Chapter 2

Windows Server 2003 Structure and Architecture

The underlying operating system structure and the relationship of component parts are fundamental to deploying and managing Windows Server 2003 environments. By understanding the operating system architecture, the system administrator should be in a better position to install, configure, optimize, and troubleshoot Windows Server 2003. This chapter is an architectural overview of Windows Server 2003. After reading it, you should have the following:

- An understanding of the operating system's structural layers, subsystems, and managers, including the executive kernel modes and user modes, subsystems, and managers
- A working knowledge of Windows Server 2003 process management, including multitasking, the interplay of processes and threads, process viewing, and management tools
- A perspective on physical and virtual memory management
- A basic understanding of the boot process
- A working grasp of the registry's function and structure
- An understanding of application dependencies and software compatibility with Windows Server 2003 through the use of its tools

There have been relatively very few underlying architectural changes made from Windows 2000. However, there are a number of enhancements that will have a beneficial impact on administration.

STRUCTURAL MODES, SUBSYSTEMS, AND MANAGERS

A surface view of the Windows Server 2003 structure reveals an eloquently simple arrangement of functions that separates system-related events from user-related events. As you move deeper into the components of Windows Server 2003, you will see that Microsoft has designed a very compartmentalized operating system. In this section we will review:

- The structural layers of the kernel mode, the Hardware Abstraction Layer (HAL), and the user mode
- The role of the Windows Server 2003 executive mode and its managers
- The role of the Windows Server 2003 user mode and its subsystems

Structural Layer Modes

Windows Server 2003 functions in two primary modes: the privileged kernel, or executive, mode and the open nonprivileged user mode (Figure 2.1). Low-level operating system services, system data, and interfaces to hardware are controlled by the kernel mode.

The user mode handles everything else that is subject to user interface or intervention, including the default Win32 subsystem, optional subsystems, and applications. The user mode interacts with system data and hardware through a tightly integrated API.

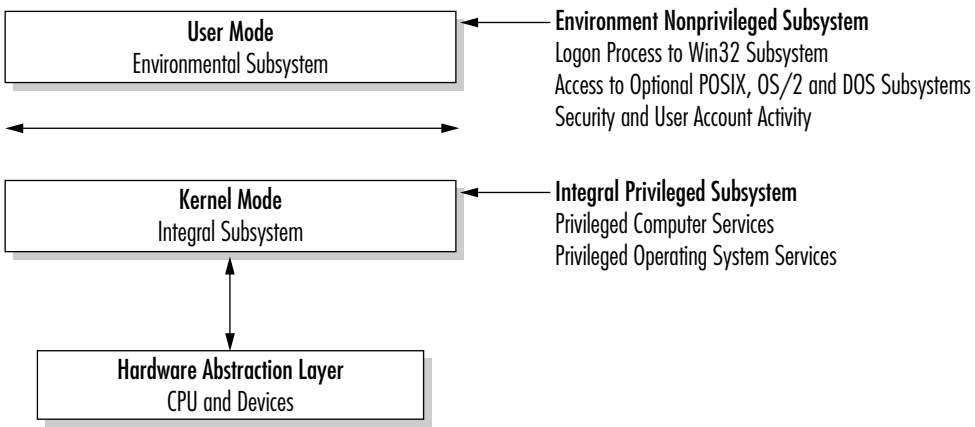


FIGURE 2.1 Relationship of the User and Kernel Modes

The Windows Server 2003 Executive Mode

The Windows Server 2003 Executive (Figure 2.2) is also known as the kernel mode and the privileged executive mode. Windows Server 2003 breaks its operations into five segments that run in the kernel or privileged mode:

- Hardware abstraction layer (HAL)
- Microkernel
- Device drivers
- Executive managers
- Executive services buffer

Collectively, these elements handle the system responsibilities that are hidden from the user. In Windows Server 2003, the Executive controls essential operating system functions. Other functions are pushed into the nonprivileged area or into protected subsystems, as discussed in the next section. The elements of the Executive are discretely independent and exchange data through defined interfaces. In theory, any component can be deleted and replaced with a technologically updated version. Assuming adherence to the interface APIs, the operating system should function without difficulty after swapping Executive components.

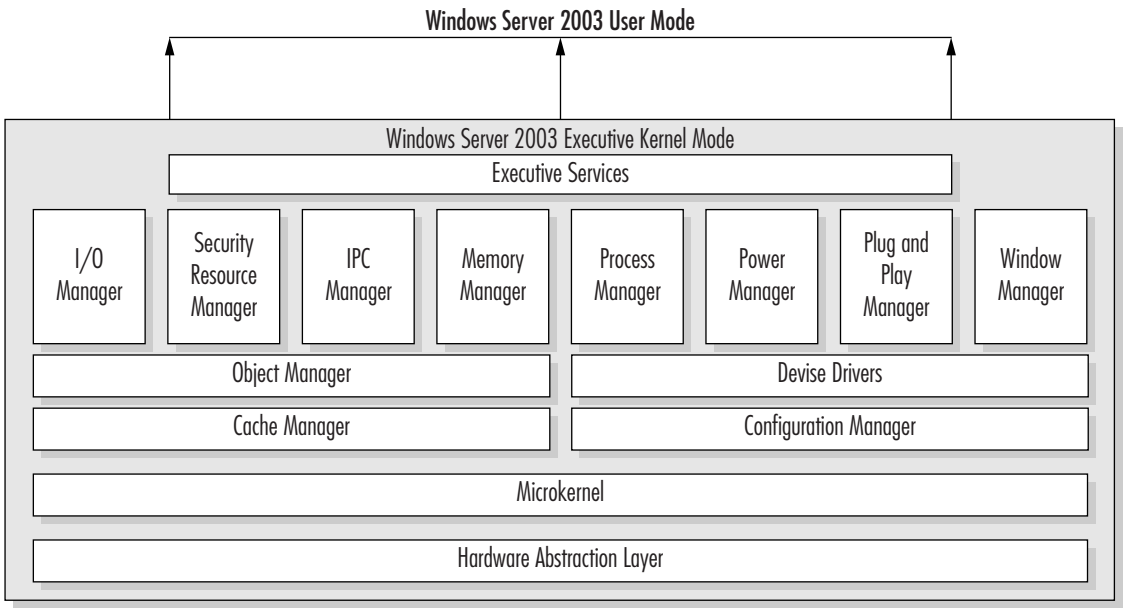


FIGURE 2.2 The Windows Server 2003 Executive Kernel Mode

Each element of the Executive provides two discrete functions. The system services are available in both user-mode and kernel-mode mode operations. By contrast, the internal routines are used only to communicate with other managers or components within the Executive itself.

THE HARDWARE ABSTRACTION LAYER

At the base of the Windows Server 2003 Executive is the Hardware Abstraction Layer. Microsoft originally placed hardware-related interfaces in a discrete segment of code as a means of ensuring greater portability across platforms. Early Windows NT was to be a cross-platform operating system in which HAL provided a layer of code that accounted for system differences. This design criterion has been eliminated. In Windows Server 2003, HAL deals with Intel-compatible CPU and related device-dependent issues.

Concentrating on a single architecture makes the writing of device drivers considerably more straightforward. With a published API, application developers can write instructions for a device that is optimized for Windows Server 2003. Whenever possible, system administrators should use such enhanced device drivers.

In multiprocessor systems, HAL serves an additional function of automatically synchronizing hardware-related threads with the next available CPU. Priorities range from real-time processes (with the highest priority) to variable, or dynamic, processes (lower priority), as discussed in a subsequent section.

THE MICROKERNEL

In operating systems such as Windows Server 2003 and UNIX, the base-level functions in operating system operations are managed by a kernel. In Windows Server 2003 this component takes the form of a nonconfigurable and nonpageable microkernel. By *nonconfigurable,* we mean that the microkernel is never modified and never recompiled; by *nonpageable* we mean that the 4-KB memory pages associated with the microkernel are fixed and not referred to the pagefile.sys file, where dynamic paging activities are retained.

The microkernel dispatches and controls threads. (Where multiprocessors are involved, it also synchronizes that workload.) Dispatcher objects implement and synchronize events, semaphores, timers, threads, and mutants (defined by mutually exclusive resource access). Control objects regulate virtual address processes, system interrupts, thread profiles, and asynchronous procedure calls.

DEVICE DRIVERS

A device driver is a set of instructions that coordinates the operating system with hardware such as printers, storage units, modems, network equipment, fax machines, scanners, and digital cameras. The Windows Device Model (WDM) theoretically allows a common set of device drivers. In theory, the WDM should greatly

reduce the system administration's burden of maintaining multiple device driver versions. With the release of Windows Server 2003 and Windows XP, this objective has been largely achieved for the first time.

In the case of streaming media software, the WDM has shifted the processing from the user mode to WDM Kernel Streaming. The objective was to improve overall speed and performance. An application must be specifically written for WDM to take advantage of this architecture. The same principle applies to the new WDM Still Image Architecture for specific support of digital cameras and scanners.

The space available for device drivers and system space has significantly increased. In Windows 2000, device drivers were limited to 220 MB; Windows Server 2003 supports up to 960 MB. Windows 2000 had a total system virtual address space of 660 MB, compared to the 128 GB of Windows Server 2003.

EXECUTIVE MANAGERS

The fourth segment of the Executive is a collection of tightly coupled applications. Known collectively as the Executive managers, they allow the subsystems and user applications to access system resources. Perhaps the biggest change since Windows 2000 is the inclusion of a Cache Manager, which replaces memory storage in paged pools, and the Configuration Manager, which now implements the Registry. The Executive managers are the following:

- *The Object Manager*. This manager is responsible for the creation, deletion, and interim management of object resources: files, directories, threads, processes, ports, semaphores, events, symbolic links, and shared memory.
- *The Virtual Memory Manager (VMM).* The VMM regulates the allocation of 32-bit or 64-bit linear memory. Windows 2003 supports virtual addressable memory—by default, half is allocated to system tasks and half to application workload. (Windows 2003 Enterprise and DataCenter Editions in certain configurations permit the shifting of memory allocation to allow additional dedicated memory for application support. Applications must be coded to use the Very Large Memory (VLM) APIs.) As required, the VMM pages information to disk or physical memory. It also regulates demand paging, which uses the physical hard drive to effectively expand total memory availability.
- *The Process Manager*. Windows Server support for program processes is controlled by threading. A thread is a logical sequence of instructions that is executed to completion or until a higher-priority thread temporarily preempts it. The Process Manager specifically monitors thread and related process objects. Later discussion will clarify the roles of threads and processes.
- *The Interprocess Communication Manager*. This Executive manager regulates both local procedure calls (LPCs) and remote procedure calls (RPCs). The Local Procedure Call Facility manages client and server communications within the

computer. As a local procedure that impacts system resources is launched from the user mode, the server elements in the kernel mode are called. The Remote Procedure Call Facility manages client/server communication across different computers.

- *Security Reference Monitor (SRM).* To create or gain access to an object, a request must first flow through the Security Reference Monitor (Figure 2.3). Unlike some of the other Executive managers, the SRM operates in both kernel and user modes. As discussed later, each Windows Server object has a descriptor known as the access control list (ACL). Each user and group with object rights is provided an individual access control entry (ACE) that contains its security ID (SID). Upon logging on, each user is assigned an access token that operates as a passkey to objects that match his or her entry levels. For greater detail, refer to Chapter 9, "Permissions Security, Folder Sharing, and Dfs."

