Windows 2000: Case Study

Windows is a modern operating system that runs on consumer and business desktop PCs and enterprise servers. Here, we will examine various aspects of Windows 2000, starting with a brief history, then moving on to its architecture. After this we will look at process, memory management, caching, I/O, the file system, and finally security.

History of Windows 2000

Microsoft's development of the Windows operating system for PC-based computers as well as servers can be divided into three eras: MS-DOS, MS-DOS based Windows and NT based Windows. Technically, each of these systems is substantially different from the others. Each of these was dominant during different decades in the history of the personal computer. Below, we will briefly sketch each of the eras shown in table:

Year	MS-DOS	MS-DOS based Windows	NT based Windows	Notes
1981	MS-DOS 1.0			Initial release for IBM PC
1983	MS-DOS 2.0			Support for PC/XT
1984	MS-DOS 3.0			Support for PC/AT
1990		Windows 3.0		Ten million copies in 2 years
1991	MS-DOS 5.0			Added memory management

1993		Windows 3.1		Runs only on 286 and later
1993			Windows NT 3.1	
1995	MS-DOS 7.0	Windows 95		MS-DOS embedded in Win 95
1996			Windows NT 4.0	
1998		Windows 98		
2000	MS-DOS 8.0	Windows Me	Windows 2000	Win Me was inferior to Win 98

Fig. Major releases in the history of Microsoft Operating Systems for desktop PCs.

1980s: MS-DOS

MS-DOS is an operating system for x86-based personal computers. It was the most commonly used member of the DOS family of operating systems, and was the main operating system for IBM PC compatible personal computers during the 1980s to the mid-1990s, until it was gradually superseded by operating systems offering a graphical user interface (GUI), in particular by various generations of the Microsoft Windows operating system.

MS-DOS based Windows

In 1990, Microsoft release Windows 3.0 for Intel 386, and sold over one million copies in six months. Windows 3.0 was not a true operating system but a graphical environment built on top of MS-DOS, which was still in control of the machine and the file system.

In 1995, Windows 95, Windows 95 was released. It contained many of the features of a full blown operating system, including virtual memory, process management and multiprogramming and introduced 32 bit programming interfaces. However, it still lacked security and provided poor isolation between applications and the operating system. The problems still continued with the subsequent releases of Windows 98 and Windows Me.

2000s: NT based Windows

NT stands for New Technology. It is because the original target processor was the new Intel 860, code named the N10). NT was designed to be portable across different processors and emphasized security and reliability, as well as compatibility with the MS-DIS based versions of Windows.

The first version of NT based windows (Windows NT 3.1) was released in 1993. It was also able to run Win32 programs, but using Win32s compatibility library. Like the first version of MS DOS based Windows, NT based Windows was not initially successful. NT required more memory, there were few 32 bit applications and incompatibilities with device drivers and applications caused many customers to stick with MS DOS based Windows which Microsoft was still improving, releasing Windows 95 in 1995. Windows 95 provided native 32 bit programming interfaces like NT, but better compatibility with existing software and applications.

Windows 2000 represented a significant evolution for Windows NT. The key technologies added were plug and play, network directory services, improved power management and an improved GUI.

Windows 2000

Windows 2000 is an operating system for use on both client and server computers. It was produced by Microsoft launched to retail in 2000. It is the successor to Windows NT 4.0, and is the last version of Microsoft Windows to display the “Windows NT” designation. It is succeeded by Windows XP (released 2001) and Windows Server 2003 (released in 2003). During development, Windows 2000 was known as Windows NT 5.0.

Four editions of Windows 2000 were released: Professional, Server, Advanced Server, and Datacenter Server. While each edition of Windows 2000 was targeted at a different market, they shared a core set of features, including many system utilities such as the Microsoft Management Console and standard system administration applications.

Support for people with disabilities was improved over Windows NT 4.0 with a number of new assistive technologies and Microsoft increased support for different languages and locale information.

All versions of the operating system support NTFS 3.0, Encrypting File System, as well as basic and dynamic disk storage. The Windows 2000 Server family has additional features,including the ability to provide Active Directory services. Distributed File System and fault-redundant storage volumes. Windows 2000 can be installed through either a manual or unattended installation. Unattended installations rely on the use of answer files to fill in installation information, and can be performed through a bootable CD using Microsoft Systems Management Server, by the System Preparation Tools.

Microsoft marketed Windows 2000 as the most secure Windows version ever at the time; however, it became the target of a number of high-profile virus attacks such as Code Red and Nimda. For ten years after its release, it continued to receive patches for security vulnerabilities nearly every month until reaching the end of its lifecycle on July 13, 2010.

Programming Windows 2000

The core of the NT operating system is the NTOS kernel mode program which provides the traditional system-call interfaces upon which the rest of the operating system is built. In Windows, only programmers at Microsoft write the system call layer. The published user mode interfaces all belong to operating system personalities that are implemented using subsystems that run on top of NTOS layers.

The Native Application Programming Interface

Windows 2000 has a set of calls it can perform. These are implemented in NTOS executive layer that runs in kernel mode. They are used internally by lower-level programs as kernel mode device drivers. Most of the native NT calls operate on kernel mode objects of one kind or another, including files, processes, threads, pipes, semaphores and so on.

The Win32 Application programming Interface

The Win32 function calls are collectively called the Win32 AO. These interfaces are publicly disclosed and documented. They are implemented as library procedures that either wrap the native NT system calls used to get the work done or, in some cases, do the work right in user mode. Though the native NT APOs, are not published, most of the functionality they provide is accessible through the Win32 API. The existing Win32 calls rarely change with the new releases of Windows, though many new functions are added to the API.

The Windows Registry

The root of the NT namespace is maintained in the kernel. Storage, such as file system volumes, is attached to the NT namespace. Since the NT namespace is constructed afresh every time the system boots, how does the system know about any specific details of the system configuration? The answer is that windows attaches a special kind of file system to the NT namespace, the file system is called the registry. The registry is divided into separate volumes called hives. When a Windows system boots, one particular hive named SYSTEM is loaded into memory by the same boot program that loads kernel and other boot files, such as boot drivers, from the boot volume.

The Windows keeps a great deal of crucial information in the SYSTEM hive, including information about drivers to use with what devices, what software run initially and many parameters governing the operation of the system.

Key	Description
HKEY_LOCAL_MACHINEHARDWARESAMSECURITYSOFTWARE SYSTEM	Properties of the hardware and softwareHardware description and mapping of hardware to driversSecurity and account information of usersSystem wide security policiesGeneric information about installed application programs Information for booting the system
HKEY_USERSUSERS-AST-IDAppEventsConsole Control Panel Environment Keyboard Layout Printers Software	Information about the userUser AST's profileWhich sound to make for specific eventsCommand prompt settingsEnvironmental variables Which keyboard: 102 key, US, AZERTY etc. Information about installed printers User preferences for Microsoft and third party software
HKEY_PERFORMANCE_DATA	Hundreds of counters monitoring system performance

HKEY_CLASSES_ROOT	Link to HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES
HKEY_CURRENT_CONFIG	Link to the current hardware profile
HKEY_CURRENT_USER	Link to the current user profile

Fig. The registry hives in Windows 2000

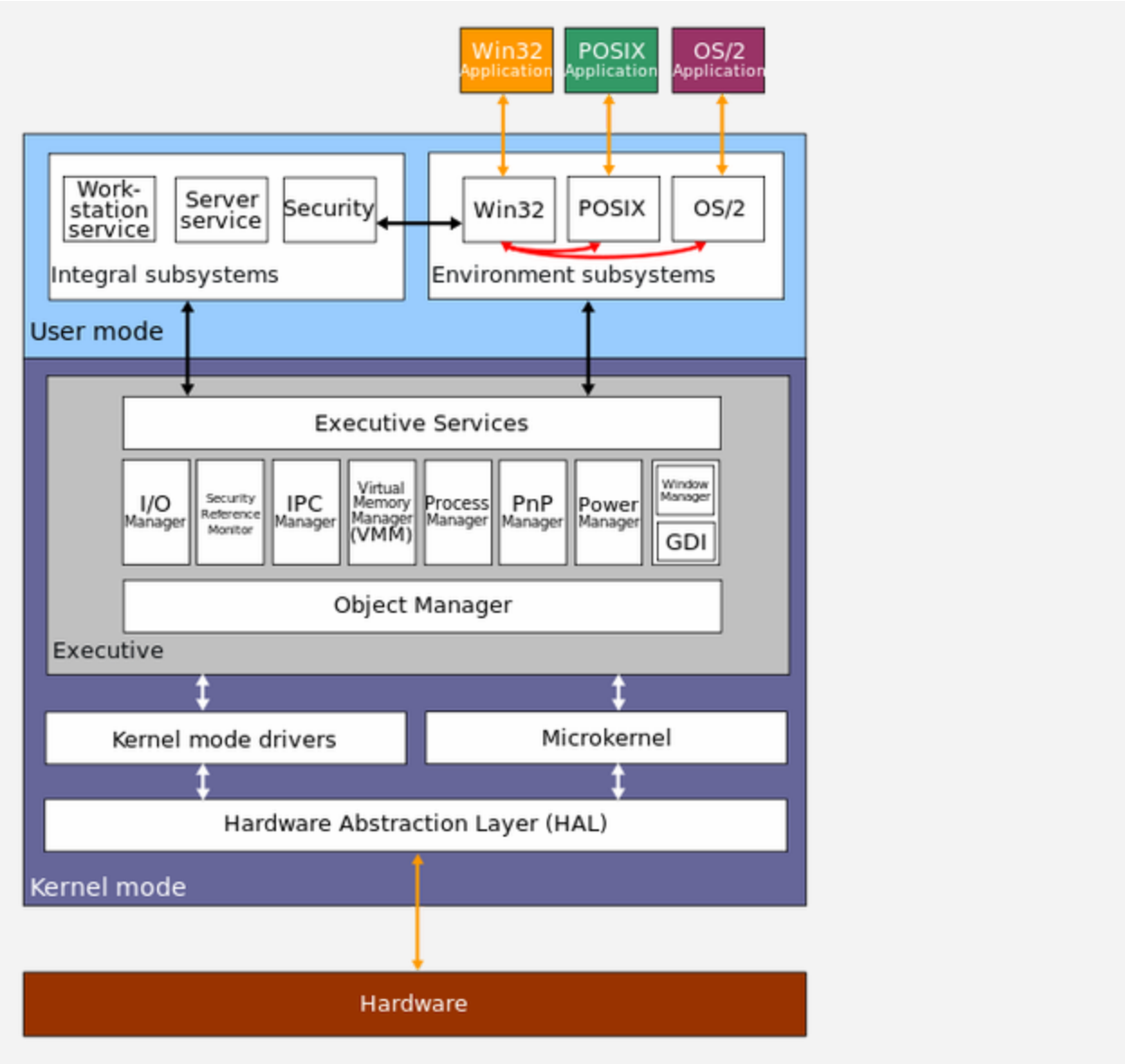
Win32 API function	Description
RegCreateKeyEx	Create a new registry key
RegDeleteKey	Delete a registry key
RegOpenKeyEx	Open a key to get a handle to it
RegEnumKeyEx	Enumerate the subkeys subordinate to the key of the handle
RegQueryValueEx	Look up the data for a vale within a key

Fig. Win32 API calls for using the registry

Architecture

The architecture of Windows 2000 is a layered design that consists of two main components, user mode and kernel mode. It is a preemptive, reentrant operating system, which has been designed to work with uniprocessor and symmetrical multi-processor (SMP)-based computers. To process input/output (I/O) requests, they use packet-driven I/O, which utilizes I/O request packets (IRPs) and asynchronous I/O. Starting with Windows

2000, Microsoft began making 64-bit versions of Windows available—before this, these operating systems only existed in 32-bit versions.



Architecture of Windows 2000

User mode in Windows NT is made of subsystems capable of passing I/O requests to the appropriate kernel mode software driversby using the I/O manager. The user mode layer of Windows NT is made up of the “Environment subsystems”, which run applications written for many different types of operating systems, and the “Integral subsystem” which operates system specific functions on behalf of environment subsystems. Kernel mode in Windows NT has full access to the hardware and system resources of the computer. The kernel mode stops user mode services and applications from accessing critical areas of the operating system that they should not have access to.

The Executive interfaces, with all the user mode subsystems, deals with I/O, object management, security and process management. The kernel sits between the Hardware Abstraction Layer and the Executive to provide multiprocessor synchronization, thread and interrupt scheduling and dispatching, and trap handling and exception dispatching. The kernel is also responsible for initializing device drivers at boot up. Kernel mode drivers exist in three levels: highest level drivers, intermediate drivers and low level drivers. Windows Driver Model (WDM) exists in the intermediate layer and was mainly designed to be binary and source compatible between Windows 98 and Windows 2000.

User mode

The user mode is made up of subsystems which can pass I/O request to the appropriate kernel mode drivers via the I/O manager (which exists in kernel mode). The user mode layer of Windows NT is made up of the Environment subsystems and the Integral subsystem.

The environment subsystems were designed to run applications written for many different types of operating systems. None of the environment subsystems can directly access hardware, and must request access to memory resources through the Virtual Memory Manager that runs in kernel mode. Also, applications run at a lower priority than kernel mode processes.

Kernel mode

Windows NT kernel mode has full access to the hardware and system resources of the computer and runs code in a protected memory area. It controls access to scheduling, thread prioritization, memory management and the interaction with hardware. The kernel mode stops user mode services and applications from accessing critical areas of the

operating system that they should not have access to; user mode processes must ask the kernel mode to perform such operations on their behalf.

While the x86 architecture supports four different privilege levels (numbered 0 to 3), only the two extreme privilege levels are used. Usermode programs are run with CPL 3, and the kernel runs with CPL 0. These two levels are often referred to as “ring 3” and “ring 0”, respectively. Such a design decision had been done to achieve code portability to RISC platforms that only support two privilege levels, though this breaks compatibility with OS/2 applications that contain I/O privilege segments that attempt to directly access hardware.

Kernel mode consists of executive services, which is itself made up of many modules that do specific tasks, kernel drivers, a kernel and a Hardware Abstraction Layer, or HAL.

