Introduction to the Windows Filesystem

1.1 The Windows Directory Structure

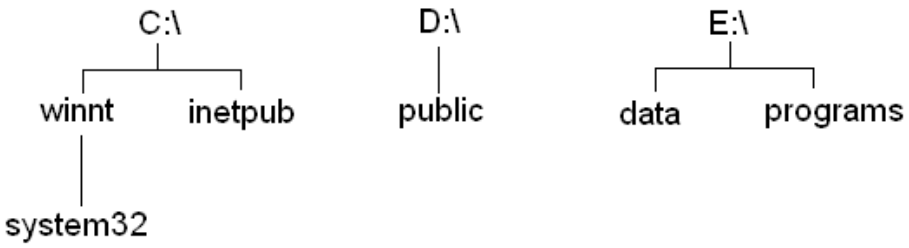
It is extremely vital that you master the usage of the Windows operating system in order to be a successful developer. Key to this is understanding how Windows stores files on it's filesystem, as well as how to traverse file paths both within the graphical desktop and the command-line interface (often called a **DOS prompt** or **shell**).

You can open a shell within Windows 7 by navigating to **Start > All Programs > Accessories > Command Prompt** (if you need to run the shell as Administrator, simply right-click the Command Prompt and choose **Run as administrator** instead). Alternatively, you could search for the **cmd** program and run it, by clicking **Start** and typing **cmd** in the Windows search box. :

The Windows directory structure is disk-based and given a drive letter – your first hard disk is C: (pronounced "C drive"). Additional disk drives, whether they be hard disks, DVDs, or USB flash drives are given D:, E:, F:, and so on when they are present in your system. A: and B: are not used today – they were used in the past for the first and second floppy drive respectively.

Since the average system has more than 100,000 files, each disk drive is normally subdivided into **directories** (also called **folders**), and files can be placed into directories for organization purposes.

Each drive has a root directory called \, so **C:** would be the root directory of **C:** (C drive) and **D:** would be the root directory of **D:** (D drive). Under the root directory, you'll find many subdirectories, each of which can contain many other subdirectories to form a hierarchy. For example, a sample Windows directory structure for three disk drives is seen below:



NOTE: Each disk drive must contain a **file system**, that contain the rules for writing files and directories by the operating system. The file system is created when you format the disk drive for the first time (this is normally done by the factory nowadays). USB flash drives normally contain the FAT32 file system, which is understood by every operating system (UNIX/Linux/Mac), but the disk drive that contain the Windows operating system (C:\) usually has the NTFS file system, since NTFS allows each directory and file to have permissions attached to them for users and groups on the system.

NOTE: Some people like to create partitions on hard disks. If you create three partitions on your first hard disk, and format each partition with a file system, each partition will appear as its own disk drive in Windows. In other words, you will have a C:\, D:\, and E:\ to represent the three partitions on your one hard disk.

To discuss the location of a directory or file, we can use **absolute pathnames** that list the full path from the drive and root directory, down to the actually directory or file. The absolute pathnames for each directory shown in the previous figure are:

C:\winnt\system32
C:\inetpub
D:\public
E:\data
E:\programs

If there was a file in the E:\data directory called oobla.txt, the absolute pathname to this file would be E:\data\oobla.txt.

NOTE: File names in Windows can be up to 255 characters long and typically have a three letter/digit extension that identifies the type of file. For example, letter.doc would be a Word document, whereas trees.jpg would be a JPEG graphic.

NOTE: Absolute and relative pathnames in Linux/UNIX/Mac are similar to those specified at a shell in Windows, however the default name delimiter is a forward slash (/) instead of a backslash (\), and drive letters are not used – instead the root directory of each disk drive is attached to a subdirectory under a single root (/) of the whole system.

Table 1-1 Common Windows Directories

C:\boot	Folder that contains the files used to start the system (hidden since Windows Vista)
C:\Documents and Settings	The folder that contains each user's home directory and user settings for Windows XP
C:\Documents and Settings\bob	Bob's home directory in Windows XP
C:\Users	The folder that contains each user's home directory and user settings for Windows Vista, Windows 7 and Windows 8
C:\Users\bob	Bob's home directory in Windows Vista, Windows 7 and Windows 8

C:\Program Files	The installed applications on the system
C:\Program Files (x86)	If your system is 64-bit, this folder contains 32-bit applications that are installed on the system
C:\Windows	Contains the files that comprise the Windows operating system

When a user logs into a Windows system, they are placed in their home directory. Following this, users may change their current directory with the **cd** (change directory) command. To change the current directory, simply specify the absolute pathname as an argument to the **cd** command. Alternatively, you can use a **relative pathname** as an argument. To understand relative pathnames, you must first understand that each directory has two special hidden files:

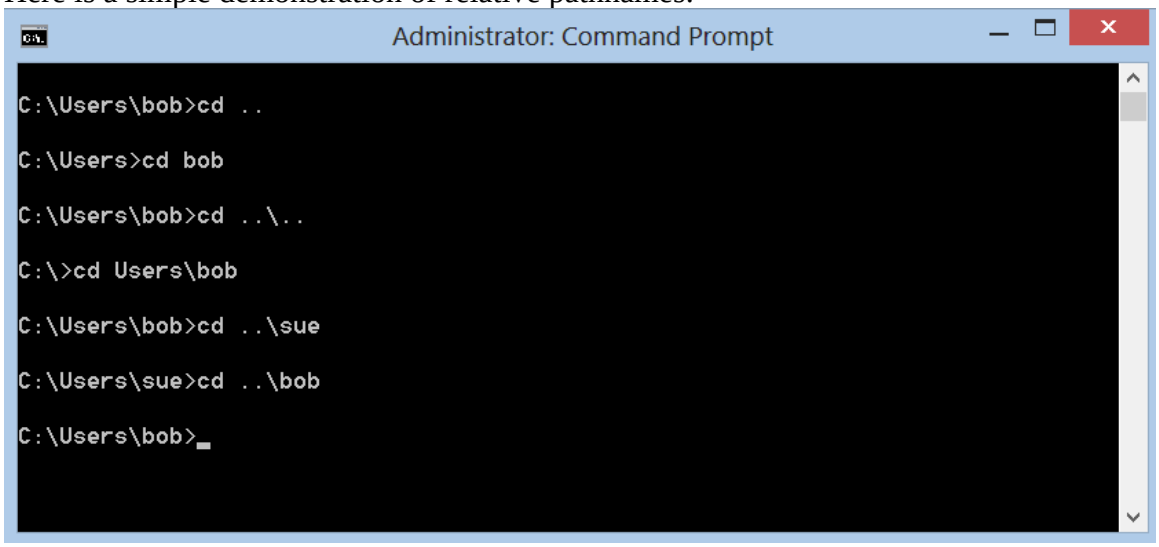
- **.** which refers to the current directory
- **..** which refers to the parent directory

Thus, if the current directory is C:\Users\bob, you can change your directory to C:\Users by typing the command **cd ..** at a command prompt. Similarly, if the current directory is C:\Users, you can move to the C:\Users\bob directory by typing the command **cd bob** at a command prompt; if the pathname does not start at the root directory (i.e. have a \ character at the beginning), the system assumes that the **user1** directory is underneath the current directory.

Say for example, that the current directory is **C:\Users\bob**. You can specify the **cd ** (or **cd C:**) at a command prompt to change to the \ directory of C: using an absolute pathname. To do the same with a relative pathname, the command would be **cd ..\..** at a command prompt.

Similarly, if the current directory is **C:\Users\bob**, you can change to the **C:\Users\sue** directory by typing **cd \Users\sue** or **cd C:\Users\bob** (both are absolute pathnames) or **cd ..\sue** (relative pathname) at a command prompt.

