CHAPTER 1

Windows Operating Systems Overview

The Windows operating system (OS) has evolved over several decades of development. To fully understand the way Windows functions today, you should know the roots of the current system. In this chapter, you will learn the history of Windows.

To troubleshoot problems on Windows systems, you must be familiar with the basic architecture of the OS. For this reason, this chapter will also explore the Windows architecture. The architecture defines how the OS functions, and understanding it is essential to grasping many of the topics discussed in later chapters.

Finally, this chapter will describe the interfaces used by administrators and users of the Windows operating systems. This discussion includes exploratory overviews of the graphical user interface (GUI), the Command Prompt, and Windows PowerShell.

- **Discovering the history of Windows**
- **Understanding the OS architecture**
- **Identifying Windows interfaces**

Discovering the History of Windows

The modern Windows OS did not begin with the graphical capabilities it has today. The OS has its roots in text-based systems and simple graphical interfaces. In this section, I'll describe these earlier operating systems to help you understand where the current system came from and why it works as it does. You will also learn about the timeline of Windows development alongside the progressive development of personal computers (PCs). It all begins with the Disk Operating System, better known as DOS.

DOS—The Precursor

The first OS Microsoft sold was MS-DOS 1.0. The very name, Disk Operating System, was indicative of the time when it was released. In 1981, there were no document scanners, Universal Serial Bus (USB) microphones, game controllers, or digital cameras. The primary function of the OS was to allow for the loading of applications and the management of disks or storage. DOS was, and is, a text-based operating system. It had no built-in GUI, and it worked with basic typed commands. Many of these commands still exist in the most current Windows OS.

The DOS OS was popular from 1981 all the way to 1999. After 1999 and the release of Windows 2000, the GUI-based OSs became more popular in business settings.

DOS was originally developed by Microsoft for IBM. In fact, Microsoft licensed a product named QDOS/86 and used it as the starting point to develop MS-DOS. The first version of MS-DOS (version 1.0) was released in August 1981 and supported a maximum of 128 kilobytes of random access memory (RAM). It also supported the File Allocation Table (FAT) filesystem. Figure 1.1 shows the text-based interface for controlling and using DOS. This example is a screen capture from a DOS 6.22 installation showing the output of the CHKDSK command, which was used to view information about the contents of the disk and to analyze the disk for potential problems.

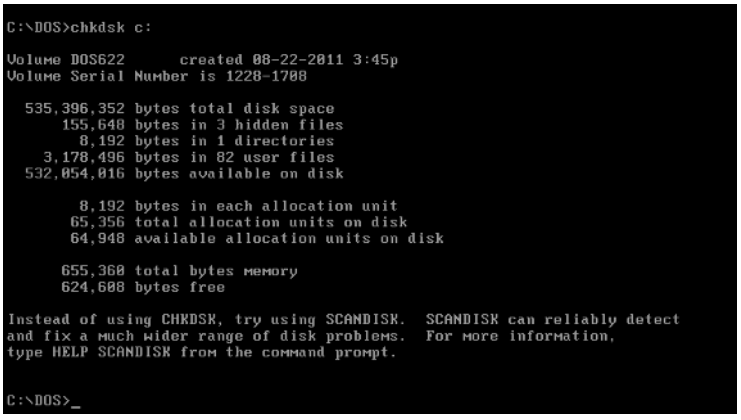

FIGURE 1.1 The DOS 6.22 text-based interface showing the output of the CHKDSK command

DOS applications could have a graphical interface, but the DOS system itself provided no greater graphical functions than a simple ASCII character–based interface. Figure 1.2 shows an example of a DOS ASCII-based application: the

ScanDisk application that shipped with DOS 6.22. ScanDisk checked the disk for errors and attempted to repair any that were discovered.

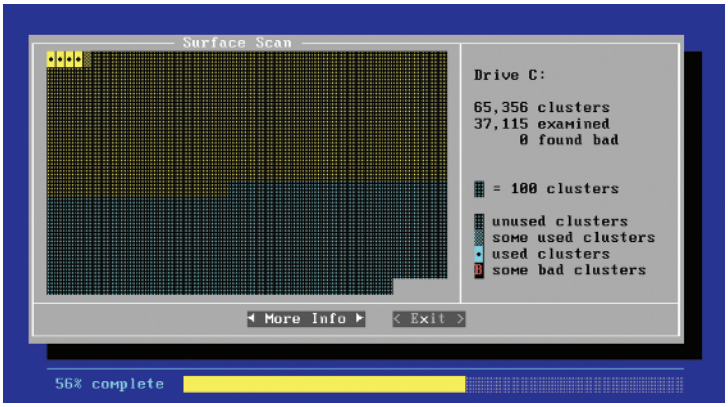

FIGURE 1.2 ScanDisk was an ASCII-based GUI application.

The DOS OS used four elements in the boot process. The first was the boot sector, or boot code. The boot code was stored on the boot drive and indicated that the `IO.SYS` file should be loaded to start the OS. The `IO.SYS` file called and loaded the `MSDOS.SYS` file. When the OS loaded, the command interpreter was loaded as the fourth and final part of the OS. The command interpreter was contained within the `COMMAND.COM` file. Most modern OSs still use the boot sector or boot code, but this code loads different files to start the OS. For example, in a Windows 7 system, the boot code loads the Windows Boot Manager (`BOOTMGR.EXE`) file to begin the OS load.

The `COMMAND.COM` file contained DOS's internal commands, among them the `DIR`, `CD`, and `CLS` commands.

During the boot process, DOS systems used two primary configuration files to determine the drivers and settings for the machine. The first file loaded and processed was `CONFIG.SYS`. This text-based configuration file was used to set system parameters and load device drivers. The second file loaded was `AUTOEXEC.BAT`. This text-based configuration file could perform any function a standard batch file could perform. It was also used to load device drivers and initial applications on the machine.

Batch files were used in DOS to group several commands together as a single unit for easy processing. They also provided scripting capabilities. Batch files are still used in Windows 7 today.

Several versions of DOS were released from 1981 to the final release of version 6.22 in 1994. DOS was the underlying OS in all versions of Windows from Windows 1.0 to Windows ME, including the very popular Windows 95 and Windows 98 operating systems of the 1990s. The version of DOS used in Windows 95 through Windows 98 is often called DOS 7.0, and the version used in Windows ME is often called DOS 8.0. Many vendors released their own DOS distributions that could be used as an alternative to MS-DOS. These competing

versions included Dr. DOS (with the latest release of Dr. DOS 8.1 in 2005), Novell DOS, IBM PC DOS, and PhysTechSoft's PTS-DOS.

Windows 3.1—The GUI

Although several companies, including Apple, Xerox, and Commodore, produced graphical interfaces, there can be no question as to which company has sold more licenses for its graphical interface—Windows GUI interfaces have outsold all the others combined many times over. This popularity is not an automatic testament to its superiority over other GUI interfaces, but it does mean that the typical computing professional is more likely to encounter it than any other interface today.

Windows shipped with several different GUIs from version 1.0 through version 3.0; however, the Windows 3.1 system became popular in the early to mid-1990s and set the path that modern Windows systems are still on today. Figure 1.3 shows the Windows 3.1 GUI, with the Program Manager in the background and the File Manager running in the foreground.

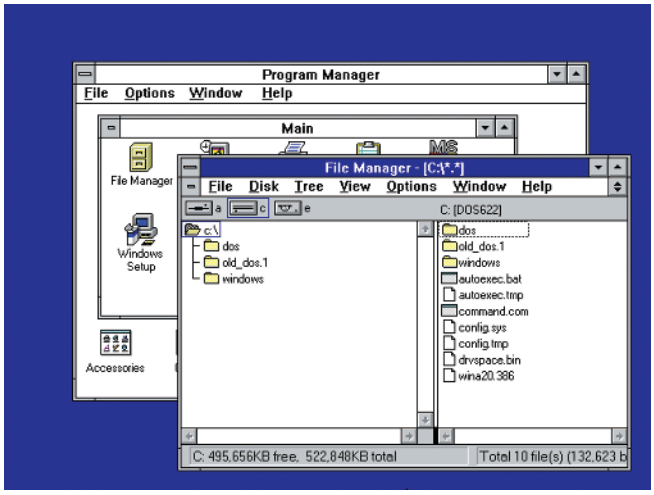

FIGURE 1.3 The Windows 3.1 GUI showing the Program Manager and the File Manager

The Windows 3.1 system included menus, windows that could be resized, and a launching system known as the Program Manager, which supported program groups and icon shortcuts. Many of the concepts used in the Windows 3.1 environment are still used in the modern Windows 7 GUI today.

The next version of Windows, which was based on the DOS and Windows 3.1 systems, was Windows 95. At the same time that the DOS and Windows 3.1 systems

were being used, Microsoft provided another operating system, called Windows NT (New Technology). Windows NT was designed as a network operating system from the start, and it was a 32-bit operating system as well. NT used the same GUI as Windows 3.1. When Windows 95 was released, it drastically changed the launching environment to include a desktop with icons, a taskbar, and a Start menu, which took the place of the Program Manager and program groups. Figure 1.4 shows the Windows 95 interface.

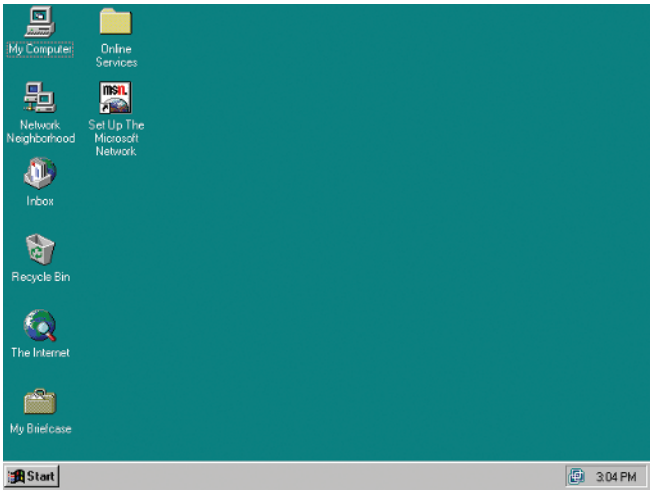

FIGURE 1.4 The Windows 95 Desktop and Start menu

Windows 95 was known as *Chicago* during beta stages. An application is placed in the beta stage for final testing before it is released to the public.

When it released Windows 95, Microsoft implemented a new interface that has lasted for more than 15 years. Windows 7 still uses an interface very similar to the one offered by Windows 95—although we now have a Start menu button that is an orb with the Windows logo on it instead of the word Start, the basic concept remains the same.

In 1996, Microsoft released Windows NT 4.0, the first version of the NT-based OS to use the Windows 95–style interface. The Windows NT–based OSs were not based on DOS, as Windows 3.1 and Windows 95 were. Instead, these more robust OSs include their own boot loaders and kernels. Windows 7 is still based on this NT architecture.

The architecture of Windows 7 is explained in more detail later in this chapter, in the section "Understanding the OS Architecture."

The newest Windows graphical interface as of the release of Windows 7 is the Aero interface. It still provides a desktop and the Start menu and taskbar, but it adds graphically rich capabilities that are only possible with the proliferation of modern powerful graphics chipsets in today's computers. Figure 1.5 shows the Windows 7 interface.

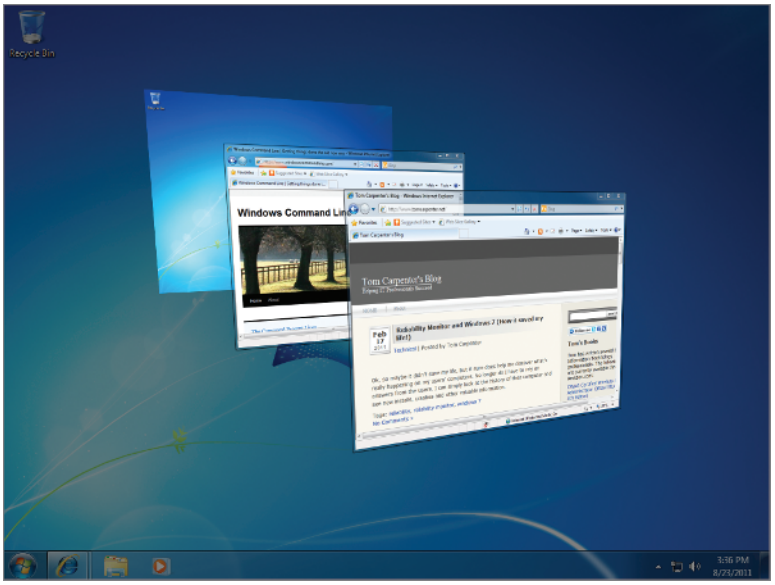

FIGURE 1.5 The Windows 7 Aero interface offers enhanced capabilities compared to its predecessors.

How the Past Is Still in the Present

Many of the elements that we use every day in modern versions of Windows have their beginnings in the early days of personal computers. First, the Command Prompt, which is still very useful in Windows 7, is based on the functionality of the command interpreter, COMMAND.COM, that was part of MS-DOS. You can still use many of the same commands today that people used in the 1980s.

Second, the use of icons has been with us since the Xerox and Apple computers first introduced them and they are still the primary way that we launch files and applications. This is true for desktop and laptop computers as well as most handheld devices.

Third, from Windows 3.1 we still have the concept of the Control Panel in Windows 7. The Windows 3.1 Control Panel had a whopping 11 applets in it right after installing Windows. Needless to say, Windows 7 has many more applets in its Control Panel, but the Control Panel remains just the same.

To see an interesting video demonstrating the history of the Windows OS through sequential upgrades from one to the next, search for "Chain of Fools: Upgrading Through Every Version of Windows" at YouTube.com.

Windows Evolution

As you learned in the preceding section, the Windows OS did not just appear but has evolved over time. Initially, Windows was a graphical shell that ran on top of DOS. This functionality continued through the Windows 95 line of OSs until it ceased with Windows Millennium Edition (ME).

Starting in 1993, Microsoft sold an alternative OS named Windows NT that depended on an entirely different architecture. It used the same graphical interface, but it did not run on top of DOS. Windows NT 4.0 inherited the modern Desktop and Start menu concept from Windows 95, and this basic interface concept is still used in Windows 7 today. DOS no longer exists in Windows 7, but the Command Prompt interface provides a command line or text mode interface for interacting with the OS. The Command Prompt is very similar to DOS, and it is clearly based on its predecessor.

For more information about the history of Microsoft Windows, visit `http://windows.microsoft.com/en-US/windows/history`.

Figure 1.6 shows a timeline that compares the evolution of PC hardware with the evolution of the Windows OS. As the image portrays, more powerful hardware has allowed for more features in the OS, so that we can use the graphically rich interfaces with animations and special effects that we use today. The memory amounts listed in Figure 1.6 do not represent the maximum supported memory of the named systems, but rather the common memory amounts used with them. (You'll notice that the timeline doesn't list every Windows operating system; its purpose is to display the "types" of operating systems. Windows for Workgroups 3.11, for example, was really just Windows 3.1 with networking built in. Windows 98 and ME are really just 95 with a few enhancements. Windows Vista, as discussed separately, never met with wide acceptance.)

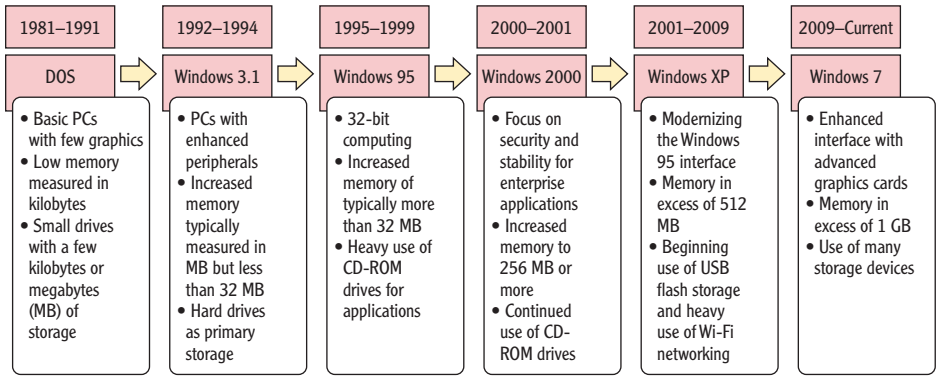

FIGURE 1.6 The Windows OS evolution alongside PC hardware evolution

What about Vista?

If you investigate Figure 1.6 closely, you will notice that Vista is not mentioned. This is not because it is an insignificant release of Windows—quite the opposite is true. Windows Vista was a revolutionary release of Windows that was a bit before its time. The hardware available at the time of its release was simply unable to take advantage of its full capabilities. For this reason, many users were frustrated with the performance of Vista on their computers and chose to run Windows XP instead. In addition, many applications and devices failed to work as needed on the Vista OS due to the lack of drivers required from hardware vendors.

Windows 7 resolved many of the problems with Vista by providing enhanced application compatibility solutions and improved hardware support. Additionally, with the passage of time, the hardware in common use at the time of Windows 7's release runs the newer OS very well. For this reason, Windows 7 has experienced an adoption rate on a scale unseen since the release of Windows 95 in the mid-1990s, with more than 240 million licenses sold in the first year of its release.

Understanding the OS Architecture

The architecture of an OS defines how it works instead of what it can do. The features define what it can do, but those features must work on top of an operational methodology. This methodology is the *architecture*. The current Windows OS architecture is based on the original architecture in the Windows NT OS first released in 1993. Windows 7, building on changes included in the less popular Windows Vista, introduces significant changes in several areas of the architecture, and they are addressed in this section. You will first explore the layers in Windows that make up its architecture and then compare it with other systems.

The Layers in Windows

The Windows OS is divided into layers, or modes of operation. The first is Kernel mode and the second is User mode, as depicted in Figure 1.7. Kernel mode is where the operating system kernel and other low-level processes operate. User mode is where your applications, such as Microsoft Word or Excel, and environment subsystems run.

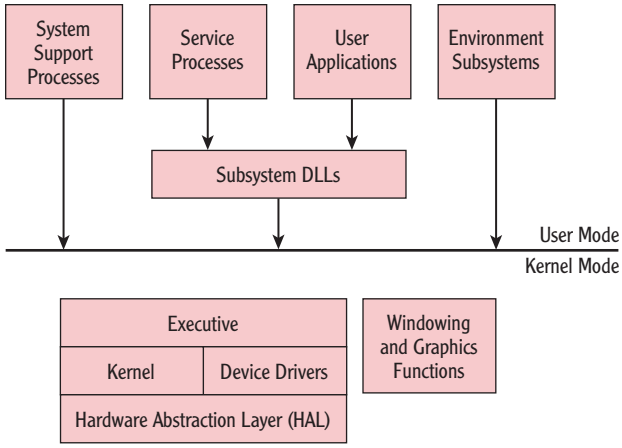

FIGURE 1.7 The basic Windows architecture components

Technically, Kernel mode operations take place in something called ring 0 of complex instruction set computing (CISC) processors. Most processors in use in computers today are CISC processors. The common x86 (32-bit) and x64 (64-bit) nomenclature refers to CISC processors based on their support for 32-bit and 64-bit computing. User mode operations take place in the processor's ring 3 (also known as nonprivileged mode). The different rings of operation simply define the level of access granted to the processes running in that ring. For example, ring 0 (also known as privileged mode) is like "God mode" in many games. In this mode, entered via cheat code, you can do anything you want without fear of dying or being injured within the game. In a similar way, ring 0 processes can do anything they want within the system. It is for this reason that only operating system functions, device drivers, and graphics capabilities reside here.

To simplify the way rings work in CISC processors, just remember that ring 3 processes can do only what ring 2 processes allow them to do. Ring 2 processes can do only what ring 1 processes allow them to do, and ring 1 processes can do only what ring 0 processes allow them to do. Because Windows uses only rings 0 and 3, we can simplify this and say that ring 3 processes can do only what ring 0 processes allow them to do.

Ring 3 processes are usually called user application processes. Because user applications must access hardware, which can only be done through Kernel mode, some method must exist to allow for this. Effectively, user applications switch from User mode processes to Kernel mode processes for short durations in order to accomplish Kernel mode tasks. However, even in such situations a Kernel mode process must authorize the switch, which brings us back to the

Rings 1 and 2 are not used in Windows systems. This decision was made when other processors were also supported that offered only two modes of operation within the processor.

Ring 0 is also known as privileged mode and ring 3 is known as nonprivileged mode.

When monitoring processor utilization on a Windows system, you will notice that most of the time is spent in Kernel mode because this mode has access to the hardware.

reality that User mode (ring 3) processes can do only what Kernel mode (ring 0) processes allow them to do.

In addition to Kernel mode and User mode, Figure 1.7 shows several components, described here:

System Support Processes System support processes include system processes that do not run as services. These include processes like the logon process (`WINLOGON.EXE`) and the Session Manager subsystem (`SMSS.EXE`). The Session Manager is the first User mode process that starts when the OS loads. The Session Manager launches environment subsystems and the `WINLOGON.EXE` process.

Service Processes Service processes are those system or application processes that do run as services. These include system services like the Task Scheduler and the Print Spooler, but they also include nonsystem services like SQL Server, which is a database server service. A service is an application or innate operating system component that provides services (capabilities) to the network, the local machine, or both.

User Applications User applications are the actual applications that users execute. These include applications like Microsoft Excel and Adobe Photoshop. Five application types are supported in the Windows OS: Windows 32-bit and 64-bit applications, Windows 3.1 16-bit applications (on 32-bit editions of Windows 7), MS-DOS 16-bit applications (on 32-bit editions of Windows 7), and Portable Operating System Interface (POSIX) 32-bit applications through the use of the Subsystem for Unix-based Applications. 16-bit applications are not supported on the 64-bit editions of Windows.

Environment Subsystems The environment subsystems and the subsystem dynamic link libraries (DLLs), which are defined next, work together to allow different application types to function on the system. For example, the Windows 32-bit environment subsystem allows 32-bit Windows applications to work on a 64-bit edition of Windows 7. This 32-bit environment subsystem will make calls to subsystem DLLs, and the subsystem DLLs may also communicate back with the environment subsystem if communication with the application is necessary.

Dynamic link libraries (DLLs) contain code that can be called upon by other processes so that programmers need not re-create the code for each application.

Subsystem DLLs The last User mode component is the subsystem DLLs. These translate application function calls into internal native system service calls. The subsystem DLLs communicate with the Kernel mode processes on behalf of the applications and may communicate with the applications through the environment subsystems as well.

Executive The executive can be compared to an executive assistant. This component is responsible for process and thread management, memory management, security functions, input and output (I/O), networking, and communication between processes (interprocess communication). The executive consists of subcomponents responsible for specific tasks. For example, the Configuration Manager implements and manages the Registry, which stores settings for applications, users, and the OS. Additionally, the I/O Manager provides device-independent I/O and passes requests to the proper device drivers for actual processing on the hardware.

Windowing and Graphics Functions A key feature of Windows is the graphical user interface, and these functions are implemented in Kernel mode. This is a major reason why unstable video drivers often cause the entire Windows OS to become unstable. The drawing of windows and user interface control objects (such as buttons, scroll bars, and title bars) is controlled here.

Kernel While the executive is in charge of process and thread management, the kernel is in charge of thread scheduling. It decides which thread gets processor time and on which processor it gets time at any moment. It is also responsible for synchronization when multiple processors are used and for interrupt handling through the interrupt objects, which work in relation to the I/O Manager.

Device Drivers Device drivers are Kernel mode components that provide a communication interface between the I/O Manager within the executive and the actual hardware for which they are written. Device drivers place calls to the Hardware Abstraction Layer, described next, to communicate with the hardware. Device drivers may be used to communicate with specific hardware or with filesystems, networks, and other protocols.

Hardware Abstraction Layer (HAL) The HAL does exactly what its name implies—it abstracts or disconnects the OS from the core hardware, such as the processor architecture. Because of the HAL, Windows can run on systems that support 32-bit processing (x86) or 64-bit processing (x64). Additionally, with the x86 processors, it can run on either Advanced Configuration and Power Interface (ACPI) PCs or Advanced Programmable Interrupt Controller (APIC) PCs.

PCs that use the ACPI HAL are single-processor machines. PCs that use the APIC HAL are multiple-processor machines.

Several files work together to provide the core functionality of the Windows OS. These files include the following:

`ADVAPI32.DLL` One of the primary Windows subsystem DLLs providing access to APIs for Registry access, system shutdowns and restarts, and management of user accounts.

GDI32.DLL One of the primary Windows subsystem DLLs providing graphics functions.

HAL.DLL The Hardware Abstraction Layer (HAL) DLL that allows the Windows OS to run on different hardware platforms.

KERNEL32.DLL One of the primary Windows subsystem DLLs providing kernel functions.

NTDLL.DLL The DLL that exposes many of the Windows native API functions to User mode applications.

NTOSKRNL.EXE The kernel image for the Windows OS; the kernel is responsible in part for process and memory management.

NTKRNLPA.EXE The same as NTOSKRNL.EXE, but used on systems with Physical Address Extension (PAE) support.

PAE allows 32-bit or x86 processors to access memory in amounts greater than 4 GB. With PAE, it is possible for a system to support up to 64 GB.

USER32.DLL One of the primary Windows subsystem DLLs providing access to the keyboard and mouse as well as window management (the actual application windows as opposed to the OS name).

WIN32K.SYS The Kernel mode portion of the Windows subsystem.

Windows 7 specifically modifies the architectural functionality of Windows in several ways. The following enhancements have been included in the Windows 7 architecture and design:

- The version number of the OS has changed to 6.1 from 6.0, which was the older version number in Windows Vista. Even though the version number seems to indicate that Windows 7 is a minor release (6.1) and Windows Vista was a major release (6.0), Microsoft considers Windows 7 a major release.
- More than 400 footprint reductions were implemented across all Windows 7 components, resulting in a smaller memory footprint than Windows Vista.
- The Desktop Window Manager (DWM), which provides the Aero window interface capabilities, has a 50 percent reduced memory footprint.
- Microsoft added power management efficiency tests to assist users in increasing battery life. The POWERCFG.EXE command can be used to generate a report on power issues.
- The Windows 7 OS is prepared for BitLocker immediately after installation because a 100 MB volume is created for BitLocker use during installation.

- BitLocker also supports encryption of USB flash drives through BitLocker to Go.
- Microsoft added the ability to boot the system from a virtual hard disk (VHD). A VHD is used to provide hard disks to virtual machines in virtualization systems like Windows Virtual PC and Hyper-V. Booting from a VHD allows for simpler dual-booting and testing. Windows 7 systems can boot from USB flash drives, VHD files, hard drives, CD/DVD drives, and other USB-type drives.

For more information on the architectural changes in Windows 7, view the following page at the Microsoft Developers Network (MSDN) website: `http://msdn.microsoft.com/en-us/library/dd371741(v=VS.85).aspx`.

Windows Compared to Other Systems

The architecture of Windows is very similar to other operating systems, including Linux and Mac OS X. A key difference is in the area of graphics processing. Windows includes graphics capabilities in the Kernel mode of the operating system, which can potentially improve graphics performance. Linux does not include the graphics functions in the kernel, which removes those functions from the core OS and can potentially improve stability.

The Mac OS X operating system is now based on BSD Unix. This flavor of Unix has been around for more than 30 years. However, the hardware on which you run it today is not 30 years old, and therefore, the length of development time does not automatically result in improved stability. For example, Windows has been developed since the official Microsoft announcement in 1983. Therefore, the length of development history is really no different when comparing Windows with any other OS.

Apple provides an open-source version of its core OS known as Darwin. Darwin is the OS used for Mac OS X, but the GUI is provided through Quartz and Aqua, which are not available in the open-source version. Quartz is the drawing engine and Aqua is the theme. The most widely used open-source version of Darwin is PureDarwin, which is available at `www.PureDarwin.org`.

Visit the following websites to learn more about differences between Windows OSs and other operating systems:

- `www.microsoft.com/canada/windowsserver/compare/default.mspx`
- `windows.microsoft.com/en-US/windows7/products/compare/pc-vs-mac`
- `www.michaelhorowitz.com/Linux.vs.Windows.html`
- `www.cio.com/article/41140/Windows_vs._Linux_vs._OS_X`

The stability versus performance issue is hotly debated in the industry. For more information on the differences between Linux and Windows, see `http://en.wikipedia.org/wiki/Comparison_of_Windows_and_Linux`.

Identifying Windows Interfaces

The Windows OS has three primary interfaces for working with applications and performing administrative tasks. Most users will spend the majority of their time in the GUI, but administrators can benefit greatly from the Command Prompt and Windows PowerShell interfaces. This section includes explanations and examples of all three interfaces in the Windows 7 OS.

Using the GUI Interface

The Windows GUI is composed of windows, buttons, text boxes, and other navigation elements. To support Windows systems effectively, you must understand the basics of these elements. Figure 1.8 shows the Start menu and taskbar elements included in the Windows 7 GUI.

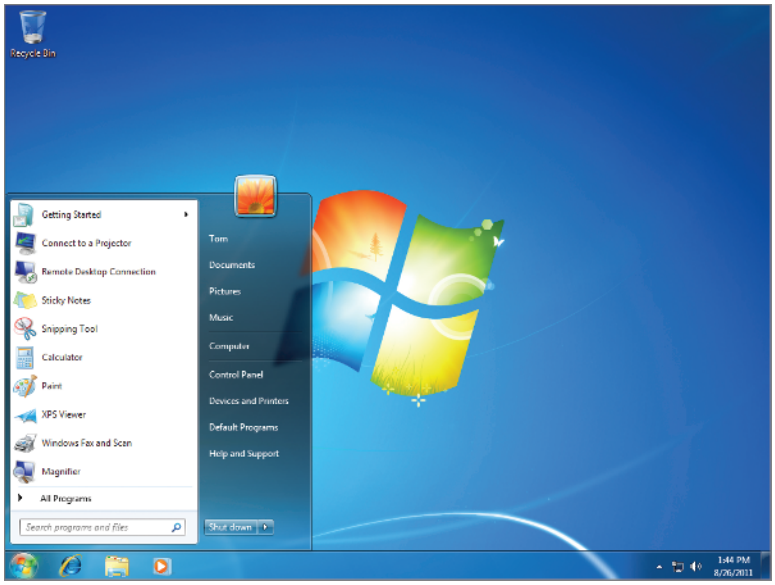

FIGURE 1.8 The Windows 7 GUI interface with the Start menu expanded

You can quickly access the Start menu by pressing the Windows key on a keyboard or by pressing Ctrl+Esc if the keyboard lacks a Windows key.

The first element is the Start menu. As you can see in Figure 1.8, the Start menu is similar to earlier versions of Windows, but it includes new capabilities. First, a graphical orb icon (first introduced in Windows Vista) is still the button used to bring up the Start menu in Windows 7. The Start menu includes the ability to pin icons so that they are always displayed on the menu, and it features a listing of the recent applications launched for quick access.

To customize the Start menu, simply right-click the Start menu button (the orb) and select Properties to access the Taskbar And Start Menu Properties dialog, shown in Figure 1.9. You can use the Customize button to change how the Start menu looks and behaves. You can also configure the Start menu's Power button to do any of the following tasks: shut down, switch user, log off, lock, restart, or sleep. Finally, you can configure Privacy settings according to your needs.

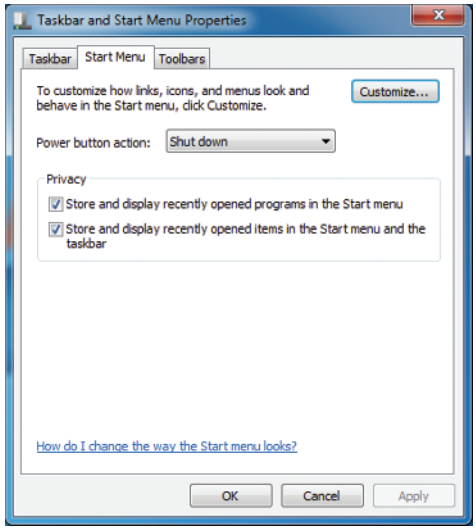

FIGURE 1.9 The Taskbar And Start Menu Properties dialog

Using the Taskbar And Start Menu Properties dialog, you can also configure the taskbar. On the Taskbar tab you have several options, including the ability to lock the taskbar, auto-hide the taskbar, and use small icons so that the taskbar can contain more icons at lower screen resolutions. You can also define the taskbar location on the screen and the way the taskbar buttons should be displayed. For example, you can display the taskbar buttons in groups so that they stack on top of each other for multiple instances of the same application, or you can display them as completely separate buttons. Finally, you can customize the Notification Area. This area is, by default, in the lower-right corner of the Desktop and is displayed as part of the taskbar. It displays icons that allow you to interact with the system and applications, such as the volume control icon and alert icons that display important information related to your system. You can control which icons should be displayed in the Notification Area and which should not.

The next major portion of the Windows OS GUI is the Desktop. The Desktop is shown in Figure 1.8 with the Windows logo in view and the Recycle Bin icon

The screen resolution defines the number of pixels used horizontally and vertically to draw the screen. Common resolutions are 1024×768 and 1280×1024.

displayed. The Desktop can contain system icons and custom icons. It can also display gadgets such as weather and system resource gadgets for quick display of potentially important information. To add gadgets to the Desktop, right-click the Desktop and select Gadgets. To customize the system icons on the Desktop, right-click the Desktop and select Personalize; then choose Change Desktop Icons on the Personalization screen. From here, you can add any of the following icon items to the Desktop and change the icon used to represent the item:

- Computer
- User's Files
- Network
- Recycle Bin
- Control Panel

Applications are not forced to use the standard application window. Many applications run in full-screen mode. Games are a perfect example of this behavior.

Figure 1.10 shows the next important component in the Windows GUI: a standard application window. In this case, I've chosen the Notepad application as an example because it represents the most basic interface elements used by Windows GUI applications. The primary elements are the title bar, menus, application control buttons, and the application workspace. The title bar typically displays the application name and any open document name. Programmers can display anything they want in the title bar, but the common practice is to display this information.

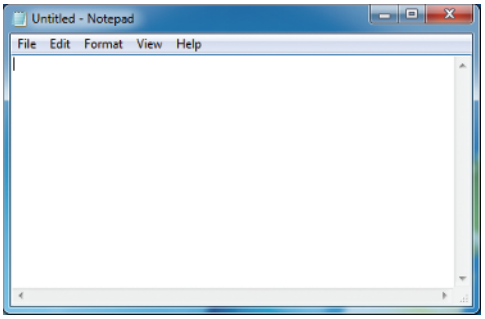

FIGURE 1.10 Notepad is an example of a typical application window.

The menus are a collection of one or more groupings of commands. For example, the File menu typically provides file management functions such as opening

or closing files and saving files. Each application has its own distinct set of menus, but the File, Edit, View, and Help menus are common across applications.

The application control buttons are located in the upper-right corner of the application window. The first button, shown on the left in Figure 1.10, is used to minimize the application to the taskbar. The second button is used to maximize the application. The final button, depicted with an X, is used to close the application. The following keyboard shortcuts apply to these buttons in Windows 7:

- Maximize: Windows Key+Up Arrow
- Minimize: Windows Key+Down Arrow
- Close: Alt+F4

You will use the graphical interface for most of the instructions provided throughout the rest of this book; however, it is important to understand the text-mode interfaces as well, and the following two sections explain them. Chapter 3, "Managing the Desktop," provides detailed information about desktop management in the Windows GUI.

Using the Command Prompt Interface

The Command Prompt is based on the oldest interface Microsoft has provided: the command interpreter in DOS. In Windows 7, the Command Prompt is either a 32-bit or 64-bit command-line interface to the operating system. It is not DOS, but it has many similarities to it.

On 32-bit editions of Windows 7, the Command Prompt is a 32-bit application. On 64-bit editions, the Command Prompt is 64-bit by default; however, a 32-bit version of the Command Prompt is located in the *%WinDir%*\SysWOW64 folder as CMD.EXE. To access the 32-bit version of the Command Prompt on the 64-bit edition of Windows 7, follow these steps:

1. Click Start.
2. In the Start menu search field, enter *%windir%*\syswow64\cmd.exe.
3. Press Enter.

The Command Prompt interface is shown in Figure 1.11. You can access the this prompt using several methods. To access it from the Start menu, click Start ➢ All Programs ➢ Accessories ➢ Command Prompt. To access it from the search field, click Start and then enter **cmd.exe** in the search field and press Enter.

Throughout this book, the variable *%WinDir%* is a reference to the location where Windows is installed, which is typically C:\Windows. This is also a variable for the same location within the OS.

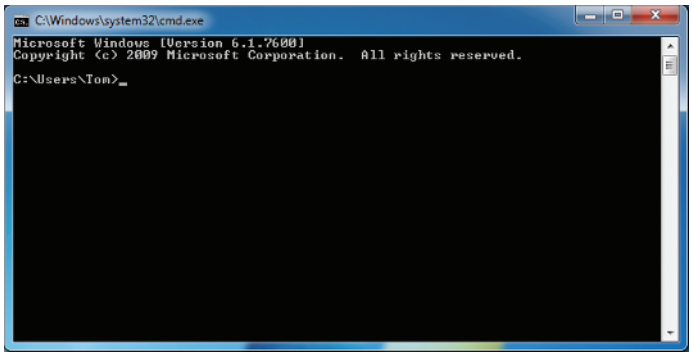

FIGURE 1.11 The Windows Command Prompt interface

You can configure several items in the Command Prompt Properties dialog. To access this dialog, with the Command Prompt open click the icon in the upper-left corner of the application window and select Properties. From here you can configure the Options, Font, Layout, and Colors settings. Many people enjoy creating a custom color scheme for the Command Prompt. One of the most important settings to configure from the Properties dialog is Screen Buffer Size. You will usually want to set the Height value to something much greater than 300, which is the default. Higher values allow you to scroll back through more information. On modern computers with plenty of RAM, it is acceptable to set the Height value to 9,999, which is the highest possible value.

For more detailed information on the various commands available at the Command Prompt, visit `www.WindowsCommandLine.com`. At this free website, I've documented each command with examples, and video demonstrations are planned for the future.

Using Windows PowerShell

Windows PowerShell is the new command-line interface Microsoft first released as an add-on to earlier versions of Windows. PowerShell 2.0 comes with Windows 7 out of the box and it is Microsoft's preferred environment for developing administrative tools in the future. In fact, many Microsoft technologies now come with a complete set of *cmdlets* (the name for commands in the PowerShell interface) for administration purposes and the graphical interfaces call the PowerShell cmdlets to do their work. For example, Exchange Server 2010 comes with a complete set of administration cmdlets and the GUI management tool for Exchange Server 2010 calls these cmdlets to perform administrative actions.

Figure 1.12 shows the Windows PowerShell interface running on Windows 7. Like the Command Prompt, PowerShell has a Properties dialog where you can modify several properties. Many of the same settings available for the Command Prompt are available for Windows PowerShell, and you will use Windows PowerShell in several examples throughout this book. It plays a significant role in Windows OS administration now and will continue to do so for the foreseeable future.

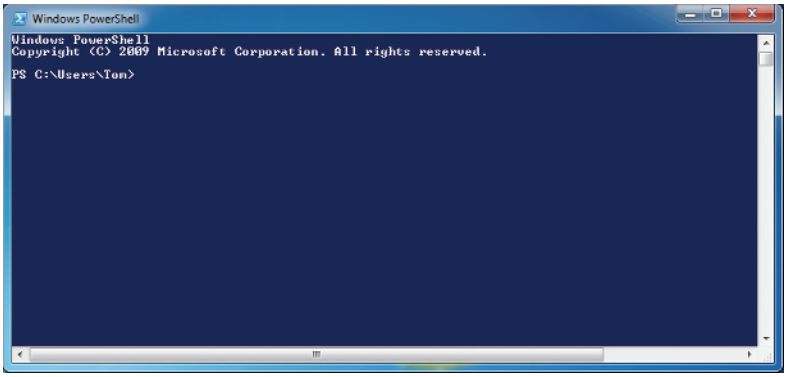

FIGURE 1.12 The Windows PowerShell interface

Among the primary benefits of Windows PowerShell are its scripting and automation interfaces. Using the built-in support for variables and logical functions, you can accomplish much more with PowerShell than you could with a batch file at the Command Prompt. Additionally, you can use PowerShell to execute commands against remote computers using the built-in support for the Windows Remote Management (WinRM) service.

For more detailed information about using Windows PowerShell and the available cmdlets for various purposes, visit `www.MasterWindowsPowerShell.com`. At this free website, I've documented the PowerShell interface, and video demonstrations are also available.

The Essentials and Beyond

In this chapter, you were introduced to the Windows OS. First, you explored the history of Windows, which began in MS-DOS, worked its way through Windows 3.1, and ended up where we are today in Windows 7. Next, you explored the architecture of Windows 7, which is based on the original Windows NT architecture from the 1990s. The architecture

(Continues)

The Essentials and Beyond *(Continued)*

employs a layering scheme with user applications running primarily in User mode and the OS and device drivers running in Kernel mode. Finally, you learned about the three basic interfaces available for Windows OS interaction: the GUI interface based on windows, icons, and graphics; the Command Prompt, which is the earliest text-mode interface to the Windows OS; and Windows PowerShell, the newest interface to the Windows systems that provides exceptional power in scripting and automation.

Additional Exercises

- Research the differences between Windows and other OSs online.
- Use the `Import-Module` cmdlet to add a Windows PowerShell module that is not included in the Windows PowerShell interface by default.
- Review the history of Microsoft Windows at the Microsoft website to learn how it has evolved to become the operating system it is today.

To compare your answers to the author's, please visit `www.sybex.com/go/osessentials`.

Review Questions

1. What OS preceded Windows and was the foundation for Windows 3.1 and Windows 95?

   A. OS/2 **C.** DOS

   B. Mac OS X **D.** Linux

2. True or false. Batch files that are commonly used in Windows 7 today were also used in DOS.

3. Which one of the following components that existed in Windows 3.1 is still in Windows 7 today?

   A. Program Manager **C.** Action Center

   B. Calendar **D.** Control Panel

4. What Windows OS operational mode includes device drivers and the window and graphics management code?

5. What new Windows 7 command can be used to generate a power management report?

   A. `POWERCFG.EXE` **C.** `TASKLIST.EXE`

   B. `SC.EXE` **D.** `NET.EXE`

(Continues)

The Essentials and Beyond *(Continued)*

6. Define Windows PowerShell.
7. Define a process.
8. In what directory or folder is the 32-bit version of the Command Prompt located on 64-bit systems?

   A. `C:\Windows\DOS32`
   B. `C:\Windows\System32`
   C. `C:\Program Files(x86)`
   D. `C:\Windows\SysWOW64`

9. What is the command used to launch the Windows command prompt in Windows 7?

   A. `CMD.EXE`
   B. PowerShell
   C. `COMMAND.COM`
   D. Prompt

10. True or false. You can add the Control Panel to the Desktop in Windows 7.