Executive

The Windows Executive services make up the low-level kernel-mode portion, and are contained in the file NTOSKRNL.EXE. It deals with I/O, object management, security and process management. These are divided into several subsystems, among which are Cache Manager, Configuration Manager, I/O Manager, Local Procedure Call (LPC), Memory Manager, Object Manager, Process Structure and Security Reference Monitor (SRM). Grouped together, the components can be called Executive services (internal name Ex). System Services (internal name Nt), i.e., system calls, are implemented at this level, too, except very few that call directly into the kernel layer for better performance.

The term “service” in this context generally refers to a callable routine, or set of callable routines. This is distinct from the concept of a “service process”, which is a user mode component somewhat analogous to a daemon in Unix-like operating systems.

Object Manager

The Object Manager (internal name Ob) is an executive subsystem that all other executive subsystems, especially system calls, must pass through to gain access to Windows NT resources essentially making it a resource management infrastructure service. The object manager is used to reduce the duplication of object resource management functionality in other executive subsystems, which could potentially lead to bugs and make development of Windows NT harder. To the object manager, each resource is an object, whether that resource is a physical resource (such as a file system or peripheral) or a logical resource (such as a file). Each object has a structure or object type that the object manager must know about.

Cache Controller

Closely coordinates with the Memory Manager, I/O Manager and I/O drivers to provide a common cache for regular file I/O. Uniquely, the Windows Cache Manager operates on file blocks (rather than device blocks), for consistent operation between local and remote files.

Configuration Manager

Implements the Windows registry.

I/O Manager

Allows devices to communicate with user-mode subsystems. It accepts file system I/O requests and translates them into device specific calls, and can incorporate low-level device drivers that directly manipulate hardware to either read input or write output.

Local Procedure Call (LPC)

It provides inter-process communication ports with connection semantics.

Memory Manager

Manages virtual memory, controlling memory protection and the paging of memory in and out of physical memory to secondary storage, and implements a general-purpose allocator of physical memory.

Process Structure

Handles process and thread creation and termination, and it implements the concept of Job, a group of processes that can be terminated as a whole, or be placed under shared restrictions.

PnP Manager

Handles Plug and Play and supports device detection and installation at boot time. It also has the responsibility to stop and start devices on demand.

Power Manager

Deals with power events (power-off, stand-by, hibernate, etc.) and notifies affected drivers with special IRPs (Power IRPs).

Security Reference Monitor (SRM)

The primary authority for enforcing the security rules of the security integral subsystem.

GDI

The Graphics Device Interface is responsible for tasks such as drawing lines and curves, rendering fonts and handling palettes.

Kernel

The kernel sits between the HAL and the Executive and provides multiprocessor synchronization, thread and interrupt scheduling and dispatching, and trap handling and exception dispatching; it is also responsible for initializing device drivers at bootup that are necessary to get the operating system up and running. That is, the kernel performs almost all the tasks of a traditional microkernel; the strict distinction between Executive and Kernel is the most prominent remnant of the original microkernel design, and historical design documentation consistently refers to the kernel component as “the microkernel”.

The kernel often interfaces with the process manager. The level of abstraction is such that the kernel never calls into the process manager, only the other way around (save for a handful of corner cases, still never to the point of a functional dependence).

Kernel-mode drivers

Windows NT uses kernel-mode device drivers to enable it to interact with hardware devices. Each of the drivers has well defined system routines and internal routines that it exports to the rest of the operating system. All devices are seen by user mode code as a file object in the I/O manager, though to the I/O manager itself the devices are seen as device objects, which it defines as either file, device or driver objects. Kernel mode drivers exist in three levels: highest level drivers, intermediate drivers and low level drivers. The highest level drivers, such as file system drivers for FAT and NTFS, rely on intermediate drivers. Intermediate drivers consist of function drivers—or main driver for a device—that are optionally sandwiched between lower and higher level filter drivers. The function driver then relies on a bus driver—or a driver that services a bus controller, adapter, or bridge—which

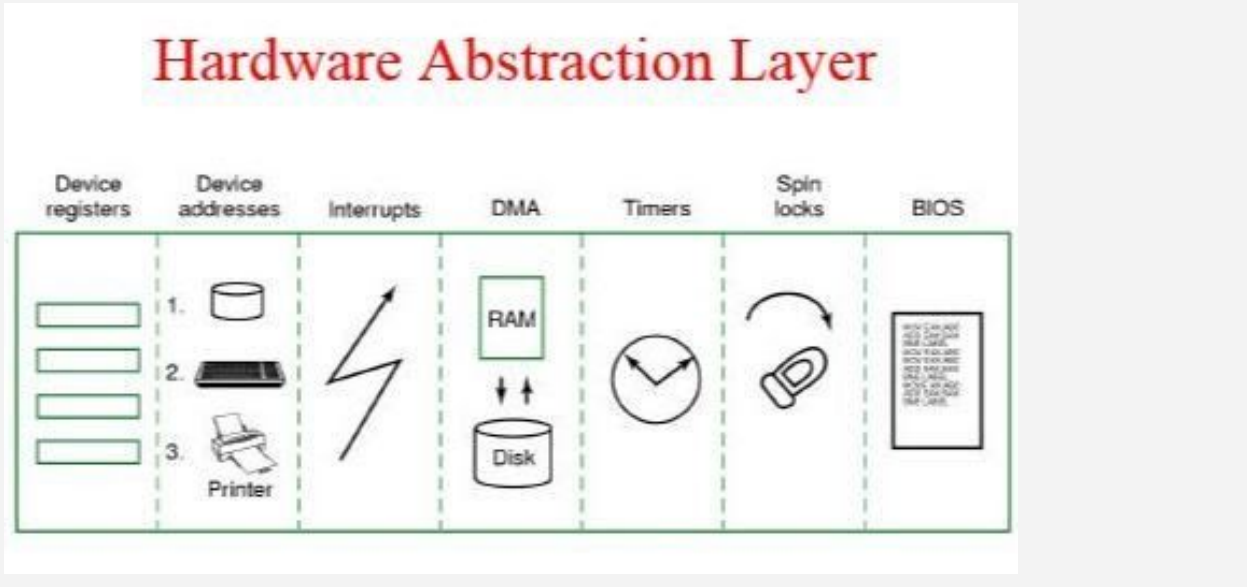
can have an optional bus filter driver that sits between itself and the function driver. Intermediate drivers rely on the lowest level drivers to function. The Windows Driver Model (WDM) exists in the intermediate layer. The lowest level drivers are either legacy Windows NT device drivers that control a device directly or can be a PnP hardware bus. These lower level drivers directly control hardware and do not rely on any other drivers.

Hardware abstraction layer

The Windows NT hardware abstraction layer, or HAL, is a layer between the physical hardware of the computer and the rest of the operating system. It was designed to hide differences in hardware and therefore provide a consistent platform on which the kernel is run. The HAL includes hardware-specific code that controls I/O interfaces, interrupt controllers and multiple processors.

However, despite its purpose and designated place within the architecture, the HAL isn't a layer that sits entirely below the kernel, the way the kernel sits below the Executive: all known HAL implementations depend in some measure on the kernel, or even the Executive. In practice, this means that kernel and HAL variants come in matching sets that are specifically engineered to work together.

In particular hardware abstraction does not involve abstracting the instruction set, which generally falls under the wider concept of portability. Abstracting the instruction set, when necessary (such as for handling the several revisions to the x86 instruction set, or emulating a missing math coprocessor), is performed by the kernel, or via platform virtualization.



Hardware Abstraction Layer

Processes and Threads

A *process* contains its own independent virtual address space with both code and data, protected from other processes. Each process, in turn, contains one or more independently executing threads. A thread running within a process can create new threads, create new independent processes, and manage communication and synchronization between the objects.

By creating and managing processes, applications can have multiple, concurrent tasks processing files, performing computations, or communicating with other networked systems. It is even possible to exploit multiple processors to speed processing.

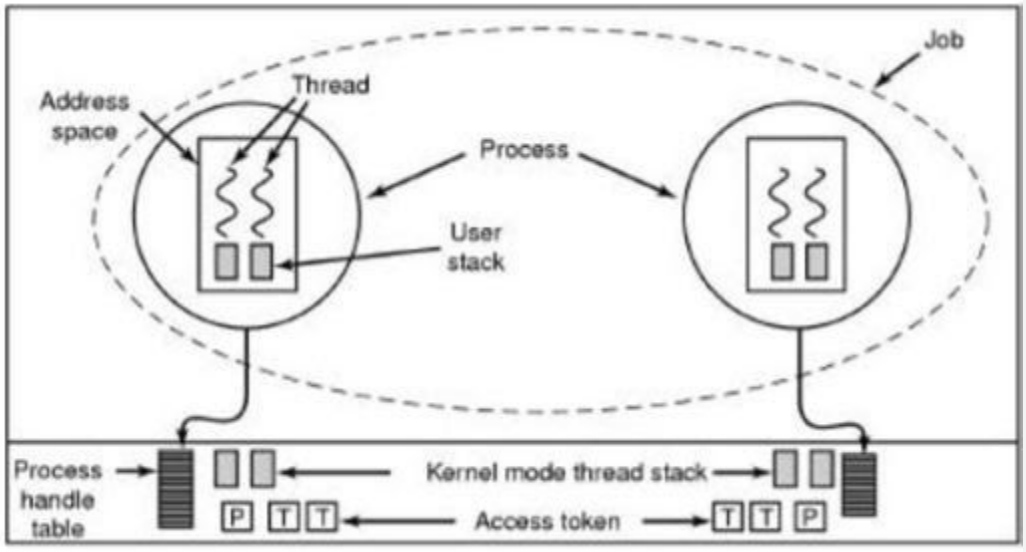
Windows process includes resources such as the following components:

- One or more threads.
- A virtual address space that is distinct from other processes' address spaces, except where memory is explicitly shared. Note that shared memory-mapped files share physical memory, but the sharing processes will use different virtual addresses to access the mapped file.
- One or more code segments, including code in DLLs.
- One or more data segments containing global variables.

- Environment strings with environment variable information, such as the current search path.
- The process heap.
- Resources such as open handles and other heaps.

Each thread in a process shares code, global variables, environment strings, and resources. Each thread is independently scheduled, and a thread has the following elements:

- A stack for procedure calls, interrupts, exception handlers, and automatic storage.
- Thread Local Storage (TLS)—arrays of pointers giving each thread the ability to allocate storage to create its own unique data environment.
- An argument on the stack, from the creating thread, which is usually unique for each thread.
- A context structure, maintained by the kernel, with machine register values.



Relationship between processes, threads, jobs and fibers

Scheduling

The Window kernel does not have any central scheduling thread. Instead when a thread cannot run anymore, the thread enters kernel mode and calls into the scheduler itself to see which thread to switch to. The following conditions cause the currently running thread to the scheduler code

1. The currently running thread blocks on a semaphore, mutex, event, I/O etc
2. The thread signals an object
3. The quantum expires

In case 1, the thread is already running in kernel mode to carry out the operation on the dispatcher or I/O object

In case 2, the running thread in kernel too. However, after signaling some object, it can definitely continue because signaling an object never blocks. Still, the thread is required to call the scheduler to see if the result of its action as released a thread with a higher scheduling priority that is now ready to run.

In case 3, an interrupt to kernel model occurs, at which point the thread executes the scheduler code to see who runs next.

File Systems

Windows 2000 supports several file systems, the most important of which are FAT-16, FAT-32, and NTFS.

The following table contains the partition and file size limitations of the various file system formats under Windows NT:

File System	Max. Partition Size	Max. File Size
FAT	4 gigabytes	4 gigabytes

NTFS	16 exabytes	16 exabyte

The FAT File System

The FAT file system for personal computers was designed at a time that floppy diskettes were the most used media, and hard drives had a capacity of ten megabytes on the average. Because of this, FAT was not designed with larger capacities in mind, and has since required the use of new operating systems and system BIOS's to allow for the use of larger hard drives and directory trees of files that number in the thousands or millions. Partitions formatted with FAT are broken into clusters. The FAT file system is also prone to fragmentation, which is the result of data being written to noncontiguous clusters, which can slow down the read/write process. FAT writes files to the first available cluster it can find, and then skips ahead past used clusters to complete writing a file. These clusters are broken down into sectors. FAT also keeps track of a few attributes for each file, such as the name of the file, the address of the starting sector, whether the file was deemed a system file, a read-only attribute, an archive bit (denoting if the file had been backed up or changed since the last time it was backup up), and a date for the file's creation or the last time the file was modified.

Because the overhead of using the FAT file system grows as partition size increases, it is advisable to not use FAT on partitions that are greater than 200 megabytes.

Whenever Windows NT formats a floppy disk, it will format using FAT because the overhead of the NTFS file system would create too much extraneous, although important, information making the actual capacity of the floppy disk too small. Therefore, NTFS is not a supported format on floppy disks.

FAT is handled slightly differently under NT. While NT's FAT implementation is one hundred percent backward compatible, NT also adds features to the FAT file system. Long filenames are allowed for FAT partitions, and are handled the same way as long filenames on a Windows 95 system. That is, the generated 8.3 filename is stored along with the long filename.

The NTFS File System

Individual file names in NTFS are limited to 255 characters; full paths are limited to 32,767 characters. File names are in Unicode, allowing people in countries not using the Latin alphabet to write file names in their native language. For example, file is a perfectly legal file name. NTFS fully supports case sensitive names (so foo is different from Foo and FOO). Unfortunately, the Win32 API does not fully support case-sensitivity for file names and not at all for directory names, so this advantage is lost to programs restricted to using Win32 An NTFS file is not just a linear sequence of bytes, as FAT-32 and UNIX files are. Instead, a file consists of multiple attributes, each of which is represented by a stream of bytes. Most files have a few short streams, such as the name of the file and its 64-bit object ID, plus one long (unnamed) stream with the data. However, a file can also have two or more (long) data streams as well. Each stream has a name consisting of the file name, a colon, and the stream name, as in foo:stream1. Each stream has its own size and is lockable independently of all the other streams. The idea of multiple streams in a file was borrowed from the Apple Macintosh, in which files have two streams, the data fork and the resource fork. This concept was incorporated into NTFS to allow an NTFS server be able to serve Macintosh clients.

Memory Management

In Windows 2000, applications and many system processes always reference memory by using virtual memory addresses. Virtual memory addresses are automatically translated to real (RAM) addresses by the hardware. Only core parts of the operating system kernel bypass this address translation and use real memory addresses directly.

Virtual memory is always being used, even when the memory that is required by all running processes does not exceed the volume of RAM that is installed on the system.

Processes and address spaces

All processes (for example, application executables) that are running under 32-bit versions of Windows are assigned virtual memory addresses (a virtual address space), ranging from 0 to 4,294,967,295 (2*32-1 = 4 GB), regardless of how much RAM is actually installed on the computer.

In the default Windows configuration, 2 gigabytes (GB) of this virtual address space are designated for the private use of each process, and the other 2 GB is shared between all

processes and the operating system. Typically, applications (for example, Notepad, Word, Excel, and Acrobat Reader) use only a fraction of the 2 GB of private address space. The operating system assigns RAM page frames only to those virtual memory pages that are being used.

Physical Address Extension (PAE) is the feature of the Intel 32-bit architecture that expands the physical memory (RAM) address to 36 bits. PAE does not change the size of the virtual address space (which remains at 4 GB), but just the volume of actual RAM that can be addressed by the processor.

Pagefile

RAM is a limited resource, whereas for most practical purposes, virtual memory is unlimited. There can be many processes, and each process has its own 2 GB of private virtual address space. When the memory being used by all the existing processes exceeds the available RAM, the operating system moves pages (4-KB pieces) of one or more virtual address spaces to the computer's hard disk. This frees that RAM frame for other uses. In Windows systems, these "paged out" pages are stored in one or more files (Pagefile.sys files) in the root of a partition. There can be one such file in each disk partition. The location and size of the page file is configured in **System Properties** (click Advanced, click Performance, and then click the **Settings** button).

Users frequently ask "how big should I make the pagefile?" There is no single answer to this question because it depends on the amount of installed RAM and on how much virtual memory that workload requires. If there is no other information available, the typical recommendation of 1.5 times the installed RAM is a good starting point. On server systems, you typically want to have sufficient RAM so that there is never a shortage and so that the pagefile is basically not used. On these systems, it may serve no useful purpose to maintain a really large pagefile. On the other hand, if disk space is plentiful, maintaining a large pagefile.

Caching

The Windows cache improves the performance of files systems by keeping recently and frequently used regions of file sin memory. The cached regions of files are called views because they represent the regions of kernel virtual addresses that are mapped onto file system files. Thus the actual management of the physical memory in the cache is provided by the memory manager. The role of cache manager is to manage the use of kernel virtual

address for views, arrange with the memory manager to pin pages in physical memory and provide interfaces for the file systems.

The cache manager facilities are shared among all the file systems. Because the cache is virtually addressed according to individual files, the cache manager is easily able to perform read-ahead on a per-file basis. Requests to access cached data comes from each file system. Virtual caching is convenient because the file systems do not have to first translate file offsets into physical block numbers before requesting cached file page. Instead the translation happens later when the memory manager calls the file system to access the page on the disk.

Windows 2000 allocates a portion of the virtual memory in your system to the *file system cache* . The file system cache is a subset of the memory system that retains recently used information for quick access. The size of the cache depends on the amount of physical memory installed and the memory required for applications. The operating system dynamically adjusts the size of the cache as needed, sharing memory optimally between process working sets and the system cache.

I/O Management

The goals of the Windows I/O manager is to provide the functionality and flexible framework for efficiently handling a very wide variety of I/O devices and services, support automatic device discovery and driver installation and power management for devices and the CPU-all using the fundamentally asynchronous structure that allows computation to overlap with I/O transfers. For a large number of common drivers, it is not necessary to install a driver, because there is already a driver that shipped with Windows.

The I/O manager is on intimate terms with plug and play manager. The basic idea behind plug and play is that of an enumerable bus. Many buses have been designed so that the plug and play manager can send request to each slot and ask the device there to identify itself. The plug and play manager allocates the hardware resources, such as interrupt levels, locate appropriate drivers and loads into the memory. And then for each device, at least one device object is allocated.

In Windows 2000, all the file systems, antivirus filters, volume managers, network protocol stacks and even kernel services that have no associated hardware are initially implemented using I/O drivers. The I/O to volumes can be filtered by a special driver to produce Volume Shadow Copies. The filter driver creates a snapshot of the volume which can be separately

mounted and represents a volume at a previous point in time. Shadow copies are also valuable for making accurate backup of server systems.

Another aspect of Windows is its support for asynchronous I/O. It is possible for a thread to start an I/O operation and then continue executing in parallel with the I/O.

Security

The security infrastructure in Windows 2000 combines the protection of strong security for networked resources with the efficiency of an advanced management infrastructure. Windows 2000 security integrates the centralized information store and policy-based control provided by Active Directory with industry standard protocols for cross-platform, secure connections between clients and servers. In addition, Windows 2000 includes integrated the public key infrastructure components to support advanced security features such as smart cards for two-factor authentication and EFS for secure on-disk data storage.

The Windows 2000 security features include:

- The ability to let users logon once and access all enterprise resources.
- Strong user authentication and authorization.
- Secure communications between internal and external resources.
- The ability to set and manage required security policies.
- Automated security auditing.

1. Interoperability with other operating systems and security protocols.
2. An extensible architecture to support application development that uses Windows 2000 security features.

These features are important elements of the overall Windows 2000 security architecture. Windows 2000 security is based on a simple model of authentication and authorization. *Authentication* identifies the user when he or she logs on and makes network connections to services. Once identified, the user is authorized access to a specific set of network resources based on permissions. *Authorization* takes place through the mechanism of access control, using entries stored in Active Directory™ as well as access control lists

(ACLs) that define permissions for objects including printers, files, and network file and print shares.

Active Directory

The Windows 2000 Active Directory service plays an essential role in network security. Active Directory provides a central place to store information about the users, hardware, applications, and data on the network so users can find what they need. It also stores the authorization and authentication information required to ensure that only appropriate users can access each network resource.

In addition, Active Directory is tightly integrated with Windows 2000 security services such as the Kerberos authentication protocol, public key infrastructure, Encrypting File System (EFS), the Security Configuration Manager, Group Policy, and delegated administration. This integration allows Windows applications to take advantage of the existing security infrastructure.

Kerberos Authentication Protocol

Windows 2000 uses the Internet standard Kerberos V5 protocol (RFC 1510) as the primary method for authenticating users. The Kerberos protocol provides a mechanism for mutual authentication between a client and a server before a network connection is opened between them. This approach is ideal for a network that includes open communications, such as those conducted over the Internet.

Public Key Infrastructure

Public key cryptography is used to secure activity on open networks, such as the Internet. It lets you encrypt data, sign it, and verify the identity of clients and servers by using certificates. The challenge with this technology is tracking the certificates. A *public key infrastructure* (PKI) provides the capability to use, manage, and find public key certificates.

Encrypting File System

For added protection of data stored locally, EFS lets you encrypt designated files or folders on a local computer, so unauthorized people can't read those files. EFS is particularly useful for protecting data on a computer that might be physically stolen, such as a laptop. You can configure EFS on laptops to ensure that all business information is encrypted in the user's document folders.

Secure Networking with IPSec

IPSec is a suite of Internet-standard protocols that allow secure, encrypted communications between two computers over an insecure network. The encryption is applied at the IP network layer, which means that it is transparent to most applications that use specific protocols for network communication. Further, IPSec provides end-to-end security, meaning that the IP packets are encrypted by the sending computer, are unreadable *en route*, and can be decrypted only by the recipient computer. For further safety, the process uses encryption algorithms to generate a single encryption key that is used at both ends of the connection, so the key does not need to be passed over the network.

Summary

Kernel mode in Windows 2000 is structure in the HAL, the kernel and executive layers of NTSOS, and a large number of device drivers implementing everything from device services to file systems and networking to graphics. The HAL hides certain differences in hardware from other components. The kernel layer manages the CPUs to support multithreading and synchronization and the executive implements the most kernel mode services.

The executive is based on kernel mode objects that represent the key executive live data structures, including processes, threads, memory sections, drivers, devices and synchronization objects.

The important objects in Windows are processes, threads and sections. Processes have virtual address spaces and are containers for resources. Threads are the unit of execution and are scheduled by the kernel layer using a priority algorithm in which highest priority ready thread always runs, preempting lower priority as necessary. Sections represent

memory objects, like files, that can be mapped into the address spaces of processes. EXE and DLL program images are represented as sections, as is shared memory.

Windows supports virtual memory. The paging algorithm is based in the working set concept. The system maintains several types of page lists, to optimize the use of memory. The cache manager manages virtual addresses in the kernel that can be used to map files into memory, dramatically improving I/O performance for many applications because read operations can be satisfied without accessing disk.

I/O is performed by device drivers, which follow the Windows Driver Model. Each driver starts out by initializing a driver object that contains the addresses of the procedures that the system can call to manipulate devices. Devices are stacked and I/O request packets are passed down the stack and services by the drivers for each device stack.

Finally, Windows 2000 has a sophisticated security system. The security infrastructure in Windows 2000 features Kerberos Authentication Protocol. Windows 2000 security integrates the centralized information store and policy-based control provided by Active Directory with industry standard protocols for cross-platform, secure connections between clients and servers. In addition, Windows 2000 includes integrated the public key infrastructure components to support advanced security features such as smart cards for two-factor authentication and EFS for secure on-disk data storage