UNIT 4 CASE STUDY: WINDOWS 2000

Structure **Page Nos.**

4.0 INTRODUCTION

In the earlier unit we have covered the case study of one of the popular operating systems namely UNIX. In this unit we will make another case study, WINDOWS 2000.

Windows 2000 is an operating system that runs on high-end desktop PCs and servers. In this case study, we will examine various aspects of Windows 2000 operating system, starting with introduction, then exploring its architecture, processes, memory management, I/O, the file system, and security aspects.

4.1 OBJECTIVES

After studying this unit you should be able to:

- gain experience relating to various aspects on Windows 2000 operating system;
- understand the architecture of Windows 2000 OS, and
- understand processes, memory management, I/Os, the file system, and above all security management handled by the Windows 2000 OS.

4.2 WINDOWS 2000 : AN INTRODUCTION

The release of NT following NT 4.0 was originally going to be called NT 5.0. But, in the year 1999, Microsoft decided to change the name to Windows 2000. This is a single main operating system built on reliable 32-bit technology but using the Windows 98 user interface. Windows 2000 is based on the Windows NT Kernel and is sometimes referred to as Windows NT 5.0. Windows 2000 is a complex operating system consisting of over 29 million lines of C /C++ code among which eight million lines of code is written for drivers. Windows 2000 is currently by far one of the largest commercial projects ever built.

4.2.1 Significant features of Windows 2000

Some of the significant features of Windows 2000 are:

- Support for FAT16, FAT32 and NTFS
- Increased uptime of the system and significantly fewer OS reboot scenarios
- Windows Installer tracks applications and recognizes and replaces missing components
- Protects memory of individual apps and processes to avoid a single app bringing the system down
- Encrypted File Systems protect sensitive data
- Secure Virtual Private Networking (VPN) supports tunneling in to private LAN over public Internet
- Personalized menus adapt to the way you work
- Multilingual version allows for User Interface and help to switch, based on logon
- Includes broader support for high-speed networking devices, including Native ATM and cable modems
- Supports Universal Serial Bus (USB) and IEEE 1394 for greater bandwidth devices.

Windows 2000 is really a NT 5.0, as it inherits many properties from NT 4.0. It is a true 32bit / 64bit multiprogramming system with protected processes. Each process consists of 32-bit (or 64 bit) demand paged virtual address space. The user process runs in user mode and operating system runs in kernel mode resulting in complete protection, thereby eliminating the Windows 98 flaws. Processes can have greater than one thread, which are scheduled by the operating system. It provides security for all files, directories, processes, and other shareable objects. And above all it supports for running on symmetric multiprocessors with upto 32 CPUs.

Windows 2000 is better than NT 4.0 with the Windows 98 user interface. It contains a number of features that are found only in Windows 98 like complete support for plug-and-play devices, the USB bus, FireWire (IEEE 1394), IrDA (Infrared link for portable computers/printers), power management etc. The new features in Windows 2000 operating system are:

- Active directory service,
- Security using Kerberos,
- Support for smart cards,
- System monitoring tools,
- Better integration of laptop computers with desktop computers,
- System management infrastructure,
- Single instance store, and job objects.

- The file system NTFS has been extended to support encrypted files, quotas, linked files, mounted volumes, and content indexing etc.
- Internationalization.

Windows 2000 consists of a single binary that runs everywhere in the world and the language can be selected at the run time. Windows 2000 uses Unicode, but a Windows 2000 does not have MS-DOS (use command line interface).

Like earlier versions of NT, