CHAPTER 15

Working with the Command-Line Interface

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 Vista, 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+ certification exams test 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 Macintosh 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, better known as the PC. 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. They 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 in Windows XP and in Vista 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.

Practical Application

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.

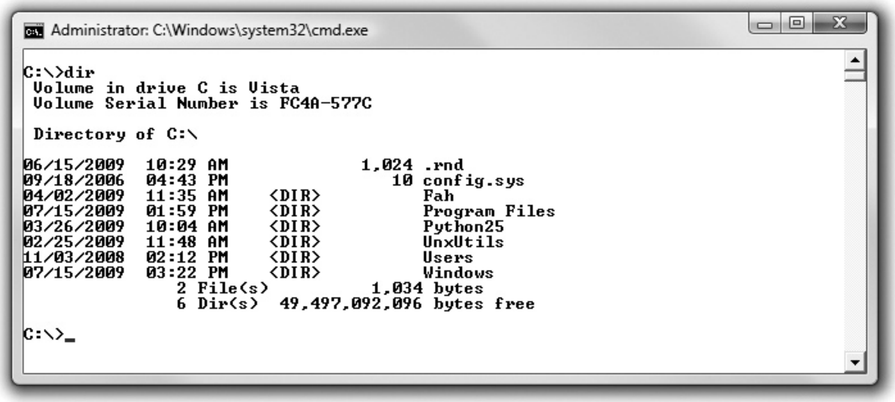
```
Mike: What kind of game?
Computer: _
```

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 graphical user interface (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 15-1.



```
C:\>dir
 Volume in drive C is Vista
 Volume Serial Number is FC4A-577C

 Directory of C:\

06/15/2009  10:29 AM             1,024 .rnd
09/18/2006  04:43 PM                10 config.sys
04/02/2009  11:35 AM    <DIR>          Fah
07/15/2009  01:59 PM    <DIR>          Program Files
03/26/2009  10:04 AM    <DIR>          Python25
02/25/2009  11:48 AM    <DIR>          UnxUtils
11/03/2008  02:12 PM    <DIR>          Users
07/15/2009  03:22 PM    <DIR>          Windows
               2 File(s)          1,034 bytes
               6 Dir(s)  49,497,092,096 bytes free

C:\>_
```

Figure 15-1 Contents of C: directory from the command line

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 15-2, which at first glance isn't much like the command-line screen; however, simply by choosing a different view (Figure 15-3), you can make the results look quite a bit like the command-line version, albeit much prettier (Figure 15-4). The point here is that whichever interface you use, the information available to you is essentially the same.

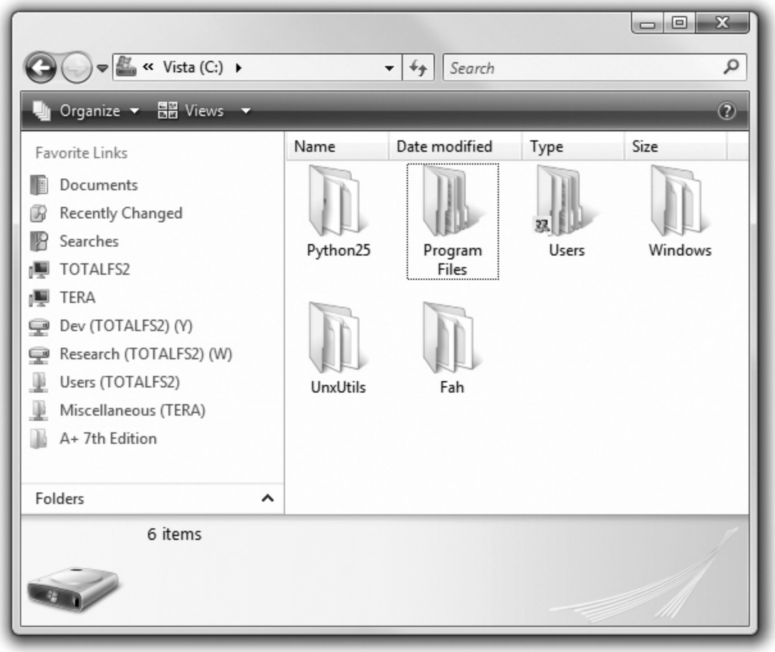


Figure 15-2 Contents of C: in Computer—Icon view

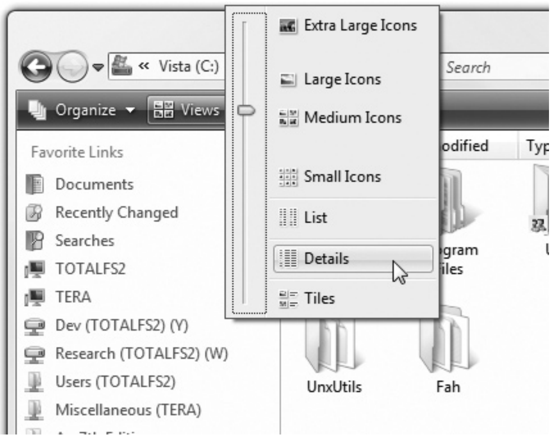
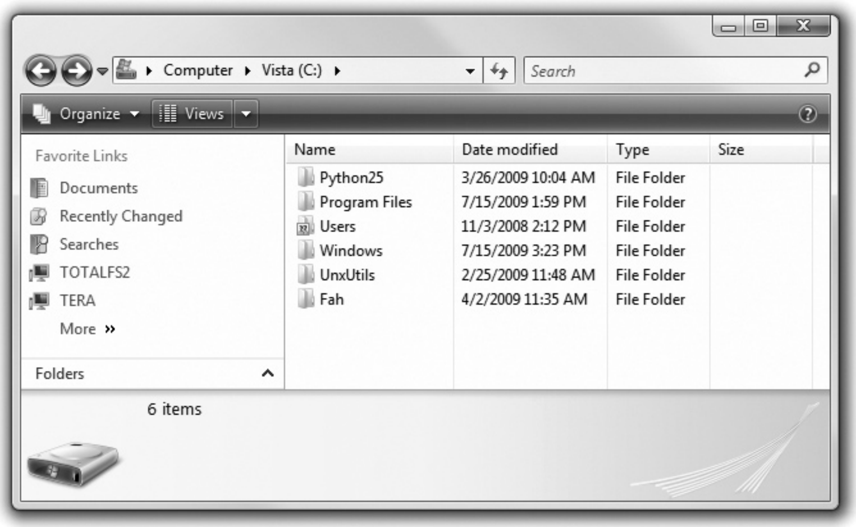


Figure 15-3 Selecting Details view in Computer

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!



Name	Date modified	Type	Size
Python25	3/26/2009 10:04 AM	File Folder	
Program Files	7/15/2009 1:59 PM	File Folder	
Users	11/3/2008 2:12 PM	File Folder	
Windows	7/15/2009 3:23 PM	File Folder	
UnxUtils	2/25/2009 11:48 AM	File Folder	
Fah	4/2/2009 11:35 AM	File Folder	

Figure 15-4 Contents of C: in Computer—Details view

One easy way to access the command-line interface in Windows 2000 or XP is by using the *Run dialog box*. Click the Start button, and then select Run. If you're using Windows 2000 or Windows XP, type **CMD** or **COMMAND** and press the ENTER key (Figure 15-5). If you are using Vista, you access the command-line interface through the Start menu Search box 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 | Accessories, is a link to the command-line interface. In Windows 2000, XP, and Vista, it's called command prompt. These links, just like the Run dialog box, pull up a nice command line-interface window (Figure 15-6). If you are displaying the command-line interface

Figure 15-5 Type **CMD** in the Run dialog box to open a command-line window.

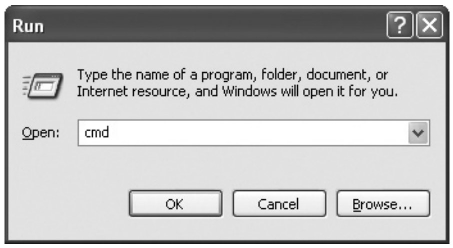
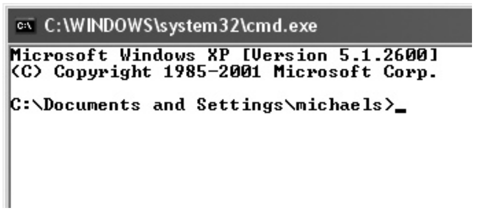
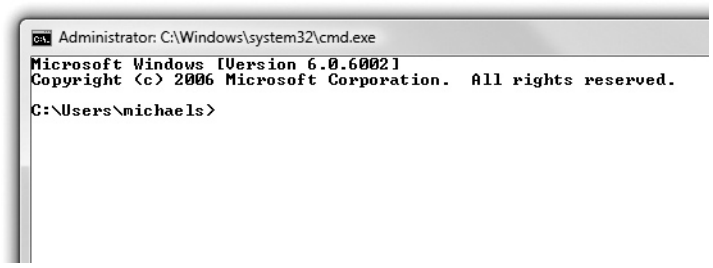

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

Figure 15-6 The command line-interface window with a C:\ prompt



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\michaels>_
```

in Windows Vista, notice that it uses a newer version number and copyright date. Also notice that the default user profile directory is C:\Users\User name rather than C:\Documents and Settings*User name* as in previous operating systems (shown in Figure 15-7). To close the command line-interface window, you can either click the Close box, as on any other window, or simply type **EXIT** at any command prompt and press ENTER.



```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\michaels>
```

Figure 15-7 The Windows Vista command-line interface window

If you attempt to enter a command at the Windows Vista command prompt that requires elevated or administrative privileges, you receive a UAC "Windows needs your permission to continue" dialog box (you'll learn more about UAC in the next chapter, "Securing Windows Resources"). 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.

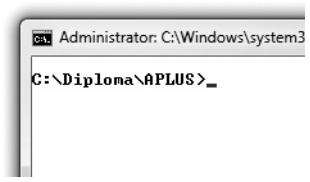
NOTE You can also create an administrator shortcut to the Windows Vista command prompt by right-clicking the Desktop, Select New | Shortcut. Then for the location of the item, type **CMD** and click Next. Type **CMD** to name the shortcut and click Finish. Your shortcut appears on the Desktop. Next, 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 command-prompt shortcut that will always run with administrative privileges.

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 15-8, 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.



```
C:\Diploma\APLUS>_
```

Figure 15-8 Command prompt indicating focus on the C:\Diploma\APLUS\ folder

NOTE You can hold down the F5 or F8 key during boot-up to access the Windows 2000, Windows XP, or Windows Vista 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. This is not the old-style command prompt–only interface!

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 9*x*, 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 using NTFS works almost exactly the same way as LFNs.

Whether you're running an ancient DOS system or the latest version of Windows Vista, 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 .DOC (.DOCX for Microsoft Office Word 2007), while WordPerfect uses .WPD and PowerPoint uses .PPT (.PPTX for Microsoft Office PowerPoint 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 15-9 shows a folder with two identical image files. The one on top shows the Photoshop icon, which is the program Windows will use to open that file; the one on the bottom shows a generic icon because I deleted the extension. Windows GUI doesn't show file extensions by default. Figure 15-10 shows the contents of that same folder from the command line.

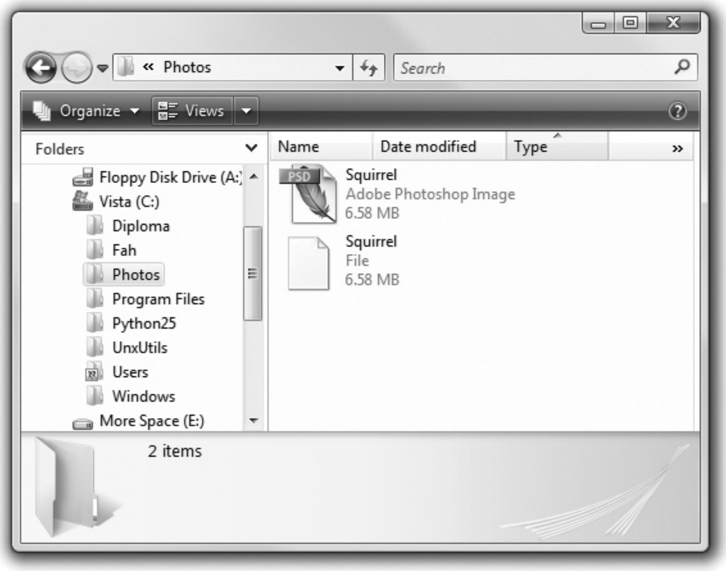
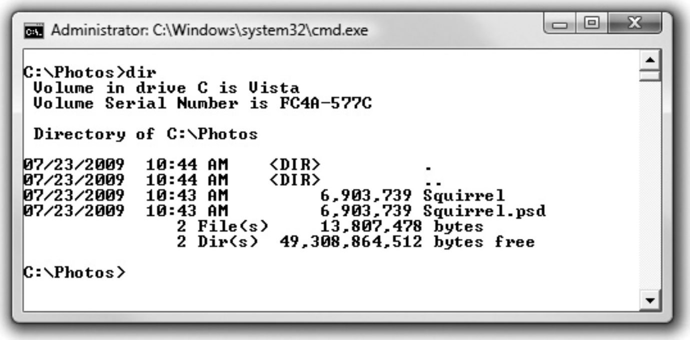


Figure 15-9 What kind of file is the one on the lower right?

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

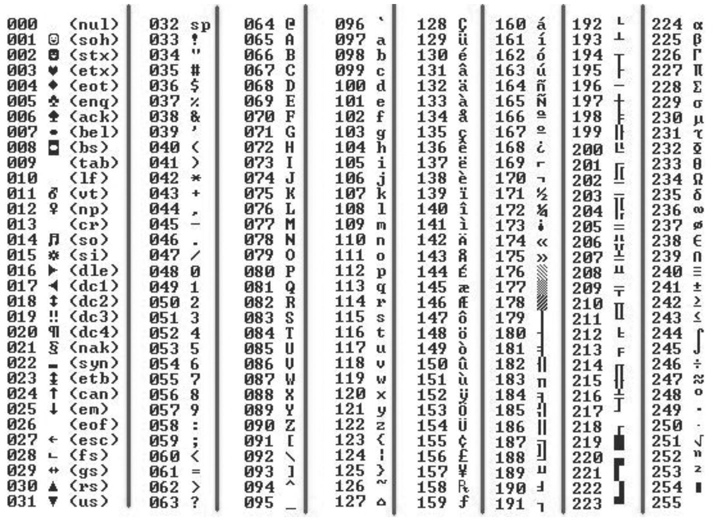


```
Administrator: C:\Windows\system32\cmd.exe

C:\Photos>dir
 Volume in drive C is Vista
 Volume Serial Number is FC4A-577C

 Directory of C:\Photos

07/23/2009  10:44 AM    <DIR>          .
07/23/2009  10:44 AM    <DIR>          ..
07/23/2009  10:43 AM         6,903,739 Squirrel
07/23/2009  10:43 AM         6,903,739 Squirrel.psd
               2 File(s)     13,807,478 bytes
               2 Dir(s)  49,308,864,512 bytes free

C:\Photos>
```

Figure 15-10 One file has no extension.

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 (Figure 15-11). 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 sends the letter of that key to the PC in ASCII code. Even the monitor outputs in ASCII when you are running DOS.



000		<nul>	032	sp	064	@	096	`	128	Ç	160	á	192	└	224	α
001	☺	<soh>	033	!	065	A	097	a	129	ü	161	í	193	┴	225	ß
002	☻	<stx>	034	"	066	B	098	b	130	é	162	ó	194	┬	226	Γ
003	♥	<etx>	035	#	067	C	099	c	131	â	163	ú	195	├	227	π
004	♦	<eot>	036	$	068	D	100	d	132	ä	164	ñ	196	─	228	Σ
005	♣	<enq>	037	%	069	E	101	e	133	à	165	Ñ	197	┼	229	σ
006	♠	<ack>	038	&	070	F	102	f	134	å	166	ª	198	╞	230	µ
007	•	<bel>	039	'	071	G	103	g	135	ç	167	º	199	╟	231	τ
008	◘	<bs>	040	(	072	H	104	h	136	ê	168	¿	200	╚	232	Φ
009		<tab>	041	)	073	I	105	i	137	ë	169	⌐	201	╔	233	Θ
010		<lf>	042	*	074	J	106	j	138	è	170	¬	202	╩	234	Ω
011	♂	<vt>	043	+	075	K	107	k	139	ï	171	½	203	╦	235	δ
012	♀	<np>	044	,	076	L	108	l	140	î	172	¼	204	╠	236	∞
013		<cr>	045	-	077	M	109	m	141	ì	173	¡	205	═	237	φ
014	♫	<so>	046	.	078	N	110	n	142	Ä	174	«	206	╬	238	ε
015	☼	<si>	047	/	079	O	111	o	143	Å	175	»	207	╧	239	∩
016	►	<dle>	048	0	080	P	112	p	144	É	176	░	208	╨	240	≡
017	◄	<dc1>	049	1	081	Q	113	q	145	æ	177	▒	209	╤	241	±
018	↕	<dc2>	050	2	082	R	114	r	146	Æ	178	▓	210	╥	242	≥
019	‼	<dc3>	051	3	083	S	115	s	147	ô	179	│	211	╙	243	≤
020	¶	<dc4>	052	4	084	T	116	t	148	ö	180	┤	212	╘	244	⌠
021	§	<nak>	053	5	085	U	117	u	149	ò	181	╡	213	╒	245	⌡
022	▬	<syn>	054	6	086	V	118	v	150	û	182	╢	214	╓	246	÷
023	↨	<etb>	055	7	087	W	119	w	151	ù	183	╖	215	╫	247	≈
024	↑	<can>	056	8	088	X	120	x	152	ÿ	184	╕	216	╪	248	°
025	↓	<em>	057	9	089	Y	121	y	153	Ö	185	╣	217	┘	249	∙
026		<eof>	058	:	090	Z	122	z	154	Ü	186	║	218	┌	250	·
027	←	<esc>	059	;	091	[	123	{	155	¢	187	╗	219	█	251	√
028	∟	<fs>	060	<	092	\	124	¦	156	£	188	╝	220	▄	252	ⁿ
029	↔	<gs>	061	=	093	]	125	}	157	¥	189	╜	221	▌	253	²
030	▲	<rs>	062	>	094	^	126	~	158	₧	190	╛	222	▐	254	■
031	▼	<us>	063	?	095	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Figure 15-11 ASCII characters

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 exactly the same as ASCII characters, making for easy backward compatibility.

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. Here's how you do it.

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 15-10, 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 *¿*.
3. If you have access to the Internet, surf over to the Tech Forums (www.totalsem.com/techforum/index.php) and say howdy. Include some Unicode in your post, of course!

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, especially in Windows 2000, Windows XP, and Windows Vista.

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 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 2000, XP, and Vista enable you to change the default lettering for drives, so you're likely to see all sorts of lettering schemes. On top of that, Windows 2000, XP, and Vista let you mount a hard drive as a volume in another drive.

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.

NOTE 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."

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 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

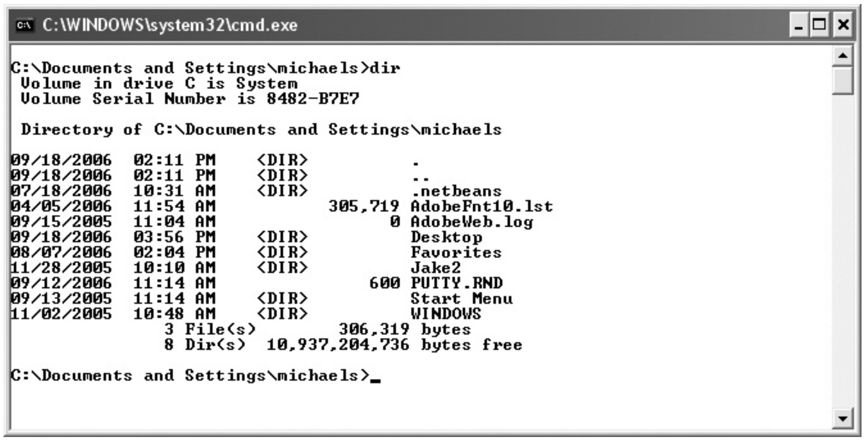
```
[command] [switches] [target (if any)]
```

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 a **/?** to get help.

DIR Command

The *DIR command* shows you the contents of the directory where the prompt is focused. DIR is used 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 2000/XP will look like this: `C:\Documents and Settings\username>`. By typing in **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 15-12.

NOTE Some commands give you the same result whether you include spaces or not. DIR/P and DIR /P, for example, provide 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.



```
C:\Documents and Settings\michaels>dir
 Volume in drive C is System
 Volume Serial Number is 8482-B7E7

 Directory of C:\Documents and Settings\michaels

09/18/2006  02:11 PM    <DIR>          .
09/18/2006  02:11 PM    <DIR>          ..
07/18/2006  10:31 AM    <DIR>          .netbeans
04/05/2006  11:54 AM           305,719 AdobeFnt10.lst
09/15/2005  11:04 AM                 0 AdobeWeb.log
09/18/2006  03:56 PM    <DIR>          Desktop
08/07/2006  02:04 PM    <DIR>          Favorites
11/28/2005  10:10 AM    <DIR>          Jake2
09/12/2006  11:14 AM               600 PUTTY.RND
09/13/2005  11:14 AM    <DIR>          Start Menu
11/02/2005  10:48 AM    <DIR>          WINDOWS
               3 File(s)        306,319 bytes
               8 Dir(s)  10,937,204,736 bytes free

C:\Documents and Settings\michaels>_
```

Figure 15-12 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 15-12! 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.

NOTE Extra text typed after a command to modify its operation, such as the /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**.

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

```
09/04/2008   05:51 PM          63,664 bambi.jpg
```

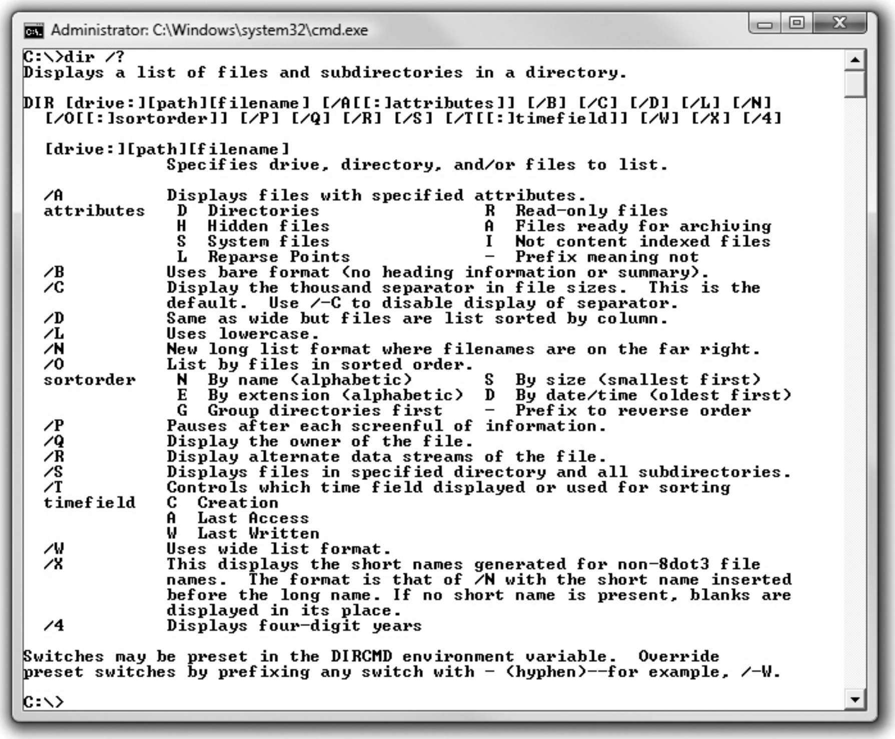
All of these entries are files. The DIR command lists the creation date, creation time, file size in bytes, filename, and extension. Any entries that look like this are folders:

```
12/31/2004  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:

```
08/06/2006  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 15-13, which lists all possible switches for the command.



Administrator: C:\Windows\system32\cmd.exe

```
C:\>dir /?
Displays a list of files and subdirectories in a directory.

DIR [drive:][path][filename] [/A[[:]attributes]] [/B] [/C] [/D] [/L] [/N]
  [/O[[:]sortorder]] [/P] [/Q] [/R] [/S] [/T[[:]timefield]] [/W] [/X] [/4]

  [drive:][path][filename]
              Specifies drive, directory, and/or files to list.

  /A          Displays files with specified attributes.
  attributes   D  Directories                R  Read-only files
               H  Hidden files               A  Files ready for archiving
               S  System files               I  Not content indexed files
               L  Reparse Points             -  Prefix meaning not
  /B          Uses bare format (no heading information or summary).
  /C          Display the thousand separator in file sizes.  This is the
              default.  Use /-C to disable display of separator.
  /D          Same as wide but files are list sorted by column.
  /L          Uses lowercase.
  /N          New long list format where filenames are on the far right.
  /O          List by files in sorted order.
  sortorder    N  By name (alphabetic)       S  By size (smallest first)
               E  By extension (alphabetic)  D  By date/time (oldest first)
               G  Group directories first    -  Prefix to reverse order
  /P          Pauses after each screenful of information.
  /Q          Display the owner of the file.
  /R          Display alternate data streams of the file.
  /S          Displays files in specified directory and all subdirectories.
  /T          Controls which time field displayed or used for sorting
  timefield   C  Creation
              A  Last Access
              W  Last Written
  /W          Uses wide list format.
  /X          This displays the short names generated for non-8dot3 file
              names.  The format is that of /N with the short name inserted
              before the long name. If no short name is present, blanks are
              displayed in its place.
  /4          Displays four-digit years

Switches may be preset in the DIRCMD environment variable.  Override
preset switches by prefixing any switch with - (hyphen)--for example, /-W.

C:\>
```

Figure 15-13 Typing **DIR /?** lists all possible switches for DIR command

Typing any command followed by a /? 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 too 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 15-13.

Directories: 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.

NOTE 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!

To return to the root directory, type **CD** and press ENTER. You can use the CD command to point DOS 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 DOS 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 DOS 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 DOS 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 floppy disk and using the CD command to point to its drive. Do the same with an optical disc. Type **DIR** to see the contents of the floppy or optical disc. Type **CD** to move the focus to any folders on the floppy or 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

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:\

07/12/2006  04:46 AM    <DIR>          Documents and Settings
06/04/2008  10:22 PM    <DIR>          STEAM
09/11/2006  11:32 AM    <DIR>          NVIDIA
08/06/2007  02:28 PM    <JUNCTION>     Other Drive
09/14/2006  11:11 AM    <DIR>          Program Files
09/12/2006  08:32 PM                21 statusclient.log
07/31/2005  10:40 PM               153 systemscandata.txt
03/13/2006  09:54 AM         1,111,040 t3h0
04/21/2006  04:19 PM    <DIR>          temp
07/12/2006  10:18 AM    <DIR>          WINDOWS
            3 file(s)       1,111,214  bytes
                         294,182,881,834   bytes free
```

What about uppercase and lowercase? Windows supports both, but it interprets all commands as uppercase. 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

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. RD 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."

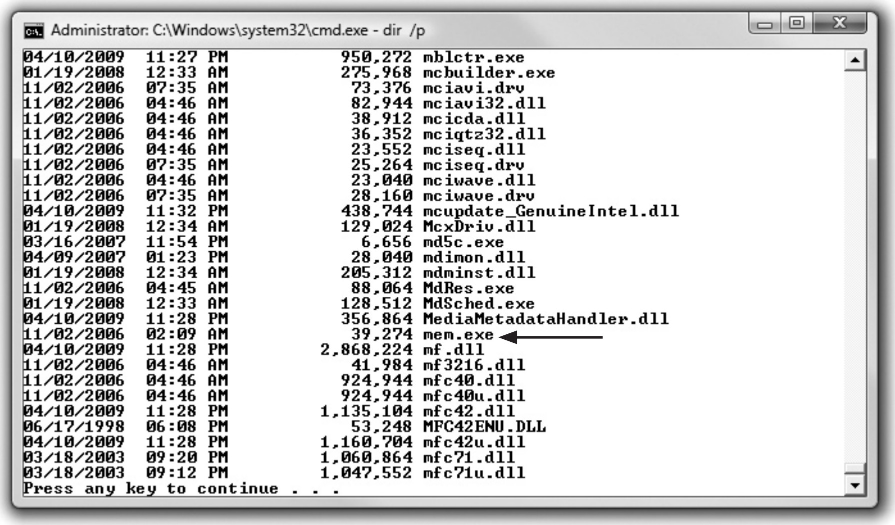
Let's delete the STEAM and GAMES directories with RD followed by the /S switch. Because the STEAM directory is in the root directory, point to the root directory with CD\. Now execute the command **RD C:\STEAM /S**. In a rare display of mercy, Windows responds with the following:

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

Press the Y key and both C:\STEAM and C:\STEAM\GAMES are eliminated.

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:\WINNT\System32 or 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 (Figure 15-14).



```
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 mciavi.drv
11/02/2006  04:46 AM            82,944 mciavi32.dll
11/02/2006  04:46 AM            38,912 mcicda.dll
11/02/2006  04:46 AM            36,352 mciqtz32.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 mciwave.dll
11/02/2006  07:35 AM            28,160 mciwave.drv
04/10/2009  11:32 PM           438,744 mcupdate_GenuineIntel.dll
01/19/2008  12:34 AM           129,024 McxDriv.dll
03/16/2007  11:54 PM             6,656 md5c.exe
04/09/2007  01:23 PM            28,040 mdimon.dll
01/19/2008  12:34 AM           205,312 mdminst.dll
11/02/2006  04:45 AM            88,064 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 MFC42ENU.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 15-14 MEM.EXE displayed in the System32 folder

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 (Figure 15-15). 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.

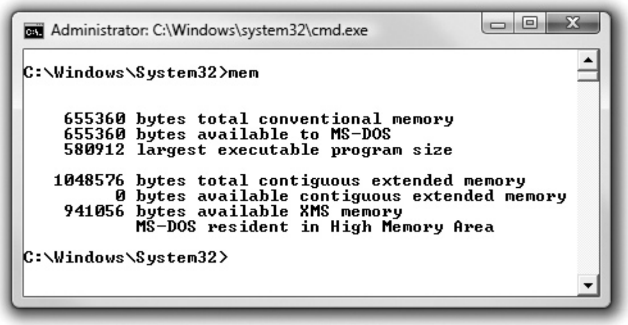


Figure 15-15 Running MEM in Windows Vista

```
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>
```

NOTE 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. The book discusses these task-specific tools in the chapters that reflect their task. Chapter 23, "Local Area Networking," goes into detail on the versatile and powerful NET command, for example. You'll read about the CONVERT command in Chapter 26, "Securing Computers." I couldn't resist throwing in two of the more interesting tools, COMPACT and CIPHER, in the Beyond A+ section of this chapter.

Working with Files

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

```
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, "Understanding Windows," when you had to make changes to the folder options in My Computer to see NTLDR, NTDETECT.COM, and other 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) 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 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.

ATTRIB.EXE is an external command-line program you can use 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, 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 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 AILog.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 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.

It's important for you to know that everything you do at the command line affects the same files at the GUI level, so run through these steps.

1. In Windows XP, go to My Computer and create a folder in the root directory of your C: drive called TEST.
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.
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 and access the properties of each file. Any changes?

Wildcards

Visualize having 273 files in one directory. A few of these files have the extension .DOC, but most do not. You are looking only for files with the .DOC extension. Wouldn't it be nice to type the DIR command so that only the .DOC 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 systemscandata.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 followed by 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 (Figure 15-16). Even the tried-and-true method of seeking help by typing the command followed by /? doesn't give you the answer.

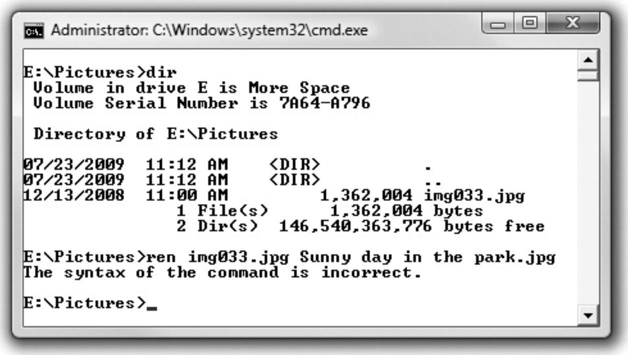


Figure 15-16
Rename failed me.

```
E:\Pictures>dir
 Volume in drive E is More Space
 Volume Serial Number is 7A64-A796

 Directory of E:\Pictures

07/23/2009  11:12 AM    <DIR>          .
07/23/2009  11:12 AM    <DIR>          ..
12/13/2008  11:00 AM         1,362,004 img033.jpg
               1 File(s)      1,362,004 bytes
               2 Dir(s)  146,540,363,776 bytes free

E:\Pictures>ren img033.jpg Sunny day in the park.jpg
The syntax of the command is incorrect.

E:\Pictures>_
```

You can use more complicated names by putting them in quotation marks. Figure 15-17 shows the same command that failed but now succeeds because of the quotation marks.

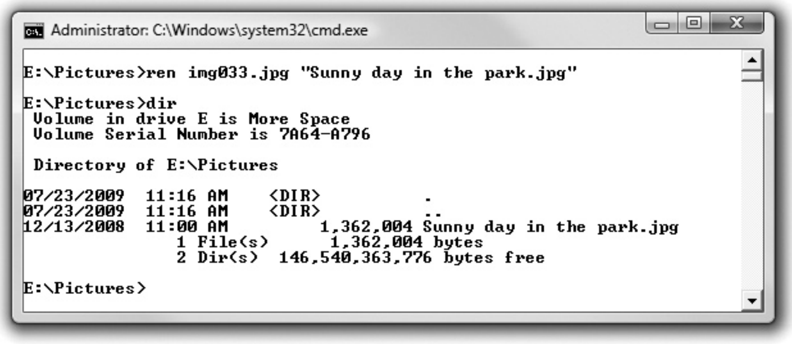


Figure 15-17
Success at last.

```
E:\Pictures>ren img033.jpg "Sunny day in the park.jpg"

E:\Pictures>dir
 Volume in drive E is More Space
 Volume Serial Number is 7A64-A796

 Directory of E:\Pictures

07/23/2009  11:16 AM    <DIR>          .
07/23/2009  11:16 AM    <DIR>          ..
12/13/2008  11:00 AM         1,362,004 Sunny day in the park.jpg
               1 File(s)      1,362,004 bytes
               2 Dir(s)  146,540,363,776 bytes free

E:\Pictures>
```

Deleting Files

To delete files, you use the *DEL* (or *ERASE*) *command*. DEL and ERASE are 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*. DEL deletes files, but it will not remove directories. Use RD to delete directories.

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 files 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 files.
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.)

XCOPY works similar 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.

Their power and utility make the DEL, COPY/MOVE, and XCOPY 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 keystroke, for it may spell your doom. Think twice and execute the command once. The data you save may be yours!

Working with Batch Files

Batch files are nothing more than text files that store a series of commands, one command per line. The only thing that differentiates a batch file from any other text file is the .BAT extension. Take a look at Figure 15-18, and note the unique icon used for a batch file compared to the icon for a regular text file.

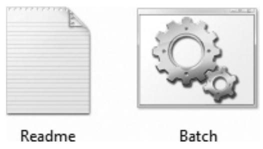


Figure 15-18 Text and batch file icons

You can create and edit batch files by using any text editor program—good old Notepad is often the tool of choice. This is the command-line chapter, though, so let's dust off the ancient but still important Edit program—it comes with every version of Windows—and use it to create and edit batch files.

Get to a command prompt on any Windows system and use the CD\ command to get to the root directory (use C: to get to the C: drive if you're not on the C: drive by default). From there, type **EDIT** at the command prompt to see the Edit program's interface (Figure 15-19).

Now that you've started Edit, type in the two commands as shown in Figure 15-20. Make sure they look exactly the same as the lines in Figure 15-20.

Great! You have just made your first batch file. All you need to do now is save it with some name—the name doesn't matter, but this example uses FIRST as the filename. It is imperative, however, that you use the extension .BAT. Even though you could probably figure this out on your own later, do it now. Hold down the ALT key to activate

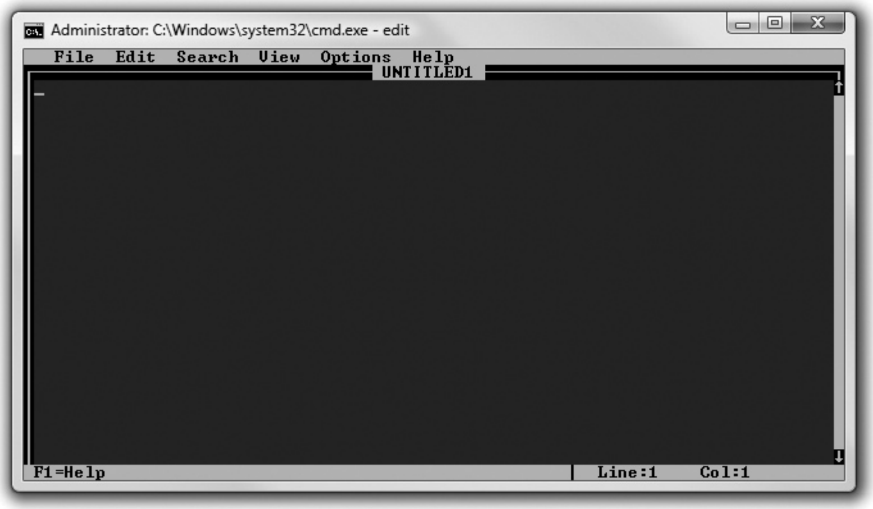
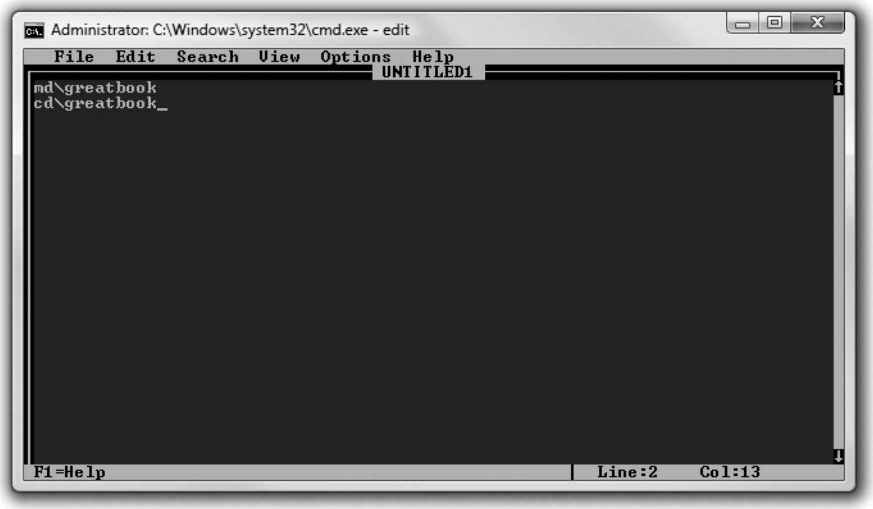

Administrator: C:\Windows\system32\cmd.exe - edit
File Edit Search View Options Help
UNTITLED1
F1=Help
Line:1 Col:1

Figure 15-19 Edit interface


Administrator: C:\Windows\system32\cmd.exe - edit
File Edit Search View Options Help
UNTITLED1
F1=Help
Line:2 Col:13

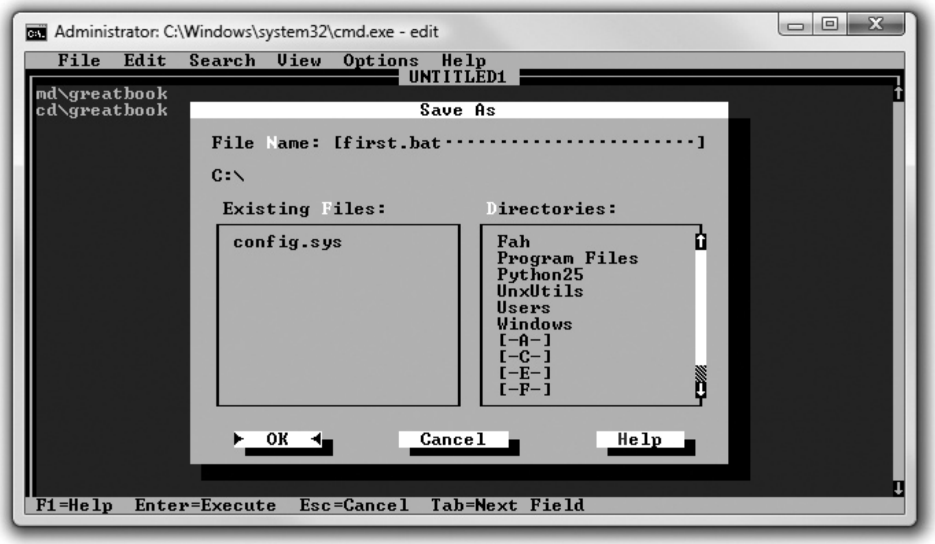
```
md\greatbook
cd\greatbook_
```

Figure 15-20 Edit with two commands

the menu. Press the F (File) key. Then press S (Save). Type in the name **first.bat** as shown in Figure 15-21. Press ENTER and the file is now saved.

Now that you've saved the file, exit the Edit program by pressing ALT-F and then pressing X (Exit). You're back at the command prompt. Go ahead and run the program by typing **FIRST** and pressing ENTER. Your results should look something like Figure 15-22.

Super! The batch file created a folder and moved the prompt to focus on that folder. Don't run the First batch file again or you'll create another folder inside the first one.


Administrator: C:\Windows\system32\cmd.exe - edit
File Edit Search View Options Help
UNTITLED1
Save As
Existing Files:
Directories:
OK
Cancel
Help
F1=Help Enter=Execute Esc=Cancel Tab=Next Field

```
md\greatbook
cd\greatbook
```

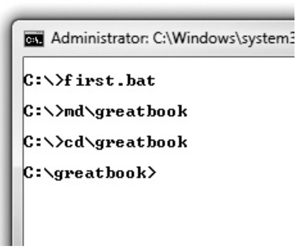
```
File Name: [first.bat.......................]
C:\
```

Existing Files: config.sys

Directories: Fah, Program Files, Python25, UnxUtils, Users, Windows, [-A-], [-C-], [-E-], [-F-]

Figure 15-21 Saving the batch file

Figure 15-22 Running the batch file

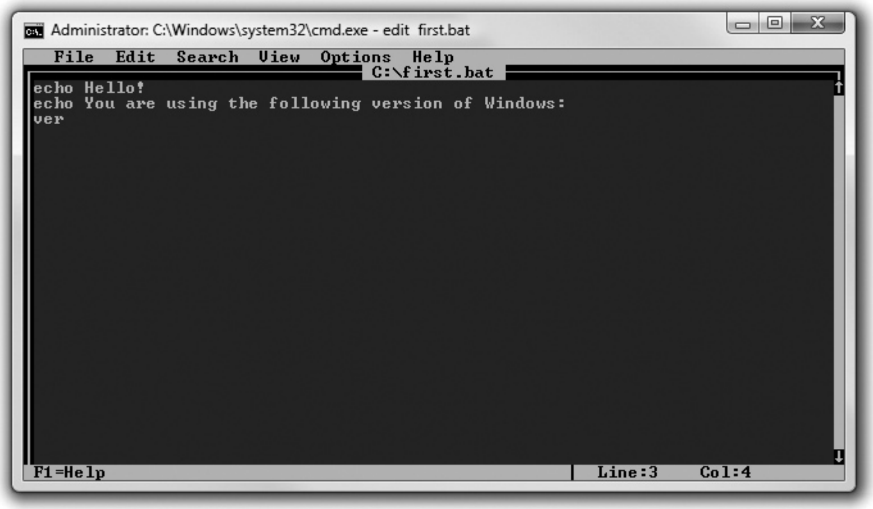


```
C:\>first.bat

C:\>md\greatbook

C:\>cd\greatbook

C:\greatbook>
```

Let's now get back to the root directory of C: and edit the FIRST.BAT file again. This time type **EDIT FIRST.BAT** and press ENTER. The batch file will come up, ready to edit. Now change the batch file to look like Figure 15-23. Use the ARROW keys to move your cursor and the DELETE key to delete.

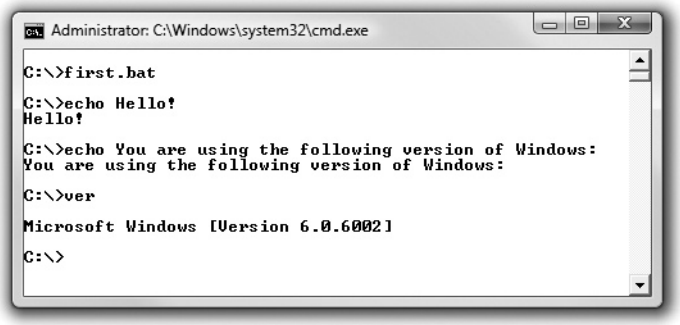
NOTE Most of the keyboard shortcuts used in WordPad, Word, and so on, were first used in the Edit program. If you know keyboard shortcuts for WordPad or Word, many will work in Edit.

The VER command shows the current version of Windows. The ECHO command tells the batch file to put text on the screen. Run the batch file, and it should look like Figure 15-24.


Administrator: C:\Windows\system32\cmd.exe - edit first.bat
File Edit Search View Options Help
C:\first.bat
F1=Help
Line:3 Col:4

```
echo Hello!
echo You are using the following version of Windows:
ver
```

Figure 15-23 New version of FIRST.BAT



```
C:\>first.bat

C:\>echo Hello!
Hello!

C:\>echo You are using the following version of Windows:
You are using the following version of Windows:

C:\>ver

Microsoft Windows [Version 6.0.6002]

C:\>
```

Figure 15-24 Running VER to show the current version of Windows

Gee, that's kind of ugly. Try editing the FIRST.BAT file one more time and add the following line as the first line of the batch file:

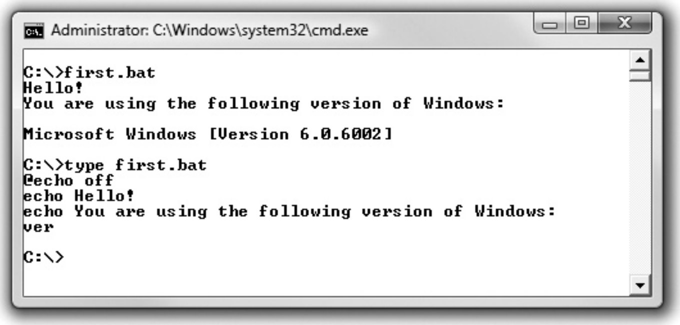
```
@echo off
```

Run FIRST.BAT again. It should look quite a bit nicer. The @echo off command tells the system not to show the command, just the result.

Sometimes you just want to look at a batch file. The TYPE command displays the contents of a text file on the screen, as shown in Figure 15-25.