Here is a simple demonstration of relative pathnames:



Most pathnames in Windows are fairly lengthy; thus relative pathnames may save a great deal of typing in certain situations. In addition, the **[Tab] completion feature** of the shell saves having to type long names since it cycles through the list of subdirectories/files as necessary! If there is a directory off of the root directory called **Users**, you can simply type **cd \U** and press the **[Tab]** key to allow the shell to fill in the remaining characters if it is the only directory underneath the \ directory that starts with the letter U. If there were two or more directories underneath the \ directory that started with the letter U, simply press the **[Tab]** key again to cycle through the list of directories that start with U until yours is shown.

Finally, to switch to a different disk drive, simply type the drive letter followed by a colon (:) at the shell prompt. For example, typing **D:** would switch to the root directory on D drive, and typing **E:** would switch to the root directory on E drive.

Exercise 1-1

1. Open a shell as your user account. What directory are you in?

2. Change to the C:\Windows\system32 directory using an absolute pathname and verify that it has been changed.

3. Use a relative pathname to change to the \ directory.

4. Use a relative pathname to change to your home directory.

5. Use a relative pathname to change to the C:\Windows\system32 directory.

6. Use a relative pathname to change to the C:\Windows\temp directory.

7. Switch to your home directory using the **[Tab]** completion feature of your shell.

1.2 Working with Directory Contents

File Types

You can usually determine the file type from the three letter/digit extension on the Windows file (some files have a four letter/digit extension). In general, there are 4 types of files in Windows:

- **Executable Files** are also called application files or command files. These files are the doers. At the shell, we type the file name (extension not needed), and some function is performed. The extensions for these files can be either .exe or .com and cannot be modified.
- **Support Files** typically end in .ini .cfg or .dll and contain application-specific data used by the executable files. As with executables, support files are generally not user-edited. Because of their co-dependence, support files and executable files together are regarded as application files.
- **Data Files** are files that contain our information or user-data. They are modified or edited using word processing, spreadsheet, database or graphic applications. Some of the most common extensions are .doc .docx .txt .xls .dbf and .dxf.
- **Batch Files** are special purpose files. Like executables, they can be "run" from the command line. Like data files, they can be modified in a simple text editor. Batch files (.bat) can automate repetitive command-line processes by listing them line-by-line in a simple text file.

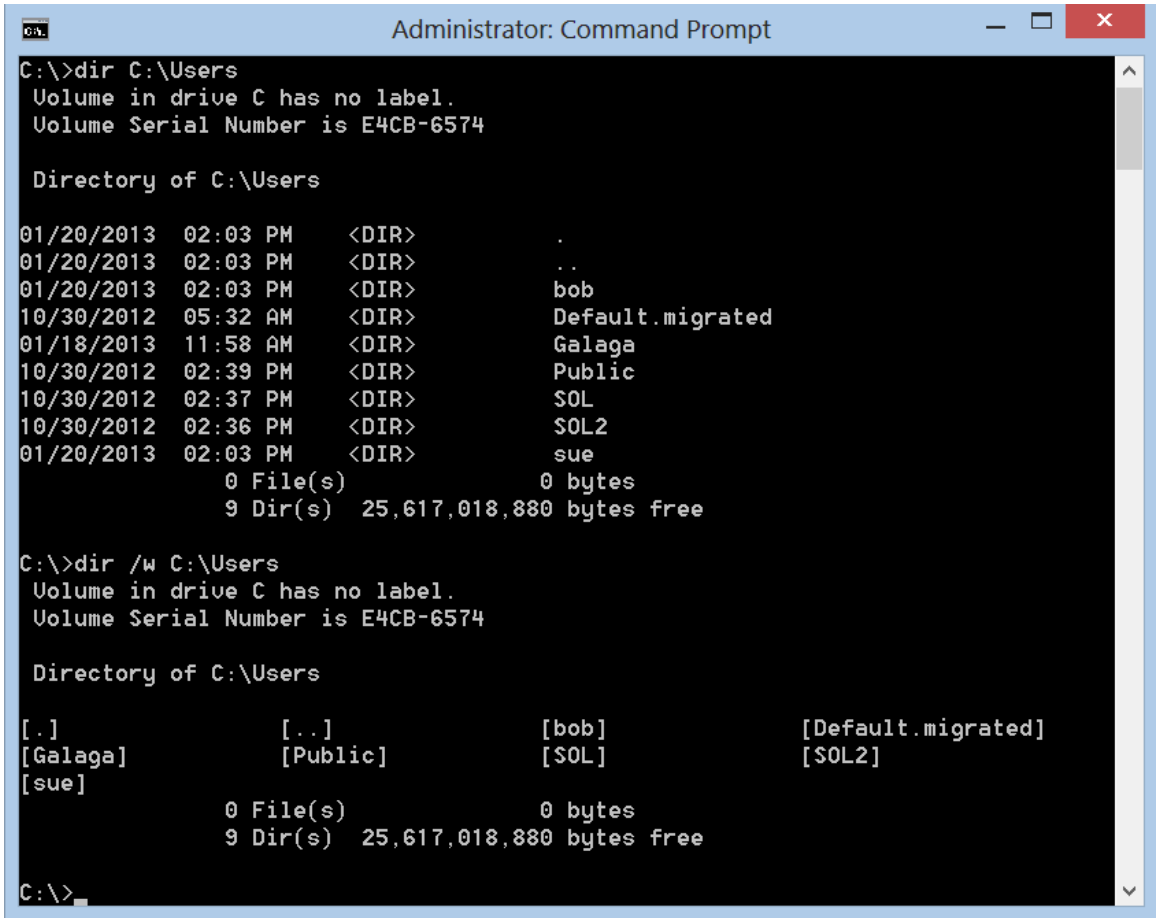
NOTE: There are several characters that can not be used within a file name because they have special meaning to the shell – these are:

\ (back-slash)
/ (forward-slash)
: (colon)
. (dot)
* (star)
? (question mark)
> < (greater-than and less-than symbols)

Viewing Files

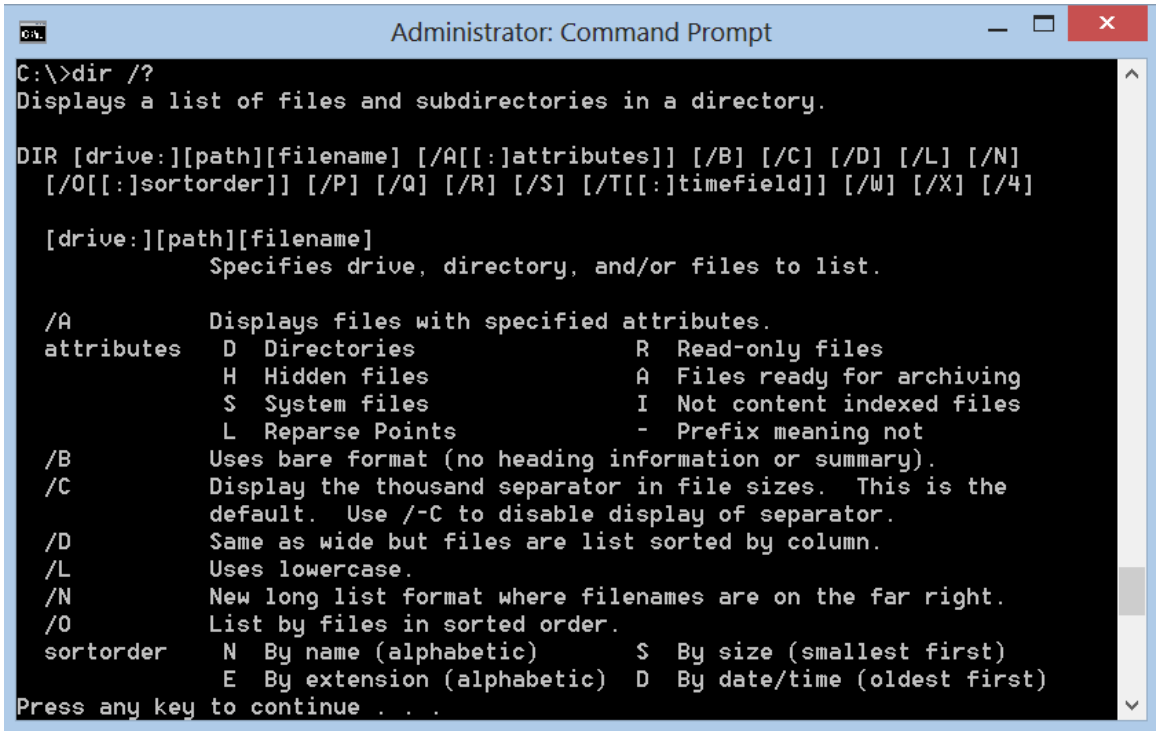
The **dir** command used in Windows shows you a listing of the files and subdirectories within a particular directory. For example, **dir** shows the contents of your current directory, whereas **dir C:\Users** will show you the contents of the C:\Users directory (without changing your current directory). You can use any absolute or relative path as an argument to the **dir** command.

You can also use **switches** (also called **options**) to modify how the dir command works. For example, you can use **dir /p** to show a directory list one screen-full at a time, or **dir /w** to show a directory list in wide format. The **dir /a** command will show all files, including hidden ones. You can combine switches together – the **dir /w/a/p** command will show a wide format one screen-ful at a time of all files.

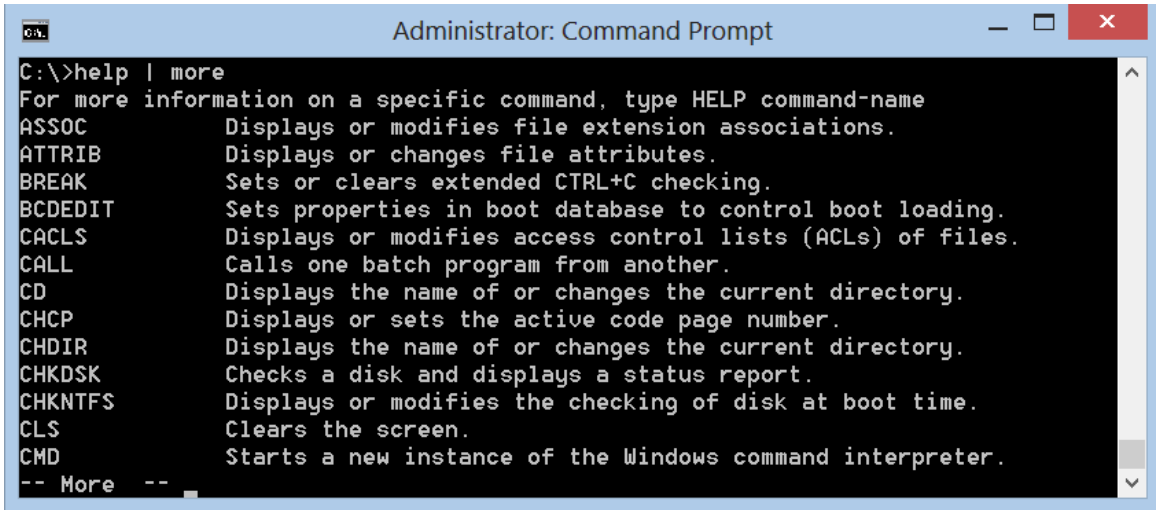


Getting Help

Every command has a **/?** or **/h** option that displays usage information. For example, to see the usage for the **dir** command we used earlier, simply type **dir /?** as shown:



You can also type **help** at any time to see a list of available commands. Since the help command does not support the /p switch (which pauses after each page to list one screen-full at a time), you can type **help | more** (the | is the pipe symbol above the Enter key on the keyboard). You can add the **| more** syntax to any command to force the output to pause page-by-page. To exit out of the page-by-page output, simply press **q**.

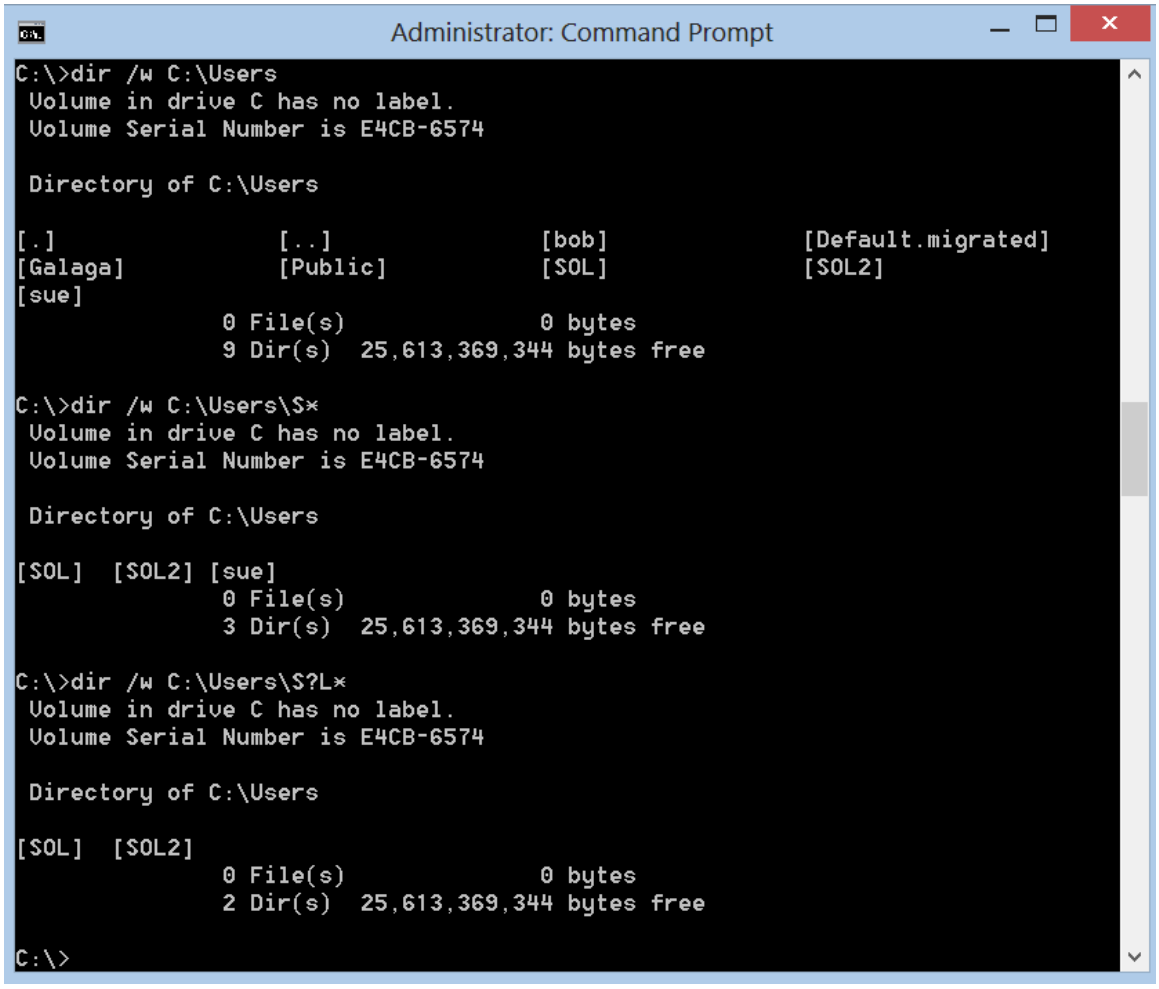


Wildcards

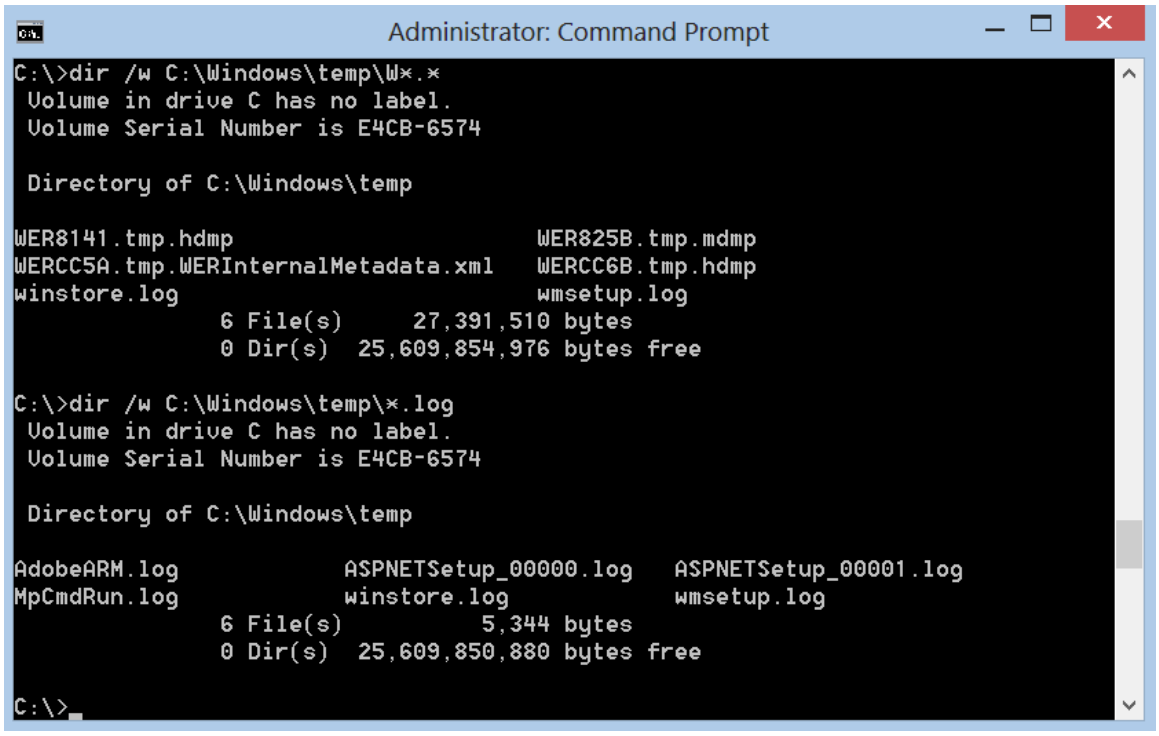
Wildcards in Windows may be used to specify multiple directories or files; these wildcards are listed in Table 2-3.

Table 1-2 Common Shell Wildcards

*****	Specifies zero or more characters	let* matches let, let5, letter…..
?	Specifies one character (any character)	let? matches let1, let2, let3, letA…..



As another example, if you want to view files within the C:\Windows\temp folder that start with W and have any extension, and then view only those that have a .log extension, you could run the following commands:



Managing Files and Directories

There are many commands that you can use within the shell to manage files and directories.

NOTE: Each of these commands take either an absolute or relative pathname as an argument.

Use **md** (make directory) to make a directory. To create a directory named cprograms under the current directory, you could use **md cprograms**, and to remove the same directory (if it is empty), you could use the **rd** (remove directory) command **rd cprograms**.

The **cls** command clears the screen.

To copy files, you use the **copy source destination** command. For example, **copy tm11.c e:\cprograms** copies the file tm11.c from the current working directory into the cprograms directory on E drive. If you omit the destination, the current directory is assumed. For example, **copy e:\cprograms\tm11.c** copies tm11.c from the cprograms directory on E drive to the current directory.

Alternatively, you can use the **move source destination** command to move a file from one location to another.

Use **ren** to rename a file or directory. For example **ren old.c new.c** renames the file old.c as new.c.

Use **del** to delete a file. For example, **del old.c** removes the file old.c, and **del *.*** to remove all the files in the current directory.

Here is an example of using these management commands:

Exercise 1-2

1. Open a shell and navigate around your home directory to view the files and directories present.

2. Experiment using wildcards on some of the files within the C:\Windows\temp folder.

3. Create subdirectories underneath your home directory to organize key files that you have and use the appropriate file management commands to move the files to those directories and remove any unwanted files.