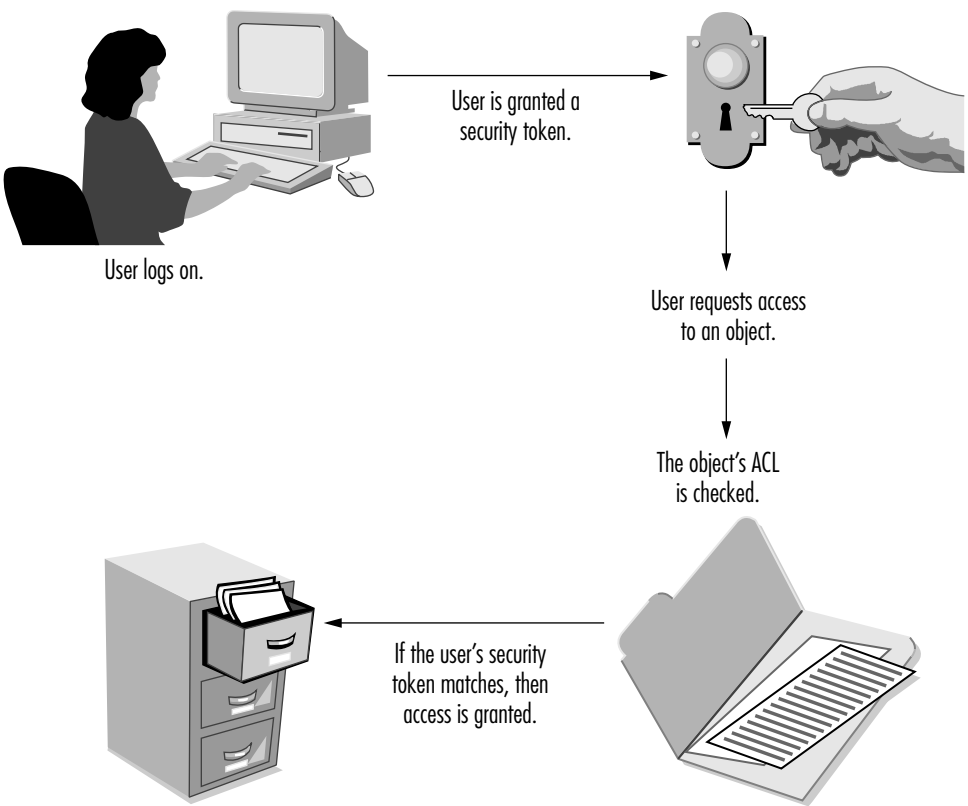


FIGURE 2.3 The Flow of Security Reference Monitoring

- *I/O Manager*. All input and output functions are controlled by the I/O Manager. These activities are broken into several components that regulate the input and output of the system cache, file system, network drivers, and specified devices (Figure 2.4).
- *Windows Manager and Graphics Device Drivers*. The Windows Manager and Graphics Device Drivers (GDDs) were moved from the user mode, where they resided in Windows NT 3.5 and earlier versions, to the kernel mode in Windows Server 2003. Applications can address the Win32K.sys interface; the GDD talks directly to HAL.
- *Plug and Play Manager*. The new Plug and Play Manager reduces the mundane system administration burden of identifying and configuring devices on the network. It activates devices and adds devices via automatic discovery or with the assistance of the Hardware Wizard.
- *Cache Manager*. This manager monitors pages faults that are required for disk reads and data already in memory. The operating system faults data and code into memory from disk in 4-KB page chunks and then releases before demand. The Cache Manager helps to prefetch pages. It also monitors the initial stages of an application startup (up to 10 seconds). The Cache Manager works in conjunction with the Task Scheduler to signal a named event, and it calls the Memory Manager to read data or code.
- *Configuration Manager.* This manager has been recoded to regulate between registry settings with the executive kernel-mode subsystem. It is designed to significantly increase the size of Registry hive (formerly 376 MB) and therefore

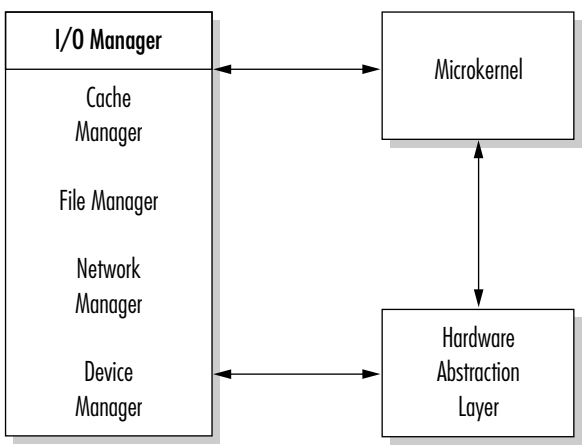


FIGURE 2.4 The I/O Manager

have no hard-coded limitations. This increases performance and adds greater support for Terminal Server systems. The Configuration Manager also plays a security role, maintaining its own cache of security descriptors that may be used by multiple keys.

- *Power Manager*. This manager regulates the power to computers and devices. In systems with power management client interfaces, Windows Server 2003 can automatically and remotely order a system to boot, shut down, or go into temporary hibernation.

THE EXECUTIVE SERVICES BUFFER

The Executive Services buffer consists of a relatively small layer of code that sits on top of the other Executive components. It separates the kernel and user modes and acts as the medium for passing API and system calls.

The Windows Server 2003 User Mode

The user mode comprises components that work together to facilitate user and application integrity. It has two parts:

- *Protected environmental subsystems*. Windows Server 2003 supports user-mode subsystems that maintain specific requirements for native Windows (16/32-bit and legacy MS-DOS), POSIX, and OS/2 applications as well as user-related system calls. An examination of protected subsystems follows.
- *Dynamic integral user intervention*. This part oversees the unprotected actions of individual users. We discuss the impact of this dynamic intervention when we deal with the processes later in this chapter.

THE PROTECTED USER MODE SYSTEM

The subsystem structure can be viewed as a buffer between user applications and the kernel-mode services structure. The term *protected* refers to these subsystems because they are not directly changed or modified by the administrator or the user but merely pass and manage API calls. They are configurable only through APIs and built-in utilities.

Windows Server 2003 supports two protected subsystems:

- The integral subsystem performs underlying operating system tasks—for example, security management.
- The environmental subsystem establishes the foundation for applications and user interfaces.

Integral Subsystems

The integral subsystems overlay and interact with the environmental subsystems. For example, the API that provides access to the network is either the Workstation Services or the Server Services subsystem, depending on the version of Windows Server installed. As another example, the integral Security subsystem acknowledges logon requests, authenticates logons, monitors the use of resources by a user, and manages user rights and permissions.

Environmental Subsystems

The user mode supports three environmental subsystems, as shown in Figure 2.5. The intent behind this was to provide support for applications originally written for other operating systems or to make porting of applications easier. Depending on the environmental subsystem, this "support" ranges from executing shrink-wrapped applications to merely providing programming APIs. With Windows Server 2003, this multiple environment subsystem support has been reduced.

Environmental subsystems may be thought of as operating system "multiple personalities." The Win32 subsystem provides native support for applications written to support Microsoft's 16- and 32-bit APIs. The other subsystems are a set of APIs that emulate other operating system calls.

- *Win32 subsystem.* Win32 is the mother of all subsystems. It supports standard Windows Server input and display output. Specifically, it controls the graphical user interface. All Win32 applications are run directly inside this subsystem. Win32 also takes on a type of arm's-length relationship with the other subsystems by switching personalities when necessary.

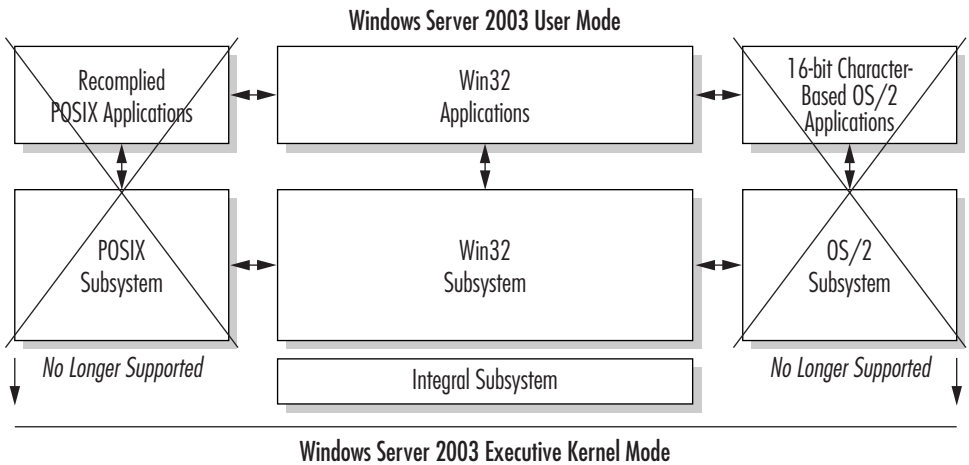


FIGURE 2.5 Windows Server 2003 Subsystem Relationships

MS-DOS and 16-bit applications use both the Virtual DOS Machine (VDM) and the Win32 system. The VDM is created automatically when these programs are launched. The application process technically runs as a VDM process, but its display handling is offloaded to Win32. Because API "stubs" support the old graphical drivers and dynamic-link libraries (DLLs), Win16 applications generally operate without affecting other operating system activities. It should be remembered that Windows Server 2003 is a preemptively multitasking operating system that supports numerous single processes simultaneously. In the case of Win16 applications, WOW, or Windows on Windows, defines the interplay between VDM and Win32.

- *OS/2 subsystem.* The OS/2 subsystem was available in Windows NT and Windows 2000 but is not supported on Windows Server 2003.
- *POSIX subsystem.* The Portable Operating System Interface computing environment subsystem was available in Windows NT and Windows 2000 but is not supported on Windows Server 2003.

For those interested in POSIX and UNIX interoperability, Microsoft has made available its Services for UNIX 3.0. This offers an assortment of third-party UNIX applications and utilities, including Korn and C shell support and NFS. By adding these features, it is possible to use many scripts written in a UNIX environment and move them directly across to Windows Server 2003. This is to be offered as part of an optional service package known as Microsoft Services for UNIX. The Interix code actually replaces Microsoft's POSIX subsystem and overlays a complete UNIX 95 environment within Windows. In this configuration, true operating system interoperability is achieved. It is also possible to migrate existing UNIX applications to Windows Server 2003 with comparative ease. Among the common UNIX features that Interix provides are the following:

- More than 300 UNIX commands and utilities
- Shell support for the Korn shell, Bourne shell, and C shell
- Scripting languages—awk, Perl, sed, Tcl/Tk—with full shell job control
- POSIX.1, POSIX.2, and ANSI C interfaces
- BSD sockets implemented with Winsock
- SVID IPC (message queues, semaphores)
- Shared memory, memory-mapped files
- ODBC and OpenGL application library support
- X11R5 Windowing System clients and libraries
- The X11R6.3 Windowing System display server

- X11R6 fonts and font management
- The OSF/Motif® 1.2.4 Window Manager and libraries
- Execution of Win32 applications from Microsoft Services for UNIX
- Full tty semantics mapped to console windows and pseudoterminal support
- Full integration with the Windows NT security model, administration, file systems, networking, and printers
- Telnetd and rlogind services (multiuser logon support)
- Berkeley r-utilities (servers and clients)

MKS and its DataFocus division's NutCRACKER development product also provide POSIX utilities and application porting directly to the Win32 API.

WINDOWS SERVER 2003 PROCESSES

Windows Server 2003 processes are viewable through a surprisingly simple yet informative set of graphical utilities. Using the Task Manager (with three equally helpful screens), the Event Viewer, the Services Manager, and the Resource-Kit-based Process Viewer, a system administrator can review and govern many of the process-oriented activities of the local server. The Task Manager can also rapidly address many performance issues. These tools effectively manage a robust process-oriented, multitasking, and multithreading operating system.

After reviewing this section, a system administrator should have sufficient baseline information to do the following:

- Understand Windows Server 2003 processes, threads, pipes, and handles.
- Use the Task Manager and Services Manager to identify, start, and kill processes.
- Use the Event Viewer to diagnose process success and failure.
- Employ the Process Viewer to ascertain priorities.
- Schedule processes.

Processes, Threads, and Handles

A Windows Server 2003 process is an executable that flows through a logical sequence of events until the appointed action is complete. Technically, an executable program is composed of the base code and related data, a dedicated memory address space, defined system resources, and at least one thread. A *thread* is the portion of the process being executed. It has a unique identification, its client ID, and a register that defines its microprocessor state. Every thread

maintains reserved memory stacks for the execution in the user and kernel modes. It also has storage memory for interaction with other applications, DLLs, run-time libraries, and environmental subsystems.

Processes use threads to invoke an action with a reaction, pipes to connect threads, and semaphores to synchronize activities. Unlike UNIX processes, which involve exec/fork replication and parent/child relationships, Windows Server 2003 utilizes handles. Handles are assigned to threads and identify resources such as a Registry key used for access by a program. (Handles are also applied to events, semaphores, pipes, processes, and communications.)

A Windows Server 2003 process is treated as an object. As such it has a number of characteristics, including a virtual memory address, defined resources, and a security profile. Each process has one or more thread objects associated with its execution. As an object, the thread also has a unique memory stack and system state. The thread is an agent that does the bidding of the process. The Object Manager controls both process and thread objects.

When a new process is created, the CreateProcess() and CreateThread() calls are made. As additional threads are required to support a given process, other CreateThread() calls are invoked. The thread should be thought of as a unit of execution. Between the ends of the thread is a pipe, both ends of which must be open. If either end is broken, the process data is lost.

Windows Server 2003 also uses named pipes to transmit information. These pipes are viewed similarly to file objects and operate within the same security framework. A named pipe retains information in memory and dispenses the data as requested by a process. It acts like a regular data file, except that the information is in resident memory, not physically archived on disk. Figure 2.6 illustrates the hierarchy of processes.

Windows Server 2003 is a multitasking and multithreaded operating system. One of its strengths is its ability to manage and synchronize multiple threads. On a single-processor system, only one thread can execute at a time, but, as a result of context switching, it appears to the user that multiple threads are running. In this scheme (Figure 2.7), a thread executes until it has completed its task, is in-

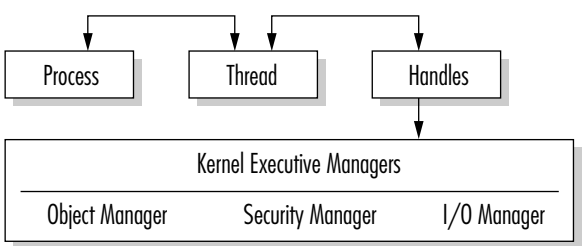


FIGURE 2.6 Process Hierarchy

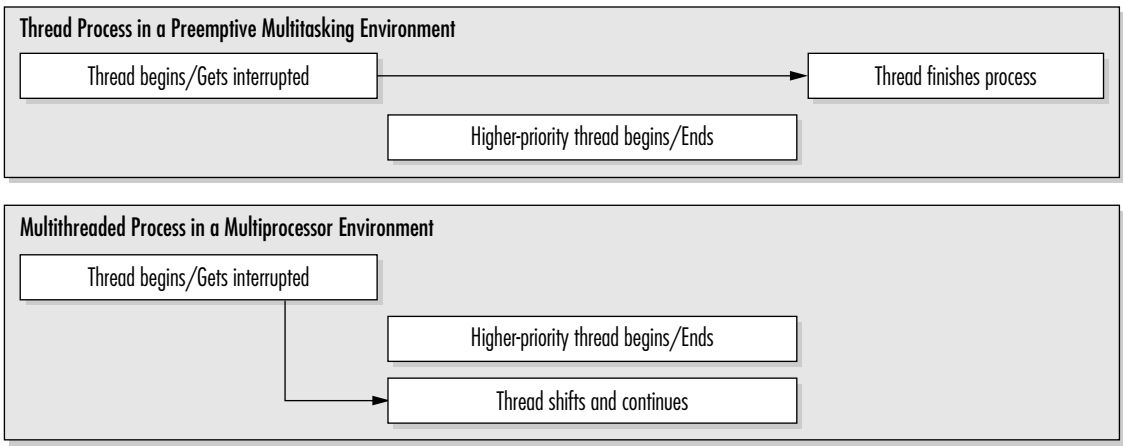


FIGURE 2.7 Single and Multiprocessor Preemptive Multitasking

terrupted by a thread with a higher priority, or waits for system resources. During an interruption, the Windows Server saves the "context" of the thread and reloads it when the CPU is free to continue.

Windows Server 2003 provides a robust process and thread priority facility. The programming API permits the setting of application process priorities, or states, that include idle, normal, high, and real-time; within each of these states two subpriorities can be established. It is important for a programmer to consider the impact of this facility on the entire system. There are 32 priority levels numbered 0 through 31. The first 16 (0–15) are reserved for the user mode, and the remainder are reserved for the kernel mode. After a base priority level is set for a process or a thread, the kernel may dynamically raise it in the event of user interaction. By the same token, the kernel can lower the priority rating for strictly computer-associated operations.

How long a thread is allowed to execute is controlled by a property known as a quantum. A quantum type can be of fixed or variable length. By default, Windows Server 2003 versions give priority to fixed-quanta background services, although Windows XP gives higher priority to applications that generally use variable-length quanta. In theory, these priorities provide greater smoothness to key multitasking functions of the respective environments.

MULTIPROCESSOR SUPPORT

Where processing load is significant, systems with more than one CPU can reduce thread executing wait times. Windows Server 2003 uses symmetric multiprocessing (SMP), which allows user and kernel-mode activities to use any available processor.

Symmetric multiprocessor systems have identical processors and functions. (By contrast, an asymmetric multiprocessor system allocates resources to a specific processor even if that CPU is overloaded and others are relatively free.) The clear advantage is the balancing of the processing load across all resources.

Process Accounting and CPU Throttling

With its Internet Information Services, Windows Server 2003 adds two features that make multiuser environments more efficient. The first is processor accounting, which measures the number of processor cycles consumed by Web requests. The second is CPU throttling, which restricts Web applications from overwhelming CPU time. Both are enabled by job objects, which allow the operating system to manage groups of processes as a single unit. The use and administration of process accounting and CPU throttling are discussed in Chapter 16.

Interprocess Communication

Interprocess communication (IPC) and remote procedure call (RPC) are basic to Windows Server 2003. (At the risk of being overly simplistic, RPC can be regarded as the distributed network version of IPC.) Microsoft claims that its implementation of RPC is compliant with Distributing Computing Environment (DCE).

Windows Server 2003 employs a direct relationship between processes and threads. At any given time, there is a producer and a consumer (similar to BSD socket library calls in UNIX). The objective is to achieve a very scalable operating system that permits easy load balancing of discrete measured threads. Windows Servers also employs a FIFO (first-in, first-out) message model.

Windows Server 2003 Component Services

A Windows Server 2003 service is a special class of process with fixed characteristics. The Component Services Manger is similar to the UNIX daemon in that they perform tasks defined as in either the foreground or the background. They can be managed either locally or remotely (Figure 2.8).

The Windows Server 2003 Component Services Manager provides the list of all defined services, their current status, and their status at startup. These services can be started, stopped, paused, continued, or defined for system startup with a simple mouse click.

To view or change the status of a service, press the **Start** menu → select **Programs** → select **Administrative Tools** → select **Computer Services**. In the left pane, click **Services**. All defined services on the system will appear in the right pane. Double-click a target service to view its properties. You can make changes or simply view the current status via four tabs: General, Logon, Recovery, and Dependencies.

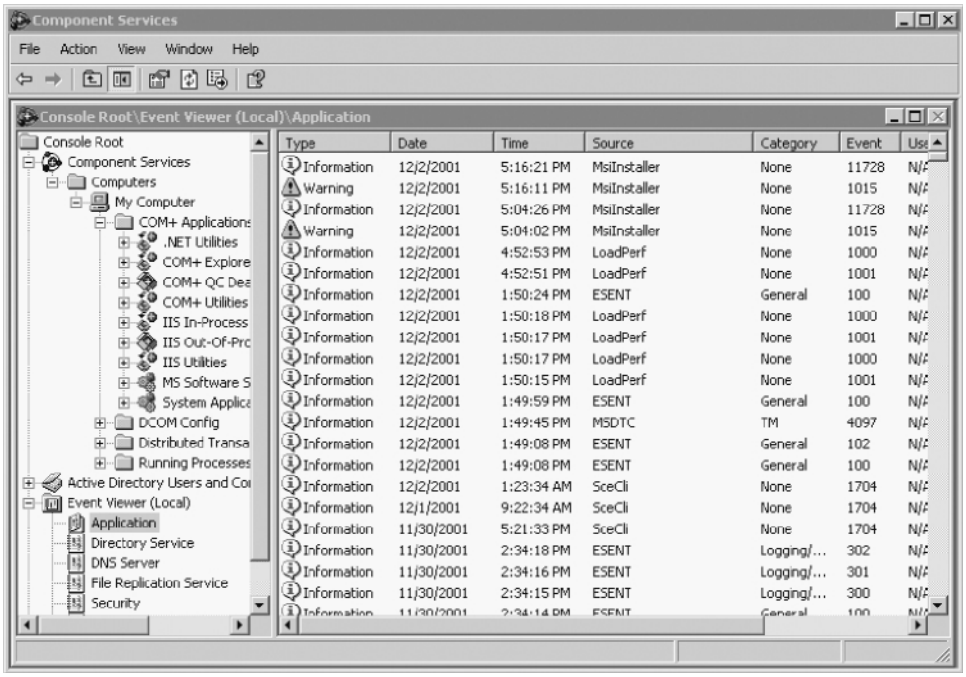


FIGURE 2.8 Windows Server 2003 Component Service Manager

The Task Manager

The Task Manager is a handy system administration tool that provides a snapshot of computer system activity. It is readily available to an administrator at any point during a logon session. The two most common methods of executing the Task Manager are:

1. Click the right mouse button on the **Task** bar and select **Task Manager**.
2. Press CTRL+ALT+DEL and select **Task Manager** from the Windows Server Security window.

The Task Manager provides four window tabs for Windows XP and five tabs for Windows Server 2003 and provides very different views of operating system process activities:

- The Applications window permits the view and control of all user-mode applications. Applications may be responsible for one or more processes (Figure 2.9).

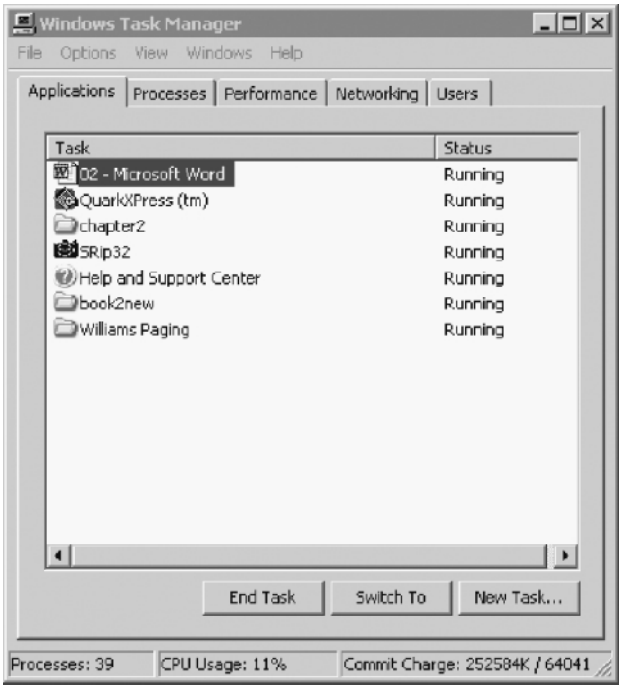


FIGURE 2.9 The Task Manager Applications Window

- The Processes window provides in-depth information about individual processes, including resource utilization.
- The Performance window provides a graphical and numeric view of the system based on the current application and process load.
- The Networking tab identifies the amount of current network activity on the computer.
- The Users tab (Windows Server 2003 only) identifies local users and remote users by name, ID, activity status, and client system name.

A simple dissection of the Applications window shows the application name in the first column followed by the current status. The bottom set of buttons permits the administrator to **End Task**, **Switch to** another task, or invoke a **New Task.** At the bottom of the screen is a system summary status that includes a number of open processes, CPU usage, and memory utilization.

FIGURE 2.10 The New Task Dialog Box

STARTING AND KILLING PROCESSES

Invoking an application from an icon or command-line instruction ordinarily starts processes. As discussed in a later section, processes can also be initiated through scheduling services like the At utility. The Task Manager can start a new task as well if this option is selected at the bottom of the screen. When **New Task** is selected, a new dialog box, such as that shown in Figure 2.10, appears.

Ordinarily, processes are terminated by the graceful exit of the application through its own kill signals. However, from time to time more radical solutions are required. Processes can be easily killed using the Task Manager. In the Applications window, simply highlight the offending process and select **End Task** at the bottom of the screen. In the Processes window, select **End Process**, also at the bottom of the screen. In both cases, a confirming message will appear before termination. At this point, all handles, threads, and pipes are destroyed and data being transmitted is lost.

Figure 2.11 provides a real-time view of network utilization on the local client or server system.

NOTE Administrators coming from the UNIX environment occasionally face a runaway process known as a zombie. Microsoft claims that zombies will not occur in Windows Server 2003 because its processes do not use the parent/child relationship inherent in UNIX. Each Windows Server process is discrete and independent of the presence or health of a parent. While this may be technically true, certainly some processes (generated by either Microsoft or third-party software) begin sapping very large amounts of memory and become very difficult to kill. Use the Task Manager or the Process Viewer to kill such processes. On rare occasions, you may have to invoke the action more than once.

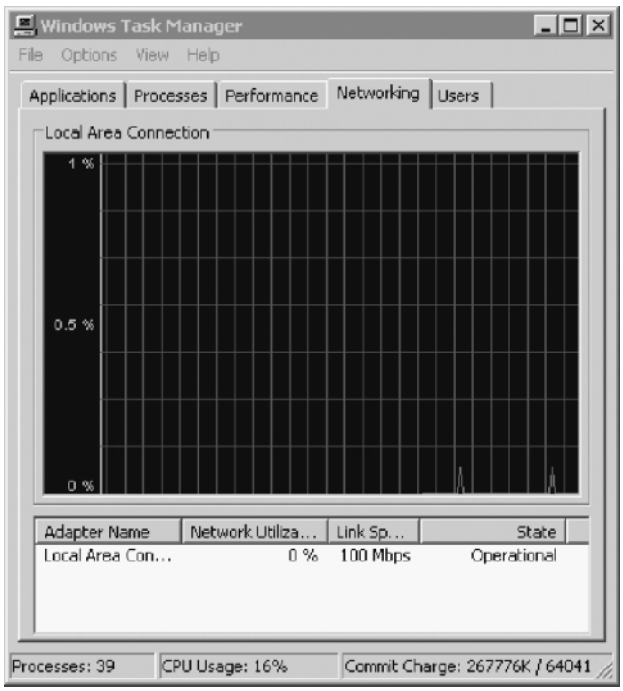


FIGURE 2.11 Task Manager Local Area Connection Monitor

SCHEDULING TASKS

Early versions of Windows NT supported the UNIX-like utility known as the **at** command. This command is still supported, but its functions are greatly enhanced by the Task Scheduler. The Task Scheduler and the **at** command can work together, which is important for older scripts that rely on **at**. When a task is invoked by the **at** command, it appears in the list in the Scheduled Tasks window. The Task Scheduler also can use **at** to run tasks automatically.

The **at** program is invoked from the CMD command line for single-event scheduling. The **at** command schedules commands and programs to run on a computer at a specific time and date. The Schedule service must be running to use the **at** command.

Scheduled Tasks are accessible from Control Panel. The Scheduled Tasks Setup (Figure 2.12) is invoked by double-clicking this icon. Tasks are added by dragging scripts, programs, or documents from Windows Explorer or the desktop to the Scheduled Tasks window. They can also be deleted or modified this way.

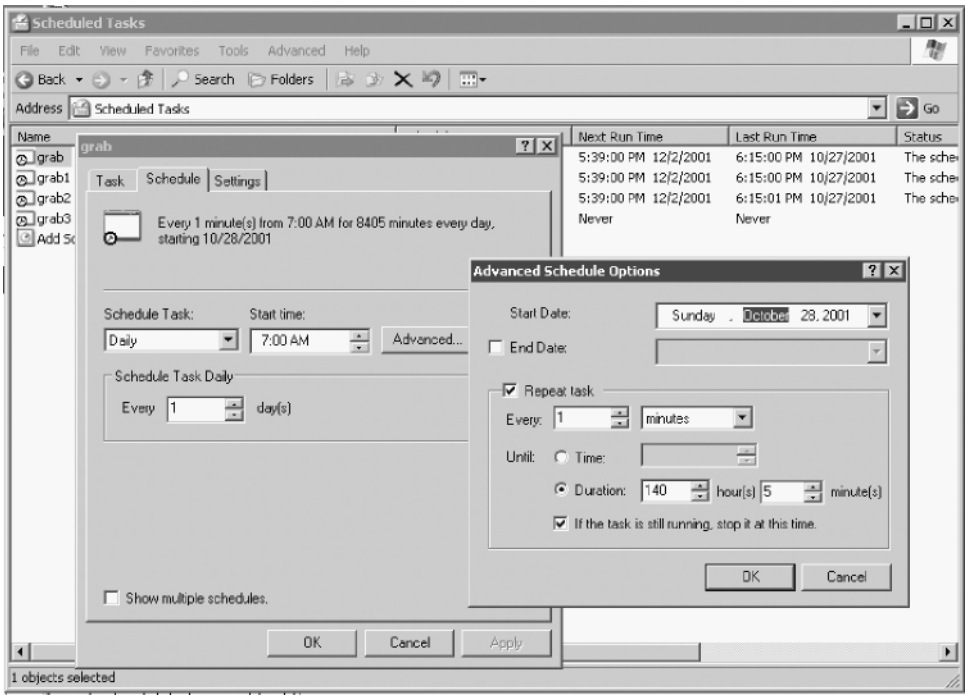


FIGURE 2.12 The Scheduled Tasks Setup Window

VIEWING PROCESSES

The Processes window provides system administrators with a very good snapshot of their system's activity. Its default screen has five process checkpoints: Image Name, User Name, Session ID, CPU, and MEM Usage (Figure 2.13). A total of 22 items may be selected for analysis by changing **Select Columns** in the **View** menu. When the dialog box appears, use the mouse to click the additional reporting parameters. If all options are selected, the system administrator has a reasonably comprehensive view of active processes and their impact on the overall system, as shown in Figure 2.14.

The Task Manager Performance screen (Figure 2.15) displays four graphical representations and four boxes of numeric data for the entire system:

- *CPU Usage* shows current CPU utilization (in this example, the CPU has reached a critical utilization state).
- *CPU Usage History* is a longer-term view generally showing CPU usage over time. Unfortunately, in this window there is no ability to change time intervals or to save or print this information. A tool to refine this view can be accessed by pressing **Start → Programs → Administrative Tools → Performance**.

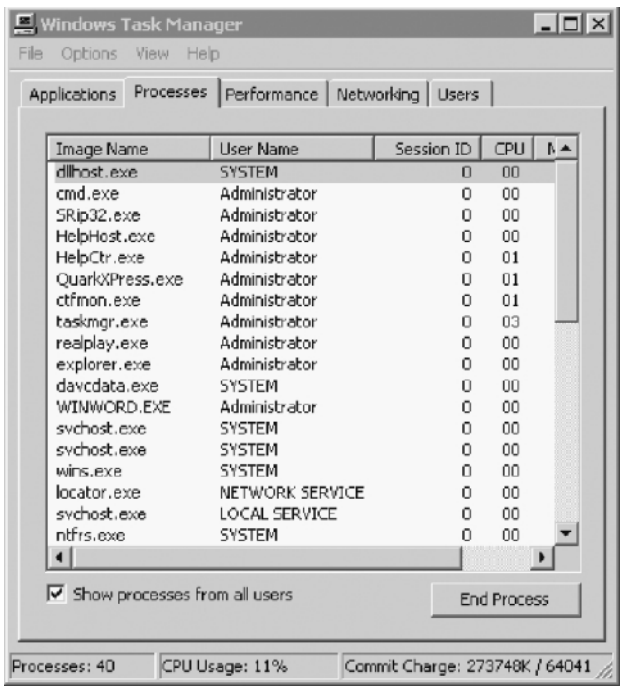


FIGURE 2.13 The Task Manager Process List

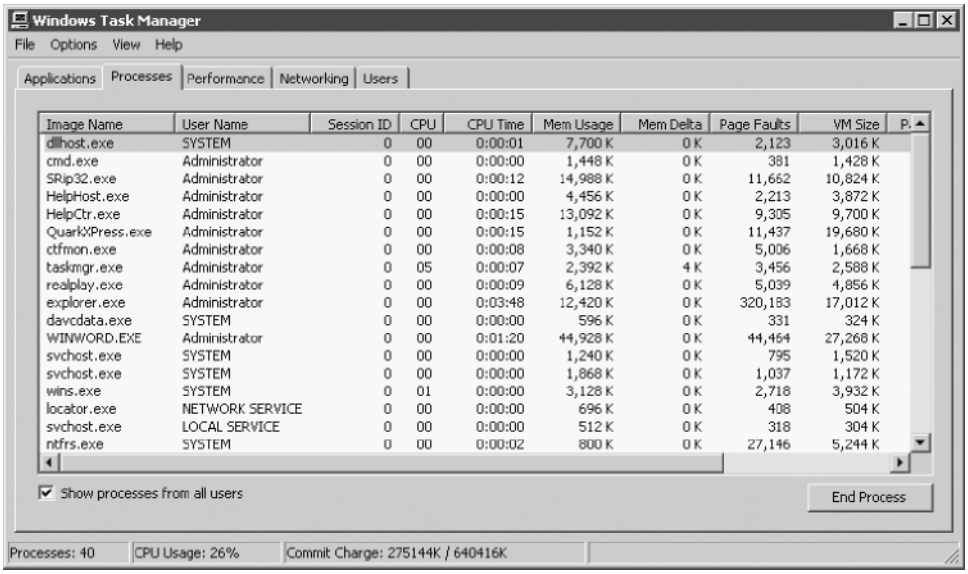


FIGURE 2.14 Expanded View of Process Activities

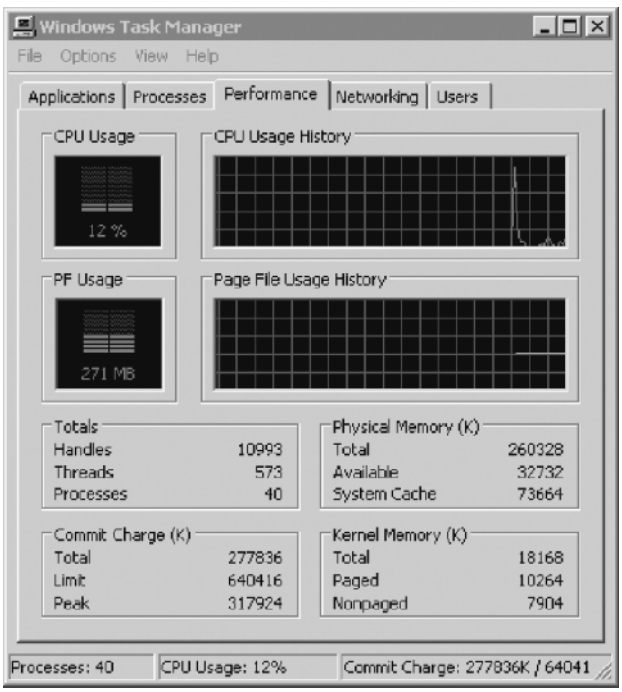


FIGURE 2.15 The Task Manager Performance Window

- *PF Usage* shows utilization based on pagefiles.
- *PF History* shows fluctuation of pagefile usage over time. You can refine this view by pressing **Start → Programs → Administrative Tools → Performance**.
- *Totals* shows the number of open handles, threads, and processes.
- *Physical Memory* summarizes the system memory defined as a total, currently available, and cache available in kilobytes.
- *Commit Charge* shows the total memory allocated to programs, the system limit, and peak memory, including swap memory.
- *Kernel Memory* defines the total memory allocated to kernel-mode activities, broken down by paged and nonpaged events.

New performance tools available with Windows Server 2003 include:

- Disk input and output (I/O)
- Memory management (e.g., working set management, page fault)
- Image load and unload
- Process and thread activities such as process create, thread create, and context switching
- Registry
- Driver delays
- Pool allocations
- Heap allocations
- Central profiling that examines both kernel-mode and user-mode activities

Monitoring processor activity is effectively accomplished by looking at the Process and System object counters. In addition to providing information on utilization, it can pinpoint bottlenecks. The specific items you should examine are:

- Processor\% Processor Time for processor usage
- Optionally, you can also monitor Processor\% User Time and % Privileged Time along with % Processor Time for more detail
- System\Processor Queue Length for bottleneck detection

The Processor\% Processor Time counter is used to gauge the activity of the processor. Look specifically at the counter showing the percentage of elapsed time that a processor is busy executing a non-idle thread.

An examination of processor usage is not straightforward. For example, high processor values could indicate either the efficient handling of a heavy workload or a struggle to keep up. Typically, you will need to understand which appli-

cations cause peak processor loads such as software that involves heavy computations. When these applications are churning, anticipate peak loads. If this becomes a problem that impacts other applications, then workload scheduling is appropriate. By contrast, if the processor is peaking because many clients are using the system, then consider adding system resources or offloading some users to another system.

A processor bottleneck simply occurs when too many threads place demands on the processor. A processor queue develops as the threads back up. CPU-bound applications and driver components are notorious for creating bottlenecks. The System\Processor Queue Length counter is used to determine the existence of bottlenecks. A queue of two or more items indicates a bottleneck over time (a snapshot of a number higher than 4 should also be considered a critical bottleneck number.) When this occurs, you have only a couple of options. From a hardware perspective, you can get a faster processor or add a processor. Alternatively, you can restrict application and user activity.

Interrupt activities can also cause a bottleneck. The Processor\Interrupts/sec counter measures the rate of service requests from I/O devices. An increase of the counter value without a corresponding jump in system activity suggests a hardware problem. The Processor\% Interrupt Time is also a modestly good indicator of the interrupt activity of disk drivers, network adapters, and other devices.

NOTE Windows Server 2003 has new Performance security groups. Members of the Performance Monitor Users group can monitor performance counters on the server locally and from remote clients. Members of the Performance Log Users group can manage performance counters, logs, and alerts on the server locally and from remote clients. These features permit the delegation of responsibilities without granting full administrative rights.

Performance Monitoring of Multiple Servers

Windows Server 2003 allows administrators to collect performance counter data for a group of servers. All performance objects, counters, and instances can be utilized.

The select multiple objects in Performance Monitor (PerfMon) permit visualization and analysis of performance data. Specifically, this feature supports log monitoring and alerts services. To invoke this feature: **Start → Control Panel → Performance and Maintenance → Administrative Tools → Performance → System Monitor**.

It is important to note that this feature is not available in SysMon for real-time monitors because there is too much data.

Command-Line Options

Windows Server 2003 adds performance control command-line tools. They include functions that provide management of the control performance counter, event trace log scheduling, and collection and analysis on local and remote computers. These functions are accomplished without the need for the System Monitor (Sysmon) UI. The new tools and the functions they allow managers to perform are:

- *Logman*—Start, stop, query, and schedule performance data and event trace log collection.
- *Relog*—Given specified intervals, sample performance counter logs to extract specific counters and write to a new log file in a different format.
- *Tracerpt*—Report and analyze process kernel, Active Directory, and other transaction-based trace event logs.
- *Typeperf*—Write performance data to the command line every "*n*" samples.

These command-line tools are available by typing the command in the command prompt window.

THE EVENT VIEWER

The Event Viewer provides a defined log for both normal and abnormal system events. It can be one of the most valuable system administrative tools for troubleshooting problems. A regular analysis of these logs should be undertaken if system problems are suspected.

In Windows an *event* is any system occurrence that requires notification. One type of event recorded in the Event Viewer system log is a dysfunctional driver or load failure (Figure 2.16). The Source column identifies the application that initiated the message. The Category classifies the event as being related to security, object access, logon/logoff, detail tracking, system, policy changes, account management, and miscellaneous (none).

The level of importance of an event message is shown as an icon in the extreme left column: Error, Informational, Warning, Success Audit, and Failure Audit. This application can be launched from the Event Viewer for the local system by clicking the **Start** menu → selecting **Run** → typing **eventvwr** → clicking **Enter**. Alternatively, the Event Viewer is available from Administrative Tools.

The Event Viewer has a very flexible set of options that permits long-term tracking of system, security, and application problems. These options are described in Table 2.1.

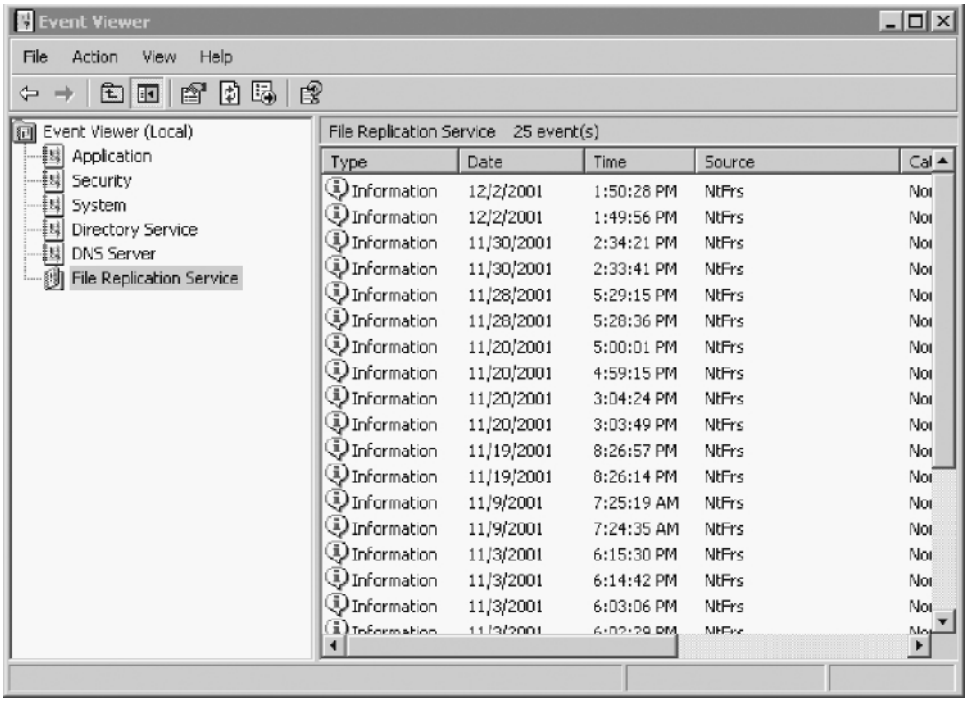


FIGURE 2.16 The Event Viewer Screen

To review details, highlight the item in the right detail panel and double-click. The appropriate event properties window will then be displayed for analysis and action. As seen in Figure 2.17, a significant amount of information is provided to

TABLE 2.1 Event Viewer Options

Option	Description
Logging	Rolls events as needed; removes events after a defined number of days; prohibits overwriting of events (requires manual event removal).
Sorting	Sorts events based on categories, sources, user, or machine.
Archiving	Saves event records in native Event format (*.EVT) or in normal or comma-delimited ASCII text (*.TXT) format.
Details	Provides additional information on the event.
Filtering	Filters desired events or characteristics.
Find	Locates events based on type, category, source, computer, or other criteria.

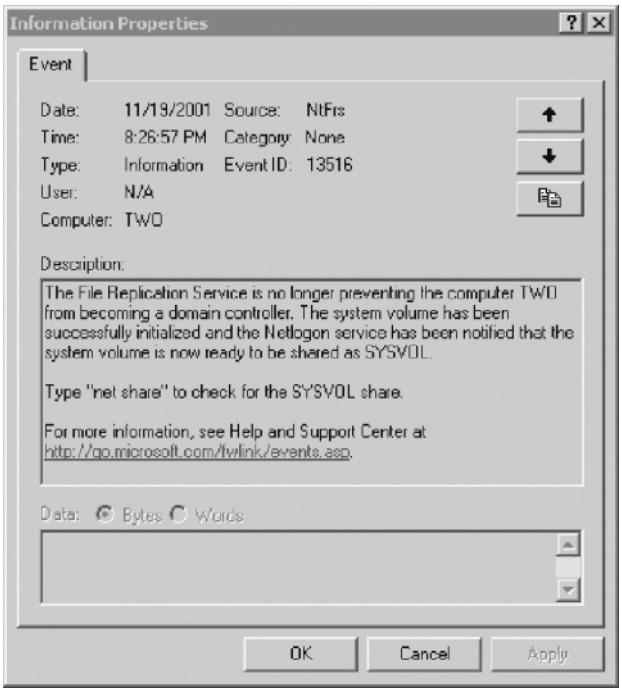


FIGURE 2.17 Event Log Properties

permit zeroing in on the precise location of the event. If a warning or error occurs, appropriate action can then be taken.

NOTE In Windows 2000, the Event Viewer feature was introduced that allowed Event Log error messages to manually add hyperlinks to Web sites by changing .RC and .MC files. Windows Server 2003 automatically adds a standard phrase and a Uniform Resource Locator (URL) to each message with the message properties dialog initialization. In addition, an optional string value can be added to the registry to specify the target URL.

NOTE System logs are essential to system performance analysis. Windows Server 2003 provides the generation of Perf Logs to the size of 1 GB. With a new file format, these logs can be appended to an existing log file.

STORED AND VIRTUAL MEMORY

Stored memory is the storage of data bits on devices such as a hard drive. Its management is predicated on the type of file system being used. The role and dynamics of storage for Windows Servers vary radically among the native NT File System (NTFS), the file allocation table (FAT), and FAT32. In general, stored memory is designated as "storable" and "retrievable archival." All three file system types are supported on x86 computer systems.

Virtual memory is defined as dynamic or temporary. *Physical memory,* known as random access memory (RAM), combines with temporary disk storage. For example, if a process requires more memory than is available from the system's RAM, the hard drive is used to temporarily store process data. It uses a set of temporary registers where information about processes is stored during execution. This is known as *paging*.

Stored Memory and File System Basics

Windows Server 2003 supports three file systems, NTFS, FAT, and FAT32, each of which has unique characteristics. For base-level security we have already established the clear advantage of native NTFS. However, under some circumstances, such as a need to switch between older Windows versions and Windows Server 2003, the FAT file system is necessary. FAT32 is an expanded version of FAT that allows larger disk partitions and is used primarily in the Windows 98 operating system. The relative merits of the three file systems are examined in detail in Chapter 14, but a brief review is appropriate at this stage.

THE FILE ALLOCATION TABLE: FAT AND FAT32

For the sake of this discussion, FAT and FAT32 are viewed as the same system except where specifically noted.

A file allocation table, FAT, is a simple file system that dates back to the days when the 5 1/4-inch floppy drive was the default medium and a 10-MB hard drive was an expensive luxury. In many cases, a system administrator may install Windows Server 2003 using FAT because of legacy MS-DOS or Windows 3.x applications that simply have not been updated to take advantage of Win32 functionality. FAT32, introduced to support Windows 98, is simply an extension of FAT with additional 32-bit application support.

A number of limitations, such as reduced security, are inherent in FAT and FAT32. Even so, these file systems are acceptable for legacy environments. One very nice feature Microsoft has included is the ability to convert the file system from FAT to NTFS (but not vice versa) when appropriate. This is accomplished by

invoking the **convert** command from the Virtual DOS Machine with the following syntax:

```
CONVERT [drive:] /fs:ntfs [/nametable:filename]
Where filename is the name of file that is created during the
conversion process that contains a name translation table for unusual
filenames.
```

The file allocation table includes the following basic information about a file:

- The name of the file or directory (folder) with a structured maximum 8-character name and a 3-character mandatory extension, known as the 8.3 format
- A pointer to the physical location of the first byte of data associated with the file
- The size of the file in bytes
- The designation of the file's attributes as hidden, read-only, archive, or system-based

In the context of file storage on a disk or partition, the FAT hierarchy is very simple (Figure 2.18). The first sector contains the machine BIOS block followed by the file allocation table and a duplicate image. The root directory is the first set of information visible to a user. Individual files are stored in the root directory or in created subdirectories.

THE NTFS FILE SYSTEM

NTFS was introduced with Windows NT 3.1 as an outgrowth of the original OS/2 High-Performance File System (HPFS). With the introduction of Windows NT Server 4.0, the NTFS file system has taken a giant leap forward in terms of stability

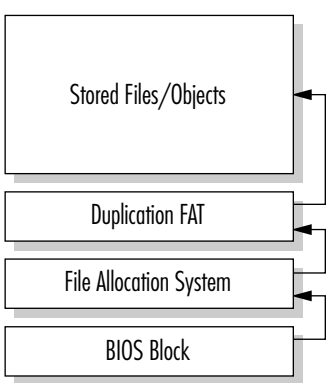


FIGURE 2.18 Disk Structure of the FAT File System

and flexibility. It boasts numerous advantages over the FAT file system, including the following:

- *Superior security* extends to files, processes, and user access protection.
- *Fault tolerance* uses a redundant array of inexpensive disks (RAID 1-5) to create duplicates of all files on servers. If a bad partition is detected, for example, Windows Server 2003 can use these copies to create a new sector.
- *Unicode* supports file names, dedicating 16 bits per character with a maximum of 255 characters. This facilitates internationalization. FAT supports only 8 characters using 7- or 8-bit ASCII and ANSI standards.
- *Master File Table redundancy* ensures recovery of system data in the case of corruption.
- *32-bit virtual memory support* has 2 GB allocated to system requirements and 2 GB to applications.
- *File name length* is 255 characters but also automatically generates the shorter DOS files names for backward compatibility.

NOTE Windows Server 2003 tightens the security of NTFS by providing only the administrator the right to write to the root directory. In previous versions, Everyone has full control in the root directory. The tightened security also prevents anyone other than administrators from modifying files in the root directory that were created by other users.

The NTFS file system maintains a number of key files that should not be removed or unnecessarily modified. Table 2.2 summarizes several of the most important keys.

TABLE 2.2 NTFS Files

File	Description
$	Root directory name
$Bitmap	Bitmap representation used by MFT for tracking volume contents
$Boot	Boot file listing bootable volumes
$Mft	Master File Table (MFT)
$MftMirr	MFT mirror
$Volume	Volume name and version information

NTFS Version 5 Differences

The Windows Server 2003 implementation of NTFS is version 5, which differs in a number of ways from the NTFS used in Windows NT for versions before Service Package 4. It is not backward compatible. Rather, during installation, if NTFS 4.0 is detected on a disk, it is upgraded to version 5.

One of the principal differences between NTFS version 5 and earlier versions is that NTFS 5 can undertake something akin to symbolic linking through the use of reparse points. The *reparse point* is 16 KB of data that takes the form of a tag that identifies the device driver. When a Windows Server 2003 NTFS process sees a reparse point tag, it redirects the action to the defined device driver. A *symbolic link* is a special type of reparse point used to redirect the NTFS process to a specified file or directory.

NTFS 5 has a useful link-tracking facility. Anytime a linked item is moved anywhere within the same domain, the link is tracked and remains intact.

NTFS 5 also supports disk quotas. These quotas are important because they permit regulation of disk usage. The specific management of disk quotas is discussed in Chapter 14.

Another important enhancement with NTFS 5 is support for the Encrypting File System (EFS). When a user wants to encrypt a file, a file encryption key (FEK) is generated. The Windows Server 2003 EFS uses an extended variant of the Data Encryption Standard algorithm (DESX).

NTFS 5 expands support of the CD-ROM File System (CDFS) that was based on a read-only principle. It now supports the Universal Data Format (UDF), which is emerging as the DVD-ROM format.

Finally, NTFS 5 supports FAT32 drives. Windows NT should not interpret FAT32 partitions or drives, nor can it boot from a FAT32 partition. Windows Server 2003 NTFS overcomes both of these limitations.

FILE OBJECTS IN THE NTFS FILE SYSTEM

NTFS boasts a relational database structure that uses B-trees to organize and stream file object management. Microsoft publishes precious little information about the internal structure of this database, known as the Master File Table (MFT), although it is the first line in an NTFS partition. The *MFT* is in a mirrored format for redundancy. Pointers from the boot sector describe the location of the MFT and its mirror.

Think of the MFT as the trunk of the B-tree in which small files and directory data are stored in residence (Figure 2.19). Large files and directories are stored as nonresident in extents (branch extensions) or runs (runners). Nonresident data can cross multiple extents or runs.

File and directory objects are similar in structure and use the same form for defining information sometimes known as metadata; however, their back ends dif-

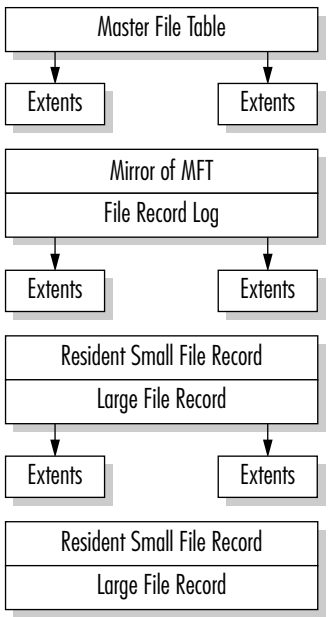
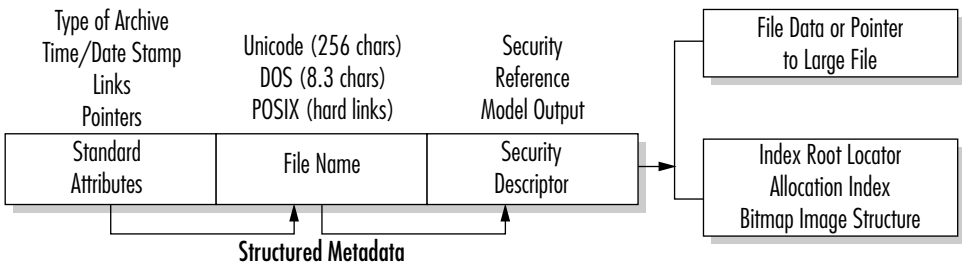


FIGURE 2.19 A Simplified View of the MFT Structure

fer. The NTFS treats directory objects as a special form of file object. Both files and directories contain standard attributes, a file name, and security descriptors (Figure 2.20). Files also include raw data that is otherwise not managed by the operating system. The directory object holds indexes of the relative location and size of the directory, plus a bitmap representation of the directory structure. The term *bitmap* does not refer to the kind of bitmap graphic commonly known to any PC user; here the bitmap is utilized by the database as a kind of road map depiction of the structure.



FIGIRE 2.20 The NTFS View of Files and Directories

File names follow a specific convention depending on the file system; FAT has an 8-character maximum with a 3-character extension (8.3) and NTFS has a maximum of 255 characters. Directories are separated in all cases with the backslash (\). Windows NT directories are similar to UNIX directories in that they are a special form of file object that merely maintains information about the files they manage.

The NTFS database structure facilitates system recovery, which is not available with the file allocation table. NTFS's transaction database methodology employs a logging process to initiate disk-writing instructions. Data enters cache memory, where it is then directed to the VMM for background or lazy writing to disk. The VMM sends the data to the fault-tolerant driver (assuming the existence of a RAID structure), which then attempts to physically write to the drive. If it is successful, the transaction in the database is released. If not successful, the record of the logged process is retained in the transaction table until restoration.

The Virtual Memory Manager and Paging

Windows Server 2003 uses a flat 32-bit virtual and linear memory scheme, reserving half of the capacity for system resources and half for applications. It allows tuning of addressable memory for applications, which was unavailable to Windows NT. For information on how this is accomplished for your version of the operating system, examine the Help files. The VMM allocates each process an address space, which is then mapped to physical memory in 4-KB pages (Figure 2.21). As memory is required, the system swaps between physical memory (hardware RAM) and pageable memory stored on the hard drive.

The VMM provides memory mapping through a table that defines virtual addresses. It also regulates the paging process—the movement from physical to disk memory. The area on a disk drive where paging occurs is known as the *pagefile*, the proper sizing of which can be critical to overall system performance. By default, Windows Server 2003 sets up a pagefile equal to the amount of installed RAM plus 12 MB (unless the available disk space is less than that formula). The mini-

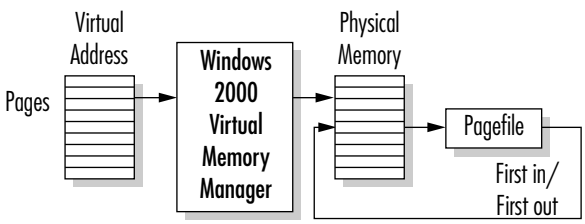


FIGURE 2.21 Virtual Memory and Paging

mum pagefile size is 2 MB. Since hard drive space is relatively inexpensive, we recommend expanding the pagefile if system performance becomes an issue.

Pagefile size is changed in the following steps after you log on as the administrator: Open **Control Panel** → double-click the **System** icon → select **Advanced System Properties** → click **Performance** → click **Change** → if multiple drives are listed in the **Drive** box, select the drive containing the pagefile → in the **Initial Size** box, increase or decrease the amount → click **Set** → click **OK** to close the **Virtual Memory** box → click **OK** on the **Performance Options** box → click **OK** on the **Systems Property** box. The computer must be restarted for the change to take effect.

THE BOOT PROCESS

The Windows Server 2003 booting sequence involves a series of system configuration checks, hardware activation, and application loading. The following lists the boot process:

- *Power on self-test (POST)* checks whether key hardware elements are present, such as sufficient memory, keyboard, video card, and so forth.
- *Startup initialization* checks the hard drive's first sector for the Master Boot Record (MBR) and partition table. If either is missing or corrupted, the process terminates.
- *Memory switch and system driver load.* An operating system load driver, Ntldr, instructs the microprocessor to convert from real mode to protected memory mode used by Windows Server 2003. Appropriate system drivers are then started that are actually built into Ntldr and that identify the type(s) of file system(s) present.
- The *boot loader* permits the selection of the desired environment where there are multiple operating systems on the system. The system looks for Ntldr in the root directory (by reading Boot.ini and invoking Ntdetect.com and Bootsect.dos). A typical boot loader screen (for a dual-boot environment) is shown here. (A single boot screen would show only Windows Server.)

```
Please select the operating system to start:
    Windows Server 2003 (or Standard, Enterprise, DataCenter, Web)
    Windows 2000
Use ≠ and Ø to move the highlight to your choice.
Click Enter to choose.
Seconds until highlighted choice will be started automatically is: 23
```

Boot.ini is divided into two sections: boot loader and operating systems. The boot loader defines the number of seconds before an automatic load of the default

operating system (whose path is defined in the next line). The operating system lists the environments available for initiation. In this case, two modes of Windows Server 2003 and Windows 98 are available.

The Boot.ini file is set as read-only, system, and hidden to prevent unwanted editing. To change the Boot.ini timeout and default settings, use the **System** option in **Control Panel** from the **Advanced** tab and select **Startup** (Figure 2.22).

NOTE If Windows 98 rather than Windows Server 2003 is selected at this stage, Ntldr executes Bootsect.dos. This file contains an image of the boot sector as it existed prior to installation of Windows Server. At this stage, the boot processes inherent in Windows 98 take control.

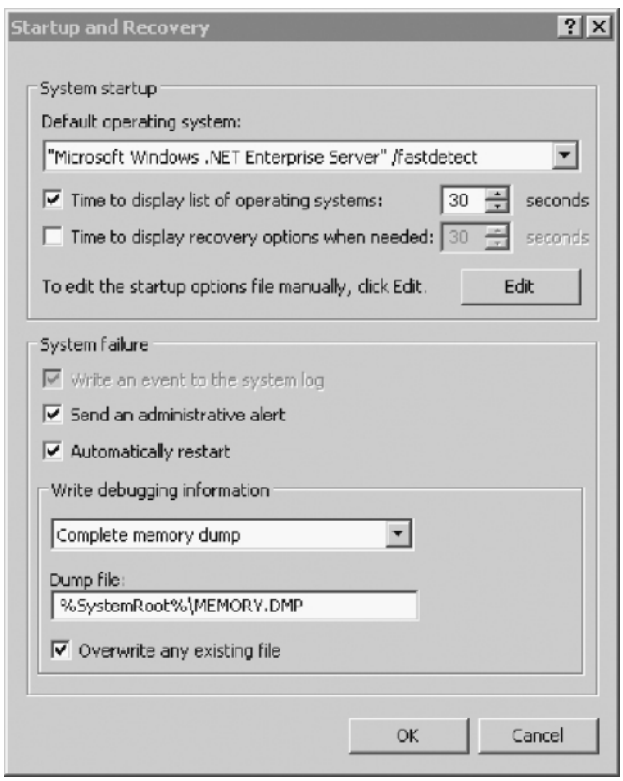


FIGURE 2.22 The System Option Screen to Change Boot.ini Timing and Default

CAUTION Dual booting, although obviously possible, is generally not recommended. A number of security issues arise in a dual-boot environment. The dual boot is generally used only if a legacy application is not supported by Windows Server 2003.

- *The hardware configuration review*. At this stage, information about the system and attached devices is gathered by Ntdetect.com. This information is later included in the Windows Server Registry as HKEY_LOCAL_MACHINE\HARDWARE.
- *Kernel load and preliminary initialization*. As dots are drawn across the computer screen, Windows Server loads the kernel with Ntosknrl.exe and the hardware abstraction layer's Hal.dll. The Windows Server Registry HKEY-LOCAL_MACHINE\SYSTEM is then loaded. Ntldr loads device drivers with a value of 0x0 and is specified in the Registry as HKEY_LOCAL_MACHINE\SYSTEM.
- *Kernel initialization*. The actual initialization of the kernel occurs when the blue screen appears and identifies the Windows Server version and build number and your system configuration. Behind the scenes several activities occur. First, a Registry key for hardware is created from the gathered information as HKEY_LOCAL_MACHINE\HARDWARE. This is where hardware system specs and device interrupts are stored. A clone set of system references is created and remains unchanged within the Registry HKEY_LOCAL_MACHINE\SYSTEM\Select subkey. The kernel initializes devices after scanning the Registry for HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services.

NOTE Depending on severity, the boot process may continue or fail if errors are detected in the initialization of devices. There are four levels of error control values. With an ErrorValue of 0x0, the boot process continues uninterrupted. An ErrorValue of 0x1 is not sufficiently critical to halt the boot process, but warning messages are displayed. A severe ErrorValue of 0x3 causes a failure in the boot process. At this stage, the system automatically restarts and uses the LastKnownGood control set, and then continues the boot process. The critical ErrorValue of 0x4 halts the process and attempts to restart, as with 0x3. A 0x4 message will then be displayed if the LastKnownGood control set also is defective.

- *Starting of services*. The Session Manager (smss.exe) launches the subsystems and services that are defined to start immediately.
- *User logon*. The Win32 subsystem starts the Winlogon.exe and then the Local Security Authority (lsass.exe). The Service Controller scans the HKEY_LOCAL_

MACHINE\SYSTEM\Set\Services subkey for all service start entries. Two additional subkeys are scanned: DependOnGroup and DependOnService. The system is ready to use when the Begin Logon box appears with instructions to press CTRL+ALT+DEL to log on. Windows Server 2003 does not consider the startup procedure complete until the first user has logged on.

NOTE During the initial boot phase, it is possible to enter an alternative boot sequence and invoke the Advanced Menu system. When F8 is pressed during the initial sequences, the **Advanced** menu is displayed, which is used strictly for repair and maintenance activities. It permits you to invoke the system in safe mode, Enable Boot Logging, Enable VGA Mode, or Last Known Good configurations. In the "Disaster Management" section of Chapter 14, we discuss how these options are applied. We also discuss other system recovery methods.

THE WINDOWS SERVER 2003 REGISTRY

The Windows Server 2003 Registry is a database that contains operating system, hardware, and software information for the local computer system. It is used by many programs, including the Windows Server kernel, device drivers, setup, and detection executables. One example of the type of information stored in the Registry is a list of all properly installed applications. Therefore, when you double-click a file name with the Windows Server Explorer, its extension is matched with a list of installed applications and launches the appropriate software. Other items stored in the Registry include:

- Hardware configuration data
- Program group and desktop settings for each user
- User profile data
- Local language and time settings
- Network configuration data
- Security information for users and groups
- ActiveX and OLE server data

NOTE The relationship between the Registry and the Active Directory is very close. The Registry maintains information for the local system, and the Active Directory provides object information about the domain network as a whole.

The Registry Structure

The Registry is based on a logical hierarchy of information, beginning with five subtrees known as keys. The concept of keys and subkeys follows the same principle as folders and subfolders within a directory tree. Every key and subkey has at least one entry that contains its name, data type, and configuration value (Table 2.3).

The keys and subkeys are stored in collections known as hives. Hive files are stored in %systemroot%\System32\Config or for user data in \%systemroot%\Profile\username. When changes are made to the Registry data, the data is compared to the logs before it is written. The log file is written first in a type of data streaming mechanism. When it is saved to disk, changes are updated to the hive key components. In this way, hive information is coupled with associated log files in an effort to minimize corruption.

If the Registry is lost or damaged, Windows Server 2003 cannot function. Therefore, its care and feeding is a critical system administrator responsibility. The first rule is to retain an emergency copy of the Registry in case of damage or loss. As stated previously, the Registry information is found in %systemroot%\System32\Config, where %systemroot% is the root directory for a system, such as \winnt. For system recovery, the Windows Server Setup program also creates a %systemroot%\Repair folder that contains the following files:

- *Autoexec.nt*—a copy of %systemroot%\System32\Autoexec.nt used to initialize the MS-DOS environment.
- *Config.nt*—a copy of %systemroot%\System32\Config.nt used to initialize the MS-DOS environment.

TABLE 2.3 Registry Keys

Key	Description
HKEY_LOCAL_MACHINE	Information about the local system such as hardware and operating system data; major keys are hardware, SAM, security, software, and system.
HKEY_CLASSES_ROOT	File allocation and OLE/ActiveX data; includes association of extensions to applications.
HKEY_CURRENT_CONFIG	System startup configuration that permits changes to device settings.
HKEY_CURRENT_USER	Profile for current active user as well as console, Control Panel, environment, printer, and software data.
HKEY_USERS	Profiles for all active users as well as default profile.

- *Default*—the Registry key HKEY_USERS\DEFAULT in compressed format.
- *Ntuser.DA*—A compressed version of %systemroot%\Profiles\DefaultUser\Ntuser.dat. The process uses Ntuser.da_ if this area needs repair.
- *Sam*—The Registry key HKEY_LOCAL_MACHINE\SAM in compressed format.
- *Security*—The Registry key HKEY_LOCAL_MACHINE\SECURITY in compressed format.
- *Setup.log*—The log of installed files with cyclic redundancy check (CRC) data. This file is read-only, system, and hidden by default.
- *Software*—the Registry key HKEY_LOCAL_MACHINE\SOFTWARE in compressed format.
- *System*—the Registry key HKEY_LOCAL_MACHINE\SYSTEM in compressed format.

During system startup, the Windows Server kernel extracts information from the Registry, such as which device drivers to load and their load order. The Ntoskrnl.exe program also passes other information, including its version number, to the registry.

The Registry Editor

System administrators may find it necessary to regularly view and edit the Registry. The Registry Editor tool (regedt32.exe) is located in the \System32 directory. Generally speaking, it is best to use the Windows Server administrative tools to resolve system issues prior to editing the Registry. However, from time to time, it is necessary to go to the source of system data. That is when the Registry Editor (Figure 2.23) comes into play.

NOTE The older style regedit is no longer supported. If regedit is invoked from the command prompt, regedit32 is automatically invoked.

CAUTION The Registry Editor is not a toy. Its improper use can result in fatal system behavior. When viewing and copying the Registry, always turn the Editor to read-only mode. When directly editing, always think twice before entering information. The Registry Editor automatically saves all changes, so once they are entered you must live with the consequences. Changes are reflected automatically.

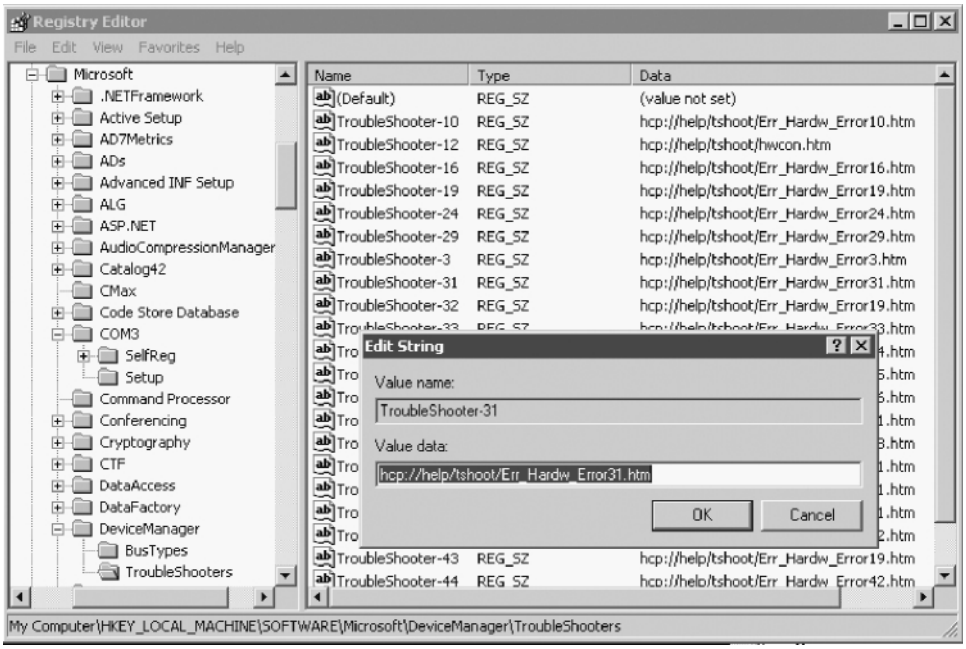


FIGURE 2.23 The Registry Editor

SOFTWARE TRACING

Tracing the activity of software applications can significantly enhance an administrator's ability to fine-tune a system's performance. The software tracing feature supports the use of macros and a preprocessor to scan code, generate message identification numbers, and create file definitions. Software tracing provides the following:

- Drivers, kernel components, services, applications, and other components can define trace events and globally unique identifiers (GUIDs).
- Components are able to use multiple-event class definition.
- Components are independently enabled and disabled.
- Components can be grouped at a higher level.
- Event information is retained in binary form.
- Trace trial activity is postprocessed for later examination.
- A failed in-memory trace is dumped by a debugger extension.

In the event of a system dump, a kernel debugger extension displays the trace elements that remain in memory. Software tracing is invoked by **Start → Control Panel → Performance and Maintenance → Administrative Tools → Performance → Performance Logs and Alerts → Trace Logs**.

VIEWING APPLICATION DEPENDENCIES

Anytime an application fails to execute properly or fails to load, a support tool in the Windows Server 2003 Resource Kit, called the Dependency Walker, can be an excellent analytical utility. It shows the minimum set of files required to run an application or load a DLL, and it shows which functions are exposed by a particular module and which ones are called by other modules. Moreover, it illustrates the full path of all modules being loaded by an application, plus their base addresses. When an application fails, the Dependency Walker provides an error message about the problematic components.

The Dependency Walker operates by recursively scanning all dependent modules required by an application. Missing files are detected first; then corrected or invalid files are identified. For example, a 16-bit version of a DLL might be present when a 32-bit version is required. Circular redundancy problems and CPU mismatches are also detected during the recursive scans.

The Dependency Walker (Figure 2.24) identifies different types of dependencies between modules, which are briefly described in Table 2.4.

To review an application's dependencies, load the Dependency Walker from the **Start** menu → choose **Programs → Windows Server 2003 Support Services → Tools**. From the **Files** menu select **Open** and browse to the desired application or DLL.

The Dependency Walker can also be used for application profiling. This technique permits it to detect dynamically loaded modules in a running application that may not be reported in the static import tables. The profiler is also used to detect that a module has failed to initialize. This activity is initiated once an application is loaded into the Dependency Walker. From the **Profile** menu, select **Start Profiling**. You will then need to select the options from the **Profile Module** dialog box. Continue the process until the desired information is obtained. To terminate the analysis, select the **Profile** menu → **Stop Profiling**.

NOTE The Windows Server 2003 Resource Kit also supplies a command-line method of launching the Dependency Walker through the depend.exe command-line utility.

Figure 2.25 displays the dialog box that permits the selection of the stable operation system for a given application.

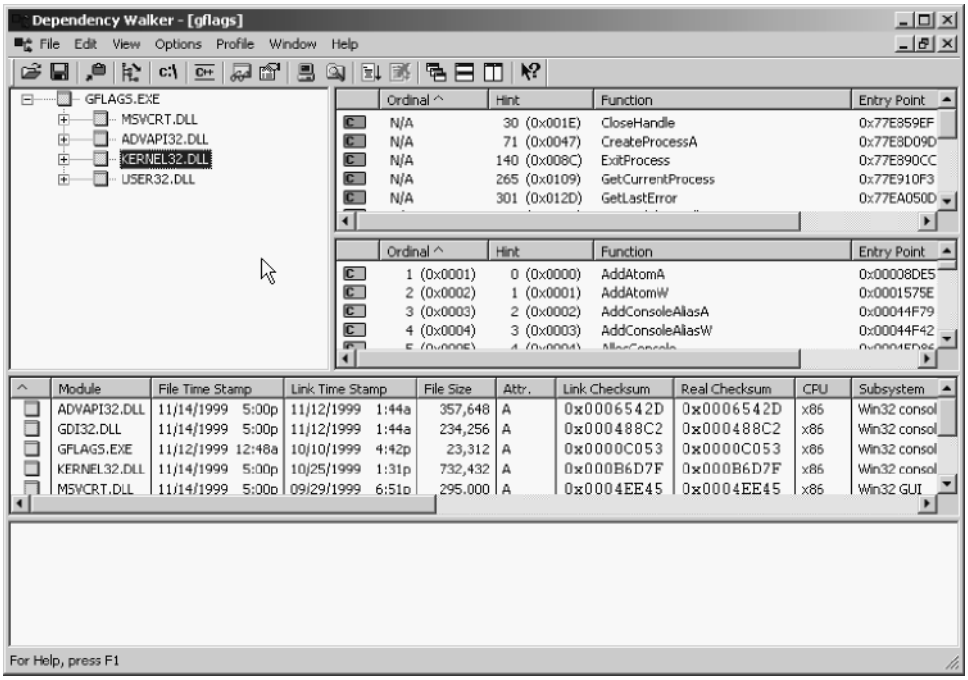


FIGURE 2.24 The Windows Server 2003 Resource Kit's Dependency Walker

TABLE 2.4 Types of Dependency between Modules

Type	Description
Implicit	A load-time dependency in which one module calls functions in another module. Even if no direct calls are made to the second module, that module is still loaded into memory.
Delay-Load	A dynamic dependency in which a module is loaded only when a specific call is made to it by the application.
Forward	A dependency in which an application calls a function in a dependent module that is actually available from another module. The call is forwarded from the second module to the third module.
Explicit	A run-time dependency, that is, a dynamic load of library functions common to OCXs, COM objects, and Visual Basic applications.
System Hook	A dependency that occurs when a specific application action is undertaken; for example, the use of a mouse will result in a new process.

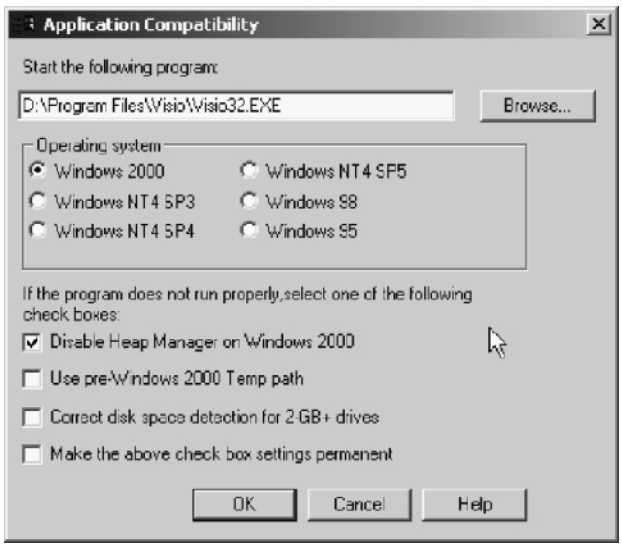


FIGURE 2.25 The Windows Server 2003 Resource Kit's Application Compatibility Tool

APPLICATION COMPATIBILITY

A major administrative headache with Windows 2000 was coping with issues of compatibility with earlier applications. Windows XP and Windows Server 2003 have made a significant leap in assuring backward compatibility. A set of new technologies and support infrastructures isolate OS differences and application bugs. This greatly increases the chances that applications that support Windows 95, Windows 98, Windows Millennium Edition, and Windows 2000 work on Windows Server 2003 and Windows XP. A key new technology called AppFixes solves such problems as the incorrect detection of the operating system by an application. It also addresses the issue of an application's referencing memory after it should have been freed. A database of applications, problems, and fixes drives which AppFixes are enabled. Microsoft maintains an updated database of these fixes as part of the Auto Update feature discussed in Chapter 3. In addition, the end user can enable an applications compatibility mode for custom-built applications.

When an application that ran on an earlier legacy version of Windows cannot be loaded during the setup function or if it later malfunctions, you must run the compatibility mode function. This is accomplished by right-clicking the application or setup program and selecting **Properties** → selecting **Compatibility** → selecting the previously supported operating system**.**

NOTE A common duty of a system administrator is to manage software versions and builds. This responsibility is extremely time consuming and often accomplished in a less than methodical manner. Software tracing provides a means to minimize the need for checked builds. With this feature, participating components—such as a driver, kernel component, service, or application—can define their own trace events and GUIDs. As such, components can have multiple event classes defined and independently enabled or disabled. Each event retains its binary form. A trace activity can be examined while it is occurring or after the event. If the system fails the in-memory trace, it can be dumped by a debugger extension or reviewed in an appropriate tool. There is also a kernel debugger extension that has the ability to display the trace elements that remain in memory at the time of a break point or a system crash, or from a crash dump. This feature is invoked via **Start → Control Panel → Performance and Maintenance → Administrative Tools → Computer Management**.

Unauthorized Applications

The Windows installer has a group policy configuration that prevents anyone except administrators from installing non-elevated applications. In versions of Windows server products prior to Windows Server 2003, this option was not enabled by default. Windows Server 2003 enables this feature by default in order to ensure greater security out of the box.

INTELLIMIRROR AND OTHER INNOVATIONS

Prior to moving on to installation of Windows Server 2003, several other innovations should be highlighted. Specifically, the manner in which IntelliMirror, Remote OS Installation, and System Management Server interplay may affect the way Windows Server is installed and administered.

IntelliMirror technology should be regarded as a collection of functions rather than a specific application. Microsoft simply calls IntelliMirror the "follow me" function. In an Active Directory environment, system settings, applications, and data will move with the user regardless of which physical computer system the user logs on to. IntelliMirror was built on the original Zero Administration Initiative in order to minimize management costs associated with user frustration in moving between computers or coping with system failures. For example, if a user accidentally removes a critical dynamic-link library (.dll), IntelliMirror can be configured to seek out that component and automatically reinstall it when the application is launched. Yet another example is a user who moves between physical locations, as

is common for consulting, sales, and technical support personnel. When such a user logs on to a computer in a different location, IntelliMirror will launch the same desktop environment and data access as would be available in the home office. In the broadest terms, IntelliMirror provides three basic functions:

1. Management of desktop settings including any customizations and restrictions
2. Software maintenance and installation including configuration, repair, and application removal known as the Software Installation and Maintenance Feature
3. User data access and management including policies that define properties and the location of user folders

Remote OS Installation is a set of technologies that supports administration of Windows Server 2003 client installation and configuration. It supports the new Preboot eXecution Environment (PXE) remote boot technology, which requests an installation of Windows XP from a delegated server. (Using Remote OS Installation is not recommended for Windows Server 2003 because of the number of configuration issues involved.) The specific application of these technologies is discussed in the next chapter. Coupled with IntelliMirror, Remote OS Installation facilitates the rollout of the Windows Server 2003 operating system and then automatically creates and maintains end-user applications and settings.

The final ingredient in the scenario is the optional System Management Server (SMS version 2.0 with Service Pack 2 or higher), which is discussed in greater detail in Chapter 17. SMS can be used to perform enterprise-wide inventory and troubleshooting. This includes management of systems other than Windows Server.

Each of these technologies offers the system administrator different advantages. The Remote OS Installation permits automatic installation and repair of Windows XP client systems. IntelliMirror provides a "just-in-time" approach to software installation and user environment management. SMS provides an advanced set of just-in-time applications for application deployment and system management. A system administrator should take a careful look at each of these technologies and apply them to more efficient management of client environments.

POSTSCRIPT

In this chapter we provided a view of Windows Server 2003 basic architecture. The chapter did not attempt to view this OS in a networked or enterprise environment. Instead, we examined the basic operating system internals and provided information about several useful system administration tools. System administrators need this high-level perspective to manage more demanding system issues. Subsequent chapters will expand on the topics reviewed here.