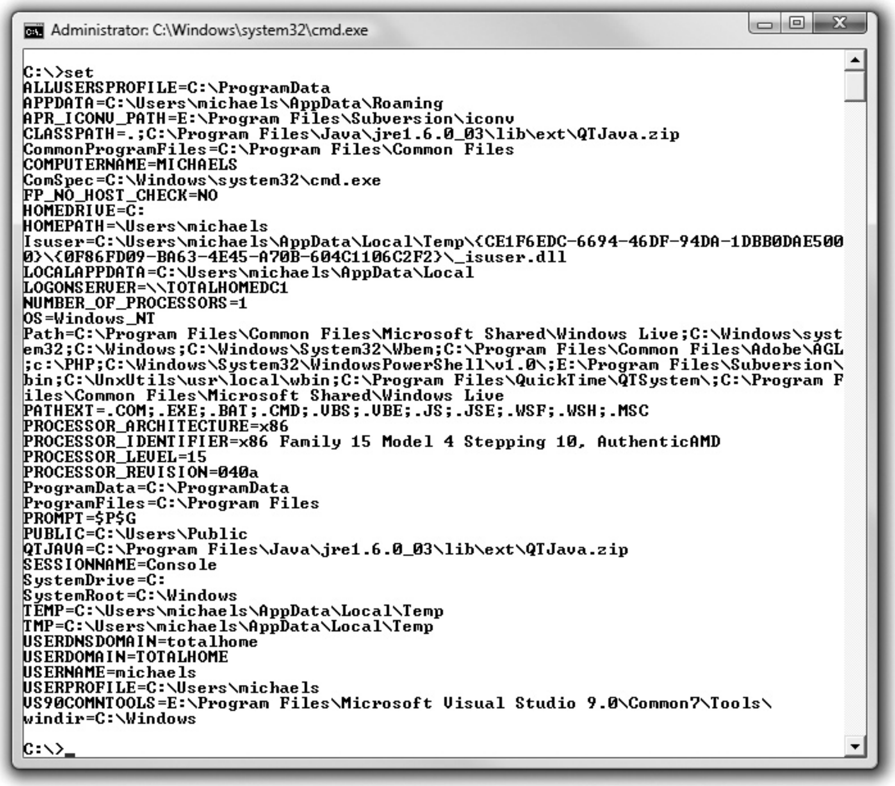
CAUTION Don't try using the TYPE command on anything other than a text file—the results will be unpredictable.

Figure 15-25 Using the TYPE command to see file contents



```
C:\>first.bat
Hello!
You are using the following version of Windows:

Microsoft Windows [Version 6.0.6002]

C:\>type first.bat
@echo off
echo Hello!
echo You are using the following version of Windows:
ver

C:\>
```

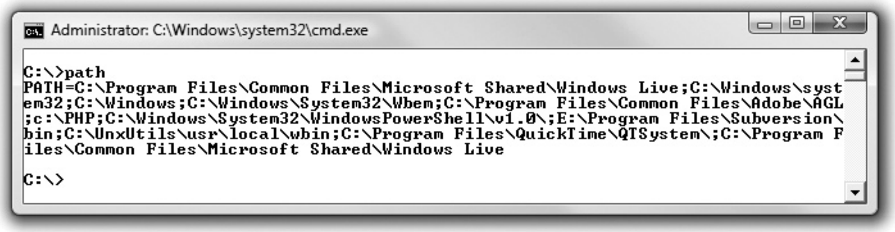
One of the more irritating aspects to batch files is that sometimes they don't work unless you run them in the folder in which they are stored. This is because of the path setting. Every time you open a command prompt, Windows loads a number of settings by default. You can see all of these settings by running the SET command. Figure 15-26 shows the results of running the SET command.



```
C:\>set
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\michaels\AppData\Roaming
APR_ICONV_PATH=E:\Program Files\Subversion\iconv
CLASSPATH=.;C:\Program Files\Java\jre1.6.0_03\lib\ext\QTJava.zip
CommonProgramFiles=C:\Program Files\Common Files
COMPUTERNAME=MICHAELS
ComSpec=C:\Windows\system32\cmd.exe
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
HOMEPATH=\Users\michaels
Isuser=C:\Users\michaels\AppData\Local\Temp\{CE1F6EDC-6694-46DF-94DA-1DBB0DAE500
0}\{0F86FD09-BA63-4E45-A70B-604C1106C2F2}\_isuser.dll
LOCALAPPDATA=C:\Users\michaels\AppData\Local
LOGONSERVER=\\TOTALHOMEDC1
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Path=C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Windows\syst
em32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\Common Files\Adobe\AGL
;c:\PHP;C:\Windows\System32\WindowsPowerShell\v1.0\;E:\Program Files\Subversion\
bin;C:\UnxUtils\usr\local\wbin;C:\Program Files\QuickTime\QTSystem\;C:\Program F
iles\Common Files\Microsoft Shared\Windows Live
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 15 Model 4 Stepping 10, AuthenticAMD
PROCESSOR_LEVEL=15
PROCESSOR_REVISION=040a
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
PROMPT=$P$G
PUBLIC=C:\Users\Public
QTJAVA=C:\Program Files\Java\jre1.6.0_03\lib\ext\QTJava.zip
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Users\michaels\AppData\Local\Temp
TMP=C:\Users\michaels\AppData\Local\Temp
USERDNSDOMAIN=totalhome
USERDOMAIN=TOTALHOME
USERNAME=michaels
USERPROFILE=C:\Users\michaels
VS90COMNTOOLS=E:\Program Files\Microsoft Visual Studio 9.0\Common7\Tools\
windir=C:\Windows

C:\>_
```

Figure 15-26 Using the SET command to see settings

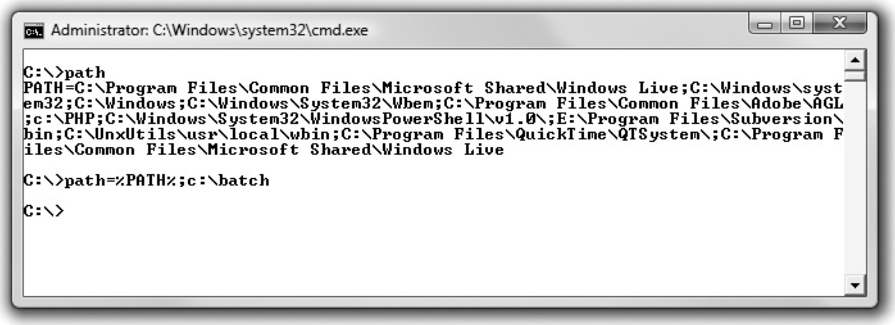
Don't worry about understanding everything the SET command shows you, but do notice a line that starts with `Path=`. This line tells Windows where to look for a program (or batch file) if you run a program that's not in your current folder. For example, let's say I make a folder called C:\batch to store all of my batch files. I can run the PATH command from the command prompt to see my current path (Figure 15-27).



```
C:\>path
PATH=C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Windows\syst
em32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\Common Files\Adobe\AGL
;c:\PHP;C:\Windows\System32\WindowsPowerShell\v1.0\;E:\Program Files\Subversion\
bin;C:\UnxUtils\usr\local\wbin;C:\Program Files\QuickTime\QTSystem\;C:\Program F
iles\Common Files\Microsoft Shared\Windows Live

C:\>
```

Figure 15-27 Using the PATH command to see the current path

I can then run the PATH command again, this time adding **%PATH%;C:\batch** (Figure 15-28). The %PATH% bit is a variable that represents what is currently in the path. By placing it before my batch folder, I am telling the path command to keep what is there and just add c:\batch. I can now place all of my batch files in this folder, and they will always work, no matter where I am in the system.



```
C:\>path
PATH=C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Windows\syst
em32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\Common Files\Adobe\AGL
;c:\PHP;C:\Windows\System32\WindowsPowerShell\v1.0\;E:\Program Files\Subversion\
bin;C:\UnxUtils\usr\local\wbin;C:\Program Files\QuickTime\QTSystem\;C:\Program F
iles\Common Files\Microsoft Shared\Windows Live

C:\>path=%PATH%;c:\batch

C:\>
```

Figure 15-28 Using PATH to add a folder

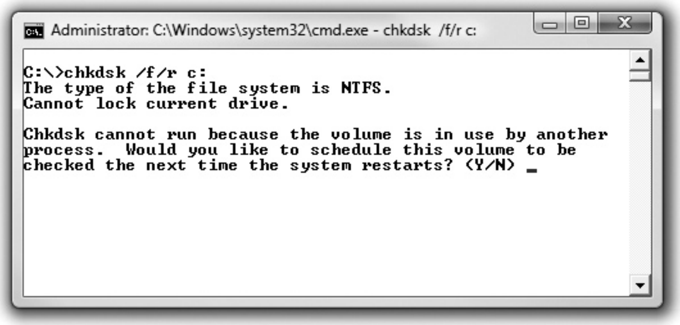
NOTE In Windows 2000 and XP, you can edit the BOOT.INI file by using the Edit program. Just make sure you use ATTRIB first to turn off the System and Hidden attributes! In Windows Vista, the boot configuration data (BCD) store contains boot configuration parameters and objects that control how the operating system starts. You use the bcdedit.exe command-line tool to add, delete, and edit the objects and entries stored in the BCD store.

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 220-702 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, and SFC.

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 (Figure 15-29).


Administrator: C:\Windows\system32\cmd.exe - chkdsk /f/r c:

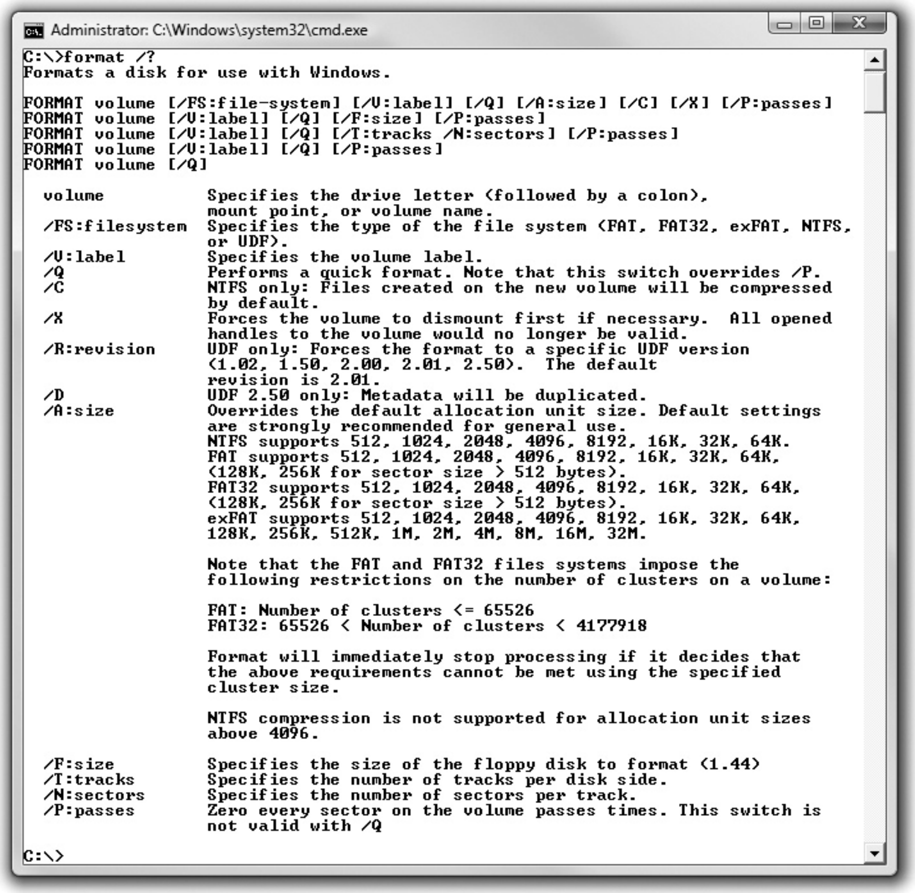
```
C:\>chkdsk /f/r c:
The type of the file system is NTFS.
Cannot lock current drive.

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 15-29 The CHKDSK /f /r utility and switches on a locked drive

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, is the process of wiping or preparing a disk to be partitioned 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 a myriad of 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 FORMAT and its available switches is simply to enter **FORMAT /?** from the command prompt. Your results should be similar to those displayed in Figure 15-30.



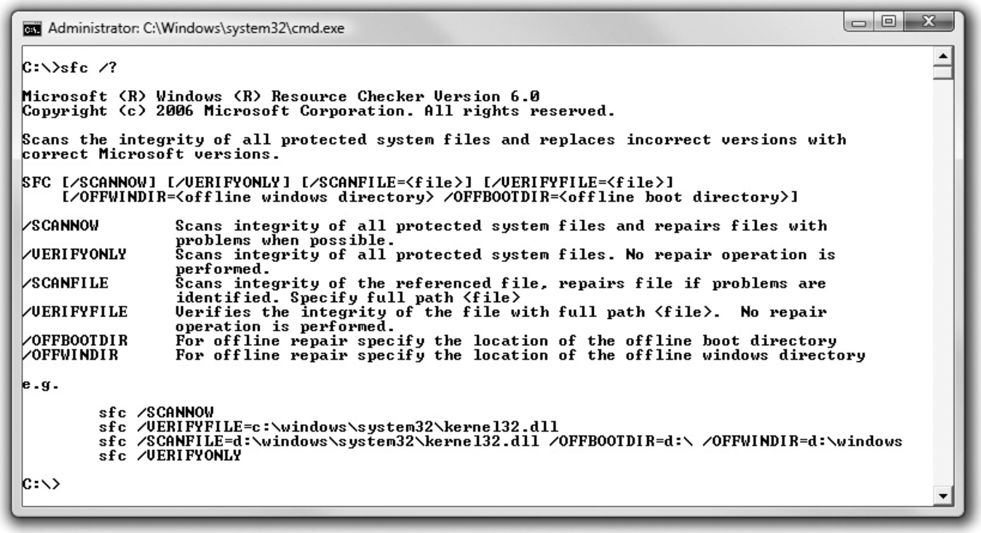
```
C:\>format /?
Formats a disk for use with Windows.

FORMAT volume [/FS:file-system] [/V:label] [/Q] [/A:size] [/C] [/X] [/P:passes]
FORMAT volume [/V:label] [/Q] [/F:size] [/P:passes]
FORMAT volume [/V:label] [/Q] [/T:tracks /N:sectors] [/P:passes]
FORMAT volume [/V: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).
  /V: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 files 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)
  /T: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 15-30 Using Format /? at the command prompt

Although the new CompTIA A+ exams focus primarily on 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.

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 /?** (Figure 15-31).



```
C:\>sfc /?

Microsoft (R) Windows (R) Resource Checker Version 6.0
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

Scans the integrity of all protected system files and replaces incorrect versions with
correct Microsoft versions.

SFC [/SCANNOW] [/VERIFYONLY] [/SCANFILE=<file>] [/VERIFYFILE=<file>]
    [/OFFWINDIR=<offline windows directory> /OFFBOOTDIR=<offline boot directory>]

/SCANNOW        Scans integrity of all protected system files and repairs files with
                problems when possible.
/VERIFYONLY     Scans integrity of all protected system files. No repair operation is
                performed.
/SCANFILE       Scans integrity of the referenced file, repairs file if problems are
                identified. Specify full path <file>
/VERIFYFILE     Verifies the integrity of the file with full path <file>.  No repair
                operation is performed.
/OFFBOOTDIR     For offline repair specify the location of the offline boot directory
/OFFWINDIR      For offline repair specify the location of the offline windows directory

e.g.

        sfc /SCANNOW
        sfc /VERIFYFILE=c:\windows\system32\kernel32.dll
        sfc /SCANFILE=d:\windows\system32\kernel32.dll /OFFBOOTDIR=d:\ /OFFWINDIR=d:\windows
        sfc /VERIFYONLY

C:\>
```

Figure 15-31 Checking SFC options with SFC /? at a command prompt

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.

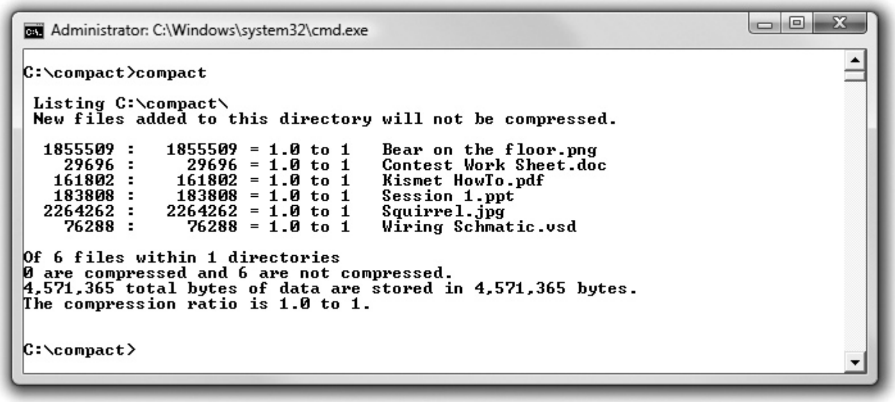
Compact and Cipher

Windows XP and Vista offer two cool commands at the command-line interface: COMPACT and CIPHER. COMPACT displays or alters the compression of files on NTFS partitions. CIPHER 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.

COMPACT

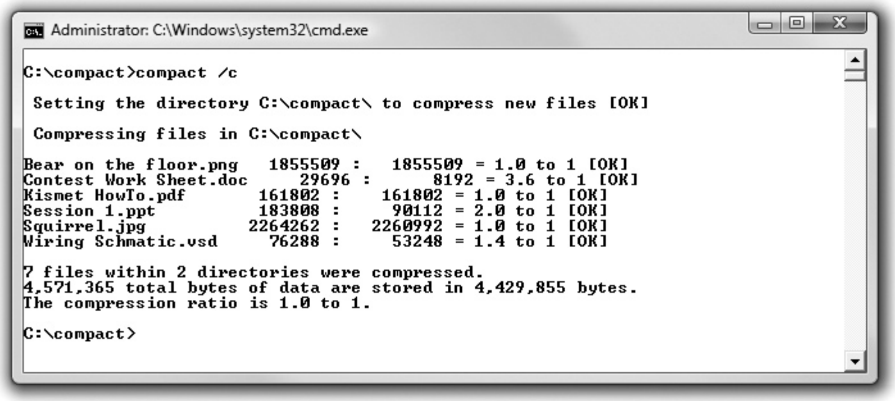
First let's try the COMPACT command. Figure 15-32 shows the result of entering the COMPACT command with no switches. It displays the compression status of the contents of a directory called compact on a system's C: drive. Notice that after the file listing, COMPACT helpfully tells you that 0 files are compressed and 6 files (all of them) are not compressed, with a total compression ratio of 1.0 to 1.



```
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 Schmatic.vsd

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>
```

Figure 15-32 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 15-33. Look closely at the listing. 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? COMPACT 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



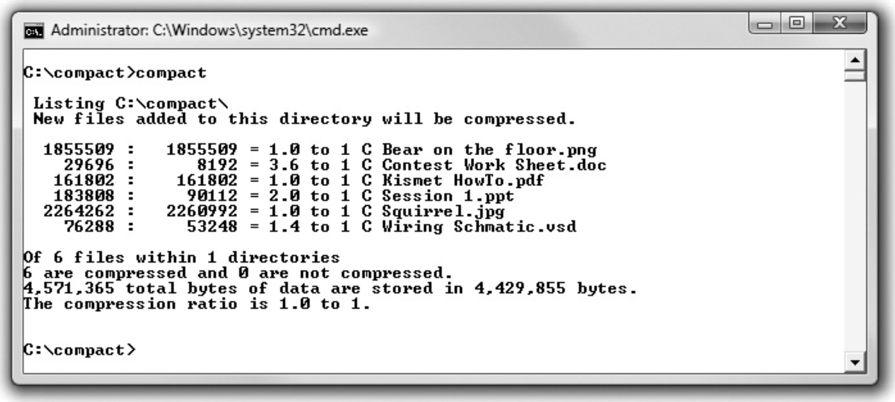
Administrator: C:\Windows\system32\cmd.exe

```
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 Schmatic.vsd      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 15-33 Typing **COMPACT /C** compresses the contents of the directory.

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 (Figure 15-34).

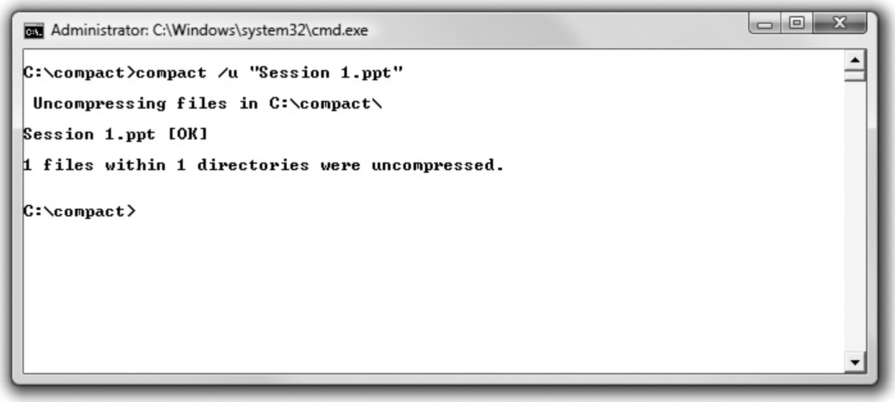


Administrator: C:\Windows\system32\cmd.exe

```
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 HowTo.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 Schmatic.vsd

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 15-34 The contents of C:\COMPACT have been compressed.

