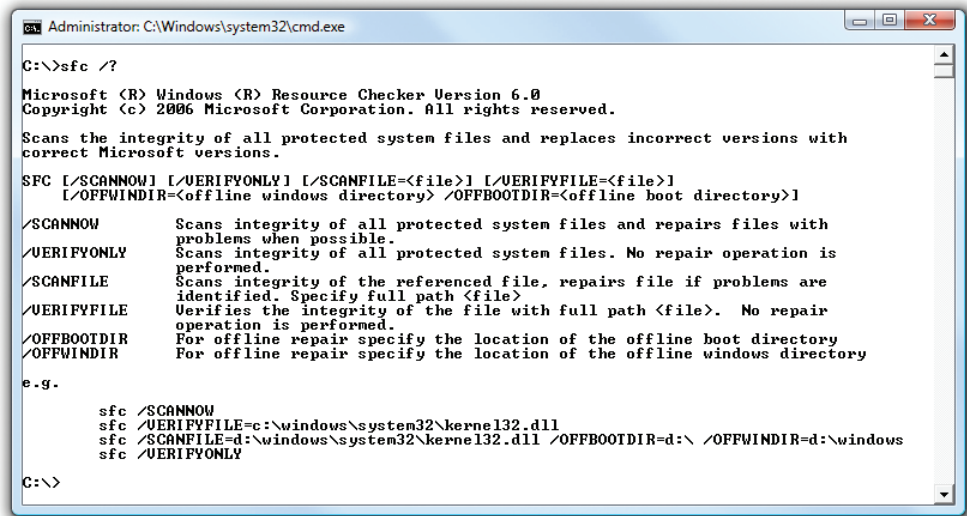
## ■ Beyond A+

### Using Special Keys

You might find yourself repeatedly typing the same commands, or at least very similar commands, when working at a prompt. Microsoft has provided a number of ways to access previously typed commands. Type the `dir` command at a command prompt. When you get back to a prompt, press `F1`, and the letter `d` appears. Press `F1` again. Now the letter `i` appears after the `d`. Do you see what is happening? The `F1` key brings back the previous command one letter at a time. Pressing `F3` brings back the entire command at once. Now try running these three commands:

```
dir /w
attrib
md fred
```

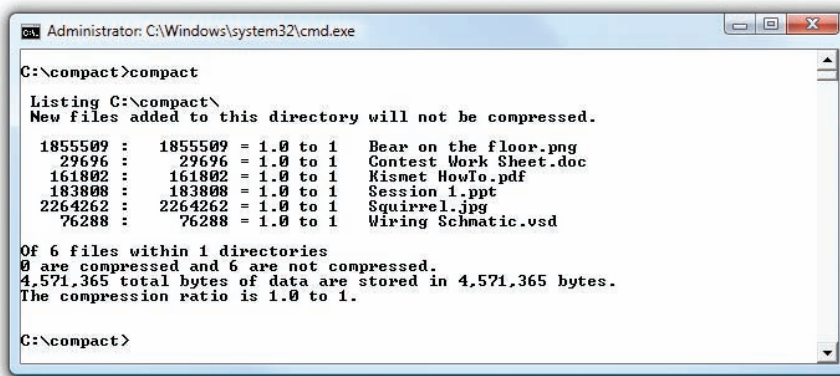
Now press the `UP ARROW` key. Keep pressing it till you see your original `dir` command—it's a history of all your old commands. Now use the `RIGHT ARROW` key to add `/p` to the end of your `dir` command. Windows command history is very handy.



• **Figure 18.20** Checking `sfc` options with `sfc /?` at a command prompt

## The compact and cipher Commands

Windows offers two cool commands at the command-line interface: `compact` and `cipher`. The `compact` command displays or alters the compression of files on NTFS partitions. The `cipher` command displays or alters the encryption of folders and files on NTFS partitions. If you type just the command with no added parameters, `compact` and `cipher` display the compression state and the encryption state, respectively, of the current directory and any files it contains. You may specify multiple directory names, and you may use wildcards, as you learned earlier in the chapter. You must add parameters to make the commands change things. For example, you add `/c` to compress and `/u` to uncompress directories and/or files with the `compact` command, and you add `/e` to encrypt and `/d` to decrypt directories and/or files with the `cipher` command. When you do these operations, you also mark the directories involved so that any files you add to them in the future will take on their encryption or compression characteristics. In other words, if you encrypt a directory and all its files, any files you add later will also be encrypted. Same thing if you compress a directory. I'll run through a quick example of each.



```
Administrator: C:\Windows\system32\cmd.exe
C:\compact>compact
Listing C:\compact\
New files added to this directory will not be compressed.

1855509 : 1855509 = 1.0 to 1   Bear on the floor.png
29696   : 29696   = 1.0 to 1   Contest Work Sheet.doc
161802  : 161802  = 1.0 to 1   Kismet HowTo.pdf
183808  : 183808  = 1.0 to 1   Session 1.ppt
2264262 : 2264262 = 1.0 to 1   Squirrel.jpg
76288   : 76288   = 1.0 to 1   Wiring Schematic.usd

Of 6 files within 1 directories
0 are compressed and 6 are not compressed.
4,571,365 total bytes of data are stored in 4,571,365 bytes.
The compression ratio is 1.0 to 1.

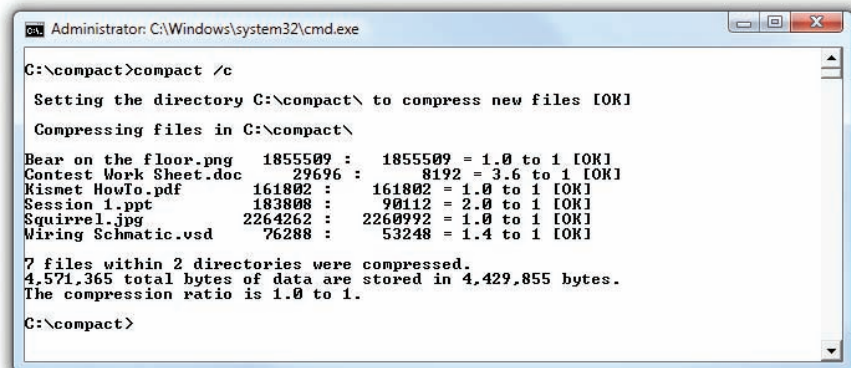
C:\compact>
```

### compact

First let's try the `compact` command. Figure 18.21 shows the result of entering the `compact` command with no switches. It displays the compression status of the contents of a directory called `compact` on a system's C: drive. Notice that after the file listing, `compact` helpfully tells you that 0 files are compressed and 6 files (all of them) are not compressed, with a total compression ratio of 1.0 to 1.

• **Figure 18.21** The `compact` command with no switches

If you enter the `compact` command with the `/c` switch, it compresses all of the files in the directory, as shown in Figure 18.22. Look closely at the list-



```
Administrator: C:\Windows\system32\cmd.exe
C:\compact>compact /c
Setting the directory C:\compact\ to compress new files [OK]
Compressing files in C:\compact\
Bear on the floor.png 1855509 : 1855509 = 1.0 to 1 [OK]
Contest Work Sheet.doc 29696 : 8192 = 3.6 to 1 [OK]
Kismet HowTo.pdf 161802 : 161802 = 1.0 to 1 [OK]
Session 1.ppt 183808 : 90112 = 2.0 to 1 [OK]
Squirrel.jpg 2264262 : 2260992 = 1.0 to 1 [OK]
Wiring Schematic.usd 76288 : 53248 = 1.4 to 1 [OK]

7 files within 2 directories were compressed.
4,571,365 total bytes of data are stored in 4,429,855 bytes.
The compression ratio is 1.0 to 1.

C:\compact>
```

• **Figure 18.22** Typing `compact /c` compresses the contents of the directory.

ing. Notice that it includes the original and compressed file sizes and calculates the compression ratio for you. Notice also that the JPG and PNG files (both compressed graphics files) didn't compress at all, while the Word file and the PowerPoint file compressed down to around a third of their original sizes. Also, can you spot what's different in the text at the bottom of the screen? The compact command claims to have compressed *seven* files in *two* directories! How can this be? The secret is that when it compresses all of the files in a directory, it must also compress the directory file itself, which is "in" the C: directory above it. Thus it correctly reports that it compressed seven files: six in the compact directory, and one in the C: directory.

Typing **compact** again shows you the directory listing, and now there's a C next to each filename, indicating that the file is compressed (see Figure 18.23).

Okay, now suppose you want to uncompress a file—say a PowerPoint file, *Session 1.ppt*. To do this, you must specify the decompression operation, using the **/u** switch and the name of the file you want decompressed, as shown in Figure 18.24. Note that compact reports the successful decompression of one file only: *Session 1.ppt*. You could do the same thing in reverse, using the **/c** switch and a filename to compress an individual file.

```
Administrator: C:\Windows\system32\cmd.exe
C:\compact>compact
Listing C:\compact\
New files added to this directory will be compressed.
1855509 : 1855509 = 1.0 to 1 C Bear on the floor.png
29696 : 8192 = 3.6 to 1 C Contest Work Sheet.doc
161802 : 161802 = 1.0 to 1 C Kismet Howlo.pdf
183808 : 90112 = 2.0 to 1 C Session 1.ppt
2264262 : 2260992 = 1.0 to 1 C Squirrel.jpg
76288 : 53248 = 1.4 to 1 C Wiring Schematic.usd
Of 6 files within 1 directories
6 are compressed and 0 are not compressed.
4,571,365 total bytes of data are stored in 4,429,855 bytes.
The compression ratio is 1.0 to 1.
C:\compact>
```

• **Figure 18.23** The contents of C:\compact have been compressed.

```
Administrator: C:\Windows\system32\cmd.exe
C:\compact>compact /u "Session 1.ppt"
Uncompressing files in C:\compact\
Session 1.ppt [OK]
1 files within 1 directories were uncompresssed.
C:\compact>
```

• **Figure 18.24** Typing **compact /u "Session 1.ppt"** decompresses only that file.

## cipher

The cipher command is a bit complex, but in its most basic implementation, it's pretty straightforward. Figure 18.25 shows two steps in the process. Like the compact command, the cipher command simply displays the current state of affairs when entered with no switches. In this case, it displays the encryption state of the files in the E:\Work Files\Armor Pictures directory. Notice the letter *U* to the left of the filenames, which tells you they are unencrypted. The second command you can see on the screen in Figure 18.25 is this:

```
E:\Work Files\Armor Pictures>cipher /e /a
```

This time the cipher command carries two switches: **/e** specifies the encryption operation, and **/a** says to apply it to the *files* in the directory, not just the directory itself. As you can see, the command-line interface is actually pretty chatty in this case. It reports that it's doing the encryption and



then tells you what it's done, and it even warns you that you should clean up any stray unencrypted bits that may have been left in the directory.

To confirm the results of the cipher operation, enter the **cipher** command again, as shown in Figure 18.26. Note that the *U* to the left of each filename has been replaced with an *E*, indicating an encrypted file. The other indication that this directory has been encrypted is the statement above the file listing:

New files added to this directory will not be encrypted.

Remember that the cipher command works on directories first and foremost, and it works on individual files only when you specifically tell it to do so.

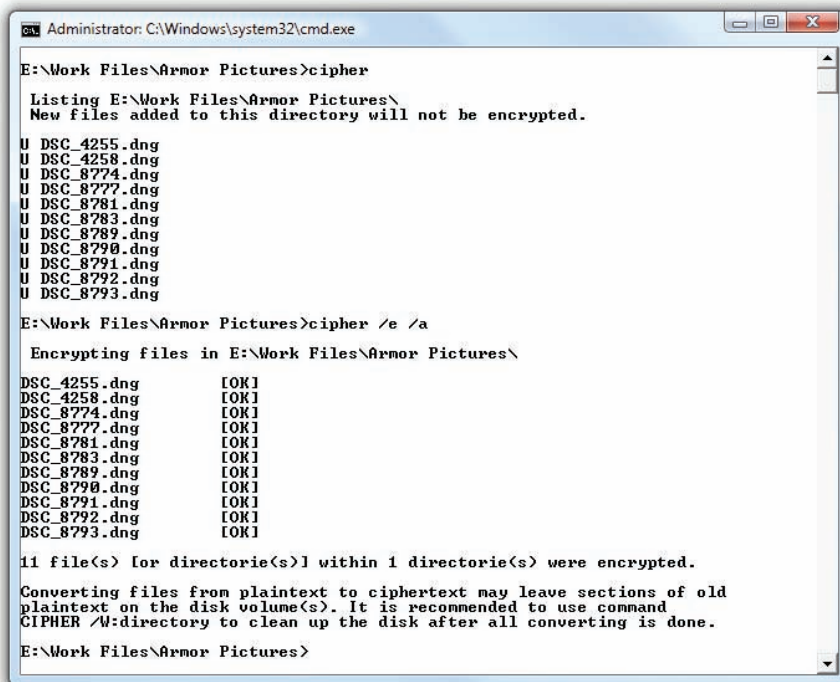
That's great, but suppose you want to decrypt just *one* of the files in the Armor Pictures directory. Can you guess how you need to alter the command? Simply add the filename of the file you want to decrypt after the command and the relevant switches. Figure 18.27 shows the cipher command being used to decipher `dsc_4255.dng`, a single file.

## PowerShell

Microsoft introduced an add-on command-line interface called PowerShell for Windows XP and finally integrated it into Windows 7 as a more powerful replacement for the traditional command-line interface. PowerShell enables you to do all the typical command-line activities, such as `dir`, `cd`, `md`, and so on, but brings a series of vastly more powerful tools called *cmdlets* that enable you to accomplish some amazing tasks. Figure 18.28 shows two commands that do the same thing by default—show the contents of a directory: `dir` and `get-childitem`.

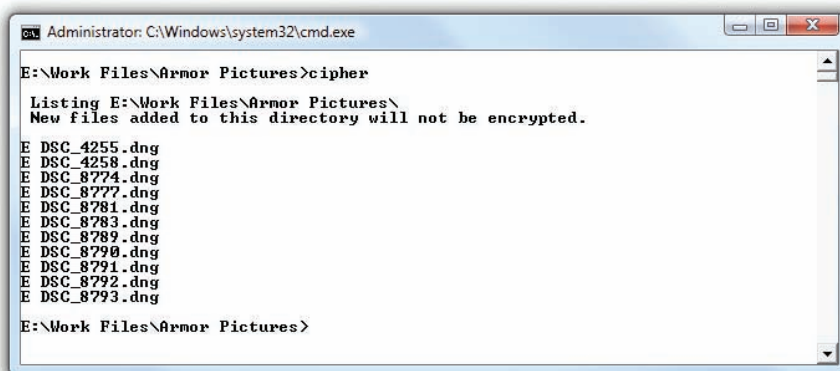
With `dir`, you know you can search for items in a directory, such as “find all the JPEG files in a folder” with this command:

```
dir *.jpg
```



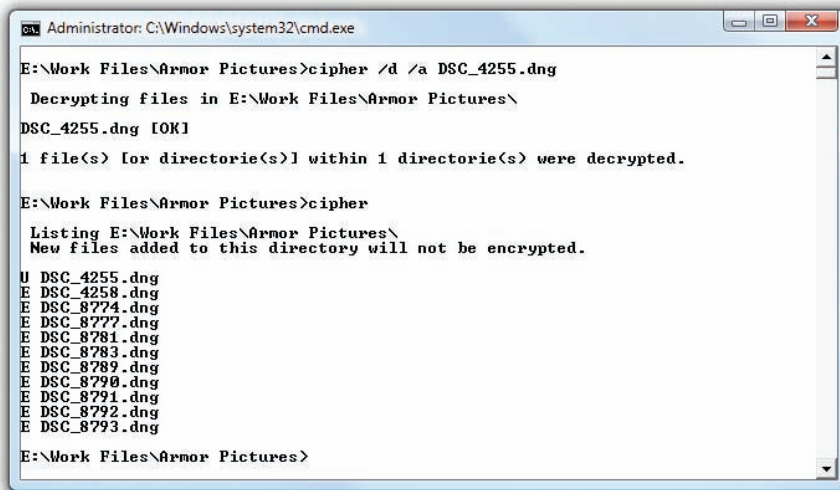
```
Administrator: C:\Windows\system32\cmd.exe
E:\Work Files\Armor Pictures>cipher
Listing E:\Work Files\Armor Pictures\
New files added to this directory will not be encrypted.
U DSC_4255.dng
U DSC_4258.dng
U DSC_8774.dng
U DSC_8777.dng
U DSC_8781.dng
U DSC_8783.dng
U DSC_8789.dng
U DSC_8790.dng
U DSC_8791.dng
U DSC_8792.dng
U DSC_8793.dng
E:\Work Files\Armor Pictures>cipher /e /a
Encrypting files in E:\Work Files\Armor Pictures\
DSC_4255.dng      [OK]
DSC_4258.dng      [OK]
DSC_8774.dng      [OK]
DSC_8777.dng      [OK]
DSC_8781.dng      [OK]
DSC_8783.dng      [OK]
DSC_8789.dng      [OK]
DSC_8790.dng      [OK]
DSC_8791.dng      [OK]
DSC_8792.dng      [OK]
DSC_8793.dng      [OK]
11 file(s) for directorie(s)] within 1 directorie(s) were encrypted.
Converting files from plaintext to ciphertext may leave sections of old
plaintext on the disk volume(s). It is recommended to use command
CIPHER /W:directory to clean up the disk after all converting is done.
E:\Work Files\Armor Pictures>
```

• **Figure 18.25** Typing `cipher /e /a` encrypts the contents of the directory.



```
Administrator: C:\Windows\system32\cmd.exe
E:\Work Files\Armor Pictures>cipher
Listing E:\Work Files\Armor Pictures\
New files added to this directory will not be encrypted.
E DSC_4255.dng
E DSC_4258.dng
E DSC_8774.dng
E DSC_8777.dng
E DSC_8781.dng
E DSC_8783.dng
E DSC_8789.dng
E DSC_8790.dng
E DSC_8791.dng
E DSC_8792.dng
E DSC_8793.dng
E:\Work Files\Armor Pictures>
```

• **Figure 18.26** The cipher command confirms that the files were encrypted.