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 15-35. 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.



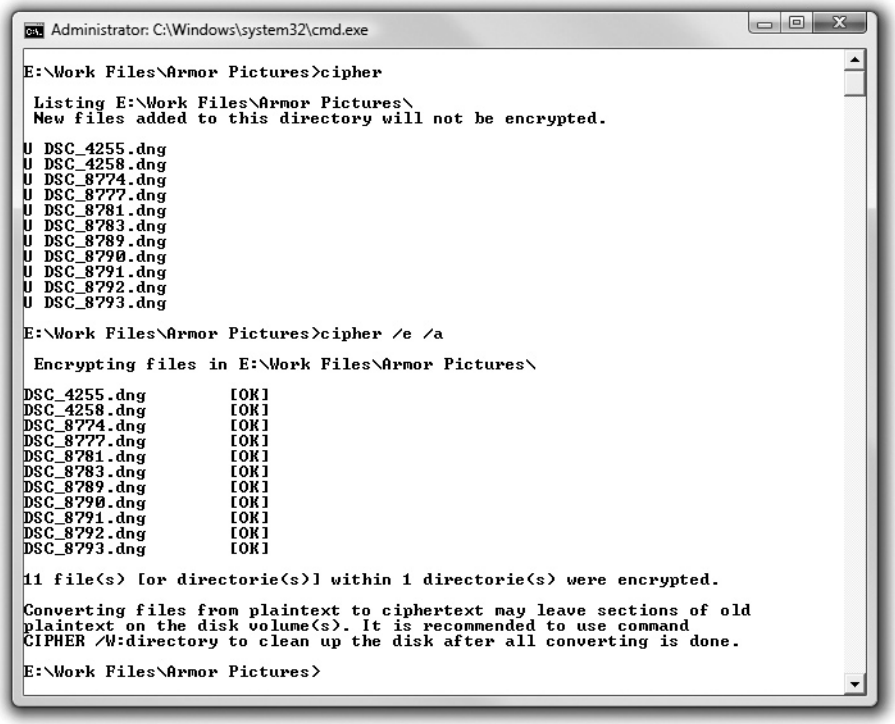
Administrator: C:\Windows\system32\cmd.exe

```
C:\compact>compact /u "Session 1.ppt"

 Uncompressing files in C:\compact\

Session 1.ppt [OK]

1 files within 1 directories were uncompressed.

C:\compact>
```

Figure 15-35 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 15-36 shows two steps in the process. Like the COMPACT command, the CIPHER command simply displays the current state of affairs when



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) [or 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>
```

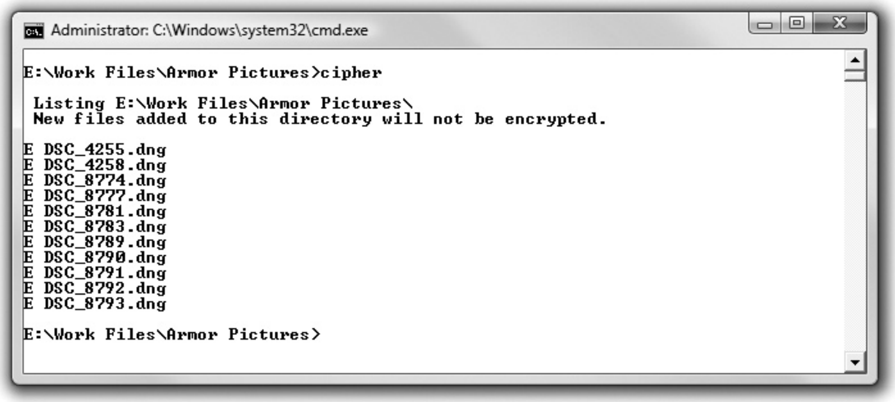
Figure 15-36 Typing **CIPHER /E /A** encrypts the contents of the directory.

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 15-36 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 15-37. Note that the *U* to the left of each filename has been replaced



```
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>
```

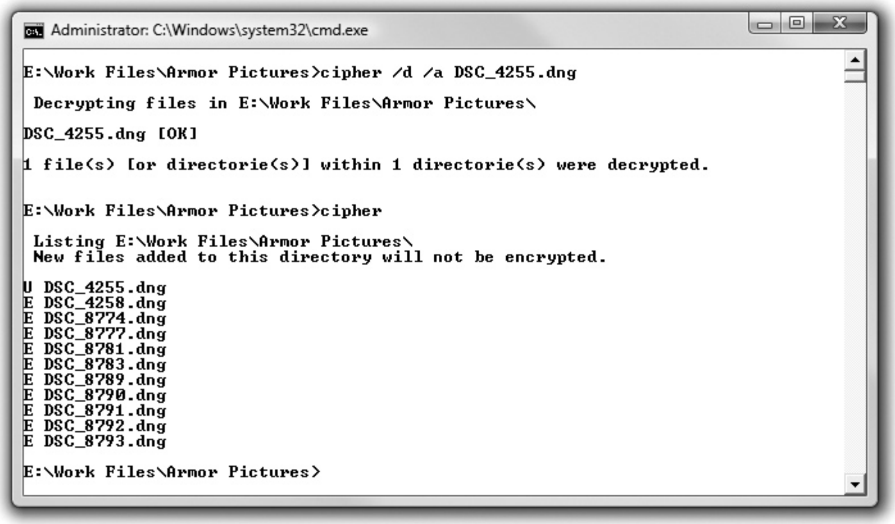
Figure 15-37 CIPHER command confirms that the files were encrypted.

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 15-38 shows the CIPHER command being used to decipher DSC_4255.dng, a single file.



```
E:\Work Files\Armor Pictures>cipher /d /a DSC_4255.dng

 Decrypting files in E:\Work Files\Armor Pictures\

DSC_4255.dng [OK]

1 file(s) [or directorie(s)] within 1 directorie(s) were decrypted.


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
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 15-38 Typing **CIPHER /D /A DSC_4255.dng** decrypts only that file.

Chapter Review Questions

1. The ASCII standard defines how many 8-bit characters?
   - A. 64
   - B. 256
   - C. 512
   - D. 64,000
2. Which of the following is the correct path for a file named YODA.TXT on the C: drive in a directory called JEDI that's in a directory called REBELS that's in the root directory?
   - A. C:\ROOT\JEDI\YODA.TXT
   - B. C:\JEDI\REBELS\YODA.TXT
   - C. C:\REBELS\JEDI\YODA.TXT
   - D. C:\ROOT\REBELS\JEDI\YODA.TXT
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. What command enables you to make a new directory in a Windows XP Professional system?
   - A. MF
   - B. MKFOL
   - C. MD
   - D. MAKEDIR
5. What command do you type at the Run dialog box to access the command-line interface in Windows XP?
   - A. CMD
   - B. CONAND
   - C. MSDOS
   - D. CP
6. 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
7. What types of characters are the asterisk (*) and the question mark (?)?
   - A. Wildcards
   - B. Optionals
   - C. Designators
   - D. Switches
8. What is the command to make MYFILE.TXT read-only?
   - A. ATTRIB +R MYFILE.TXT
   - B. ATTRIB –R MYFILE.TXT
   - C. READONLY MYFILE.TXT
   - D. MYFILE.TXT /READONLY
9. To learn the syntax of the DIR command, what can you type?
   - A. HELP DIR
   - B. DIR /?
   - C. DIR /HELP
   - D. Both A and B.

10. What is the command to quit the command-line interface?
    - **A.** EXIT
    - **B.** BYE
    - **C.** QUIT
    - **D.** STOP

Answers

1. **B.** The ASCII standard has 256 characters because that's all 8 bits can handle!
2. **C.** You'll find the YODA.TXT file in the C:\REBELS\JEDI\ folder.
3. **A.** You can use the *.* wildcard combination to affect every file in a particular folder.
4. **C.** MD enables you to make a directory or folder. You can also use the older form, MKDIR.
5. **A.** Use the CMD command in the Run dialog box to access a command line in Windows 2000/XP.
6. **B.** The REN command with the proper syntax—REN START.BAT HAMMER.BAT—will rename the file.
7. **A.** The asterisk and question mark characters are wildcards when used with command-line commands.
8. **A.** The command ATTRIB +R MYFILE.TXT will make MYFILE.TXT read-only.
9. **D.** To learn the ins and outs of any command-line program, type the command followed by the /? switch or else type HELP and then the command.
10. **A.** Type EXIT and press the ENTER key to bail out of a command-line interface in Windows.