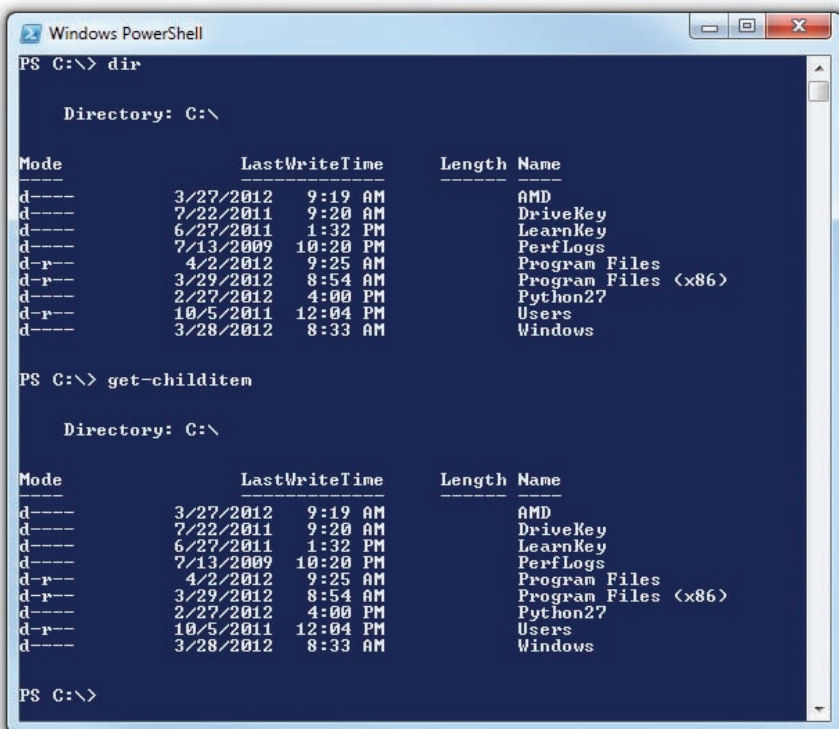
• **Figure 18.27** Typing `cipher /d /a dsc_4255.dng` decrypts only that file.

PowerShell lets you dial it all the way to 11, though, with a few more characters on a search string. The following command will find all the JPEG files on the entire computer, plus all the mentions of .jpg files in any document format, such as Word, PowerPoint, and Acrobat:

```
get-childitem . -include *.jpg -recurse -force
```

Just in case you want to know what each piece of that string in the cmdlet means, here's the scoop: `get-childitem` is the main command. The dot `<.>` indicates the current directory and all its subdirectories, while `-include` tells the command to fetch the file type indicated next, in this case `*.jpg`. `-recurse` makes the command repeat, and `-force` retrieves hidden files too. It's like `dir` on steroids!

To open PowerShell in Windows 7, simply type `powershell` in the Search bar and press ENTER. To use the tool in Windows XP or Windows Vista, you'll need to download it from Microsoft. Just go to Microsoft.com and search for **PowerShell download**. You'll find it readily enough. You might also need to install the .NET 2.0 framework if you're on an older machine. You can find that at the same Web site. Good luck!



• **Figure 18.28** Simple commands in PowerShell

# Chapter 18 Review

## ■ Chapter Summary

After reading this chapter and completing the exercises, you should understand the following about the command-line interface.

### Explain the operation of the command-line interface

- The text-based DOS-like user interface, known as the command-line interface, functions as a basic installation and troubleshooting tool for techs in Windows. When you use a command-line interface, the computer tells you it's ready to receive commands by displaying a specific set of characters called a prompt. You type a command and press `ENTER` to send it. The OS goes off and executes the command, and when it's finished, it displays a new prompt, often along with some information about what it did. The new prompt means the computer is ready for your next instruction.
- You can access the command-line interface in Windows XP by clicking `Start | Run` to open the Run dialog box, and then typing `cmd` or `command` and pressing `ENTER`. In Windows Vista or Windows 7, you enter one of the same commands in the Start menu Search bar. You can also click `Start | Programs/All Programs | Accessories | Command Prompt`. To close the command line–interface window, either click the Close box in the upper-right corner of the window or type `exit` at the prompt and press `ENTER`.
- The command prompt is always focused on some directory, and any commands you issue are performed *on the files in the directory* on which the prompt is focused. Make sure you focus the prompt's attention on the drive and directory in which you want to work.
- Windows manifests each program and piece of data as an individual *file*. Each file has a name, which is stored with the file on the drive. Names are broken down into two parts: the filename and the extension. In true DOS, the filename could be no longer than eight characters. The extension, which is optional, could be up to three characters long. No spaces or other illegal characters (`/ \ [ ] | ÷ + = ; , * ?`) could be used in the filename or extension. The filename and extension are separated by a period, or dot. This naming system is known as the “eight-dot-three” (8.3) naming system. All versions of Windows starting with 9x allow filenames of up to 255 characters.
- The filename extension tells the computer the type or function of the file. Program files use the extension `.exe` (for executable) or `.com` (for command). If the file is data, the extension indicates which program uses that particular data file. Graphics files most often reflect the graphics standard used to render the image, such as `.gif` or `.jpg`. Changing the file extension does not change the data in the file. The Windows GUI doesn't show the file extensions by default.
- All files are stored on the hard drive in binary format, but every program has a unique method of binary organization, called a file format. One program cannot read another program's files unless it can convert the other program's file format into its file format.
- ASCII (American Standard Code for Information Interchange) was the first universal file format. The ASCII standard defines 256 eight-bit characters, including all of the letters of the alphabet (uppercase and lowercase), numbers, punctuation, many foreign characters, box-drawing characters, and a series of special characters for commands, such as a carriage return, bell, and end of file. ASCII files, often called text files, store all data in ASCII format. The keyboard sends the characters you press on the keyboard to the PC in ASCII code. Even the monitor outputs in ASCII when you are running a command line.
- As a rule, the OS treats the first 32 ASCII values as commands. Some of them are both commands and characters. How these first 32 values are treated depends on the program that reads them.
- Unicode supports thousands of 16-bit characters. The first 256 Unicode characters are the same as the complete 256 ASCII character set, which maintains backward compatibility.
- At boot, the OS assigns a drive letter to each hard drive partition and to each floppy or other disk drive. The first floppy drive, if installed, is called A:, and the second, if installed, is called B:. Hard drives start with the letter C: and can continue to

Z: if necessary. Optical drives usually get the next available drive letter after the last hard drive.

- Windows uses a hierarchical directory tree to organize the contents of these drives. All files are put into groups called directories. Windows also uses directories, but it calls them folders. Any file not in a directory *within* the tree—which is to say, any file in the directory at the root of the directory tree—is said to be in the root directory. Directories inside directories are called subdirectories. Any directory can have multiple subdirectories. Two or more files or subdirectories with the same name can exist in different directories on a PC, but two files or subdirectories in the same directory cannot have the same name.
- When describing a drive, you use its assigned letter, such as C: for the hard drive. To describe the root directory, add a backslash (\), as in C:\. To describe a particular directory, add the name of the directory after the backslash. To add a subdirectory after the directory, add another backslash and then the subdirectory's name. This naming convention provides for a complete description of the location and name of any file. The exact location of a file is called its path.

### Execute fundamental commands from the command line

- The command-line interface will do what you *say*, not what you *mean*, so it always pays to double-check that those are the same before you press ENTER. One careless keystroke can result in the loss of crucial data, with no warning and no going back.
- The dir command shows you the contents of the directory on which the prompt is focused. The dir command lists the filename, extension, file size in bytes, and creation date/time. The dir /p command pauses after displaying a screen's worth of the directory contents; press the SPACEBAR to display the next screen. The dir /w command shows you filenames only, arranged in columns, with directory names in square brackets.
- Extra text typed after a command to modify its operation, such as /w or /p after dir, is called a switch. Almost all switches can be used simultaneously to modify a command. Typing any command followed by /? brings up a help screen for that particular command.
- You can use the cd command to change the focus of the command prompt to a different directory.

Type `cd\` followed by the name of the directory on which you want to focus the prompt, and press ENTER. If no such directory exists or if you mistyped the name, Windows will report "The system cannot find the path specified." To return to the root directory, type `cd\` and press ENTER. The cd command also allows you to use a space instead of a backslash. The cd command is not used to move between drives; to point the prompt to another drive, type the drive letter and a colon.

- To make a directory, use the md (or mkdir) command. Once you press ENTER, the OS executes the command, but it won't volunteer any information about what it did. You must use the dir command to see that you have, in fact, created a new directory. Make sure that the prompt points to the directory in which you want to make the new subdirectory before you execute the md command.
- To remove a directory, first point the prompt at the directory that contains the subdirectory you want to delete, and then execute the rd (or rmdir) command. If you get no response, you probably did it right, but use the dir command to be sure. The rd command will not delete a directory if it contains files or subdirectories; you must first empty that directory by using the del (for files) or rd (for subdirectories) command. The rd command followed by the /S switch deletes the directory as well as all files and subdirectories.

### Manipulate files from the command line

- All files have four special values, or attributes, that determine how they will act in special situations: hidden, read-only, system, and archive. You can set these attributes through software. If a file is hidden, it will not be displayed when the dir command is performed. A read-only file cannot be modified or deleted. The system attribute, which is used only for system files such as ntldr and ntddetect.com found in Windows XP, provides an easy identifier for these files. The archive attribute is used by backup software to identify files that have been changed since their last backup.
- attrib.exe is an external program with which you can inspect and change file attributes. Type the **attrib** command followed by the name of the file and press ENTER. In the resulting list of files, letter codes indicate each file's attributes, if any: A stands for archive, R means read-only, H is hidden, and S is system. The attrib command can change a file's



attributes. To add an attribute to a file, type the attribute letter preceded by a plus sign (+), and then the filename. To delete an attribute, use a minus sign (-). Multiple attributes can be added or removed in one command.

- Wildcards are two special characters, asterisk (\*) and question mark (?), that you can use in place of all or part of a filename to make a command act on more than one file at a time. Wildcards work with all command-line commands that take filenames. The asterisk (\*) wildcard replaces any part of a filename, before and/or after the period. The ? wildcard replaces any single character. Virtually every command that deals with files will take wildcards.
- Use the ren (or rename) command to rename files and folders. If the new name contains spaces, enclose the name in quotation marks.
- To delete files, you use the del or erase command. del and erase are identical commands and can be used interchangeably. del will not erase directories. To delete a single file, type the del command followed by the name of the file. No confirmation will appear on the screen. To delete all of the files in a directory, you can use the \*.\* (star-dot-star) wildcard. Upon receiving the del \*.\* command, Windows responds with "Are you sure? (Y/N)," to which you respond with a Y or N. Pressing Y erases every file in the directory. It pays to check twice before you delete, because the command line has no function equivalent to the Windows Recycle Bin that allows you to retrieve an accidentally deleted file; once a file has been erased, you can recover it only by using a special recovery utility.
- The copy and move commands are used to copy and move files. The only difference between them is whether the original file is left behind (copy) or not (move). Type copy or move and a space. Point the prompt to the directory containing the file(s) to be copied or moved. Type the name(s) of the file(s) to be copied/moved (with or without wildcards) and a space. Type the path of the new location for the file(s). Press ENTER. The copy/move command not only lets you put a file in a new location; it also lets you change the name of the file at the same time.
- Check for the common typo of substituting a semicolon for a colon, because the semicolon means "end of command" and therefore ignores both the semicolon and anything you type after it.
- The xcopy command works similarly to copy but has extra switches that give xcopy the power to work with multiple directories. Because xcopy works on directories, you don't have to use filenames as you would in copy, although you can use filenames and wildcards. The /s switch tells xcopy to copy all subdirectories except empty ones. The /e switch tells it to copy empty subdirectories.
- The robocopy command acts like copy on steroids, enabling you to copy full directories, including structures, and even erase anything that doesn't match on the destination drive.
- A plethora of tools, utilities, and commands are available to prepare, maintain, detect, and/or correct operating system file structures. Many of these, including chkdsk, format, sfc, and shutdown, can be started or run from a command prompt and/or from the Windows GUI.

## ■ Key Terms

### 8.3 naming system (648)

**American Standard Code for Information Interchange (ASCII) (649)**

**attrib.exe (659)**

**attributes (659)**

**cd (chdir) command (654)**

**chkdsk (checkdisk) command (667)**

**command-line interface (644)**

**copy command (663)**

**del (erase) command (663)**

**dir command (652)**

**format command (668)**

**hostname command (668)**

**md (mkdir) command (655)**

**move command (663)**

**path (651)**

**prompt (644)**

**read-only attribute (659)**

**rd (rmdir) command (656)**

**ren (rename) command (662)**

**robocopy command (666)**

**root directory (651)**



**Run dialog box** (647)  
**sfc (System File Checker)** (668)  
**shutdown command** (669)  
**switch** (652)

**syntax** (652)  
**Unicode** (650)  
**wildcard** (661)  
**xcopy command** (665)

## ■ Key Term Quiz

---

Use the Key Terms list to complete the sentences that follow. Not all terms will be used.

1. The \_\_\_\_\_ tells you it's ready to receive commands by displaying a specific set of characters called a(n) \_\_\_\_\_.
2. Extra text you type after a command to modify its operation is called a(n) \_\_\_\_\_.
3. The first universal file format was called \_\_\_\_\_.
4. The \_\_\_\_\_ shows you the contents of the directory that currently has focus.
5. Each file's \_\_\_\_\_ determine how programs (such as My Computer or Computer) treat the file in special situations.
6. The \_\_\_\_\_ enables you to restart a computer remotely from the command line.
7. An external program that enables you to inspect and change file attributes is \_\_\_\_\_.
8. The asterisk is a special character called a(n) \_\_\_\_\_ that you can use in place of part of a filename when executing a command-line command.
9. Use the \_\_\_\_\_ to rename a file.
10. To move a file from the root directory into a subfolder, use the \_\_\_\_\_.

## ■ Multiple-Choice Quiz

---

1. Which of the following is an illegal character in a Windows filename?  
A. \* (asterisk)  
B. . (dot)  
C. - (dash)  
D. \_ (underscore)
2. Which command pauses after displaying a screen's worth of directory contents?  
A. dir p  
B. pdir  
C. pd  
D. dir /p
3. Which of the following commands will delete all of the files in a directory?  
A. del \*.\*  
B. del all  
C. del ??  
D. del \*?
4. Which command do you use to change the focus of the command prompt to a different directory?  
A. dir /n  
B. cddir  
C. cd  
D. dir /c
5. Which attribute keeps a file from being displayed when the dir command is performed?  
A. Hidden  
B. Archive  
C. Read-only  
D. Protected
6. What command enables you to make a new directory in a Windows 7 Professional system?  
A. mf  
B. mkfol  
C. md  
D. makedir

7. What commands can you type at the Run dialog box to access the command-line interface in Windows XP? (Select two.)
  - A. cmd
  - B. command
  - C. msdos
  - D. prompt
8. Joey wants to change the name of a file from start.bat to hammer.bat. Which of the following commands would accomplish this feat?
  - A. ren hammer.bat start.bat
  - B. ren start.bat hammer.bat
  - C. rename /s start.bat hammer.bat
  - D. rename /s hammer.bat start.bat
9. What is the name for the rules for typing a command correctly?
  - A. Protocol
  - B. Syntax
  - C. Legacy
  - D. Instruction set
10. What is the command to make myfile.txt read-only?
  - A. attrib myfile.txt +r
  - B. attrib myfile.txt -r
  - C. readonly myfile.txt
  - D. myfile.txt /readonly
11. What is the maximum number of characters in a Windows filename?
  - A. 8
  - B. 255
  - C. 65,536
  - D. No limit
12. What user profile directory is displayed at a Windows Vista command prompt by default?
  - A. C:\All Users\User name
  - B. C:\Users\User name
  - C. C:\Documents and Settings\User name
  - D. C:\Windows\system32\drivers\etc
13. How do you run a command at the Windows 7 command prompt with elevated or administrative privileges?
  - A. Enter an elevated username and password at the command prompt.
  - B. Right-click a command-prompt shortcut and then select Run as PowerUser.
  - C. Right-click a command-prompt shortcut and then select Run as administrator.
  - D. When prompted, enter a valid root or supervisor password.
14. What tool would you use at a Windows 7 command prompt to manage files and folders more quickly and efficiently than with xcopy or copy?
  - A. recopy
  - B. copynow
  - C. attrib
  - D. robocopy
15. What tool enables you to reboot a computer from the command line?
  - A. boot
  - B. reboot
  - C. restart
  - D. shutdown

## ■ Essay Quiz

---

1. You've been tasked to teach some newbie techs, whose only computer experience involves Windows, about the beauty and power of the command line. For their first lesson, write a short essay explaining how the command-line interface works, including the directory structure, filename limitations, and interaction with the command prompt.
2. You work the help desk at a college computing center. Some applications that the students must use are run from a command line. Write a memo for the help desk personnel explaining how to use the dir, cd, md, del, and rd commands, including any appropriate cautions.

3. Your coworker needs to remove the hidden attribute from a file called payroll.xls that is stored in the Employee directory on the D: drive. Write an e-mail explaining how to open a command-line window, navigate to the file, verify its presence, and remove the hidden attribute.
4. You've been tasked to teach some newbie techs, whose only computer experience involves Windows, about the beauty and power of the command line. For their second lesson, write a short essay explaining how wildcards work.
5. Write a brief essay explaining in your own words how to use Mike's Five-Step copy/move process. Give an example.

## Lab Projects

### • Lab Project 18.1

For each of the following files, translate the location into a path you can type at a command prompt.

- 1 A file named beach.jpg in the subdirectory Pictures in the directory Eziba on the primary floppy drive:  
\_\_\_\_\_
- 2 A file named blackdog.wav in the subdirectory Ledzep in the subdirectory Rock in the directory Music on the C: drive:  
\_\_\_\_\_
- 3 A file named weapon.pcx in the subdirectory Bobafett in the subdirectory Players in the subdirectory Baseq2 in the directory Quake2 on the D: drive:  
\_\_\_\_\_

- 4 A file named autoexec.bat in the root directory on a standard PC with a single hard drive:  
\_\_\_\_\_

- 5 A file named meyers.doc in the subdirectory Contracts in the subdirectory Legal in the directory Accounts on a CD-ROM on a system with one hard drive, one CD-ROM drive, and one floppy drive:  
\_\_\_\_\_

### • Lab Project 18.2

To practice making/removing directories and copying/moving files, do the following:

- 1 Open a command-line window and point to the root directory, C:.
- 2 Use the md command to create a directory called Whales. Use the cd command to point to the new directory. Use the md command again to create a subdirectory of Whales called Mobydick. Use the cd command to point to the new subdirectory.
- 3 Populate the Mobydick subdirectory with files by opening Notepad and creating a "dummy" file called ahab.txt. Save it in the Mobydick subdirectory.
- 4 Use the move command to move the file ahab.txt to the Whales directory. Use the dir command to confirm that the move was successful. Use the copy command to put a copy of ahab.txt back into the Mobydick subdirectory while leaving a copy in Whales. Again use dir to confirm the operation.
- 5 Use the rd command to delete the Mobydick subdirectory. Did it work? Why not? Use the rd command with the /s switch to solve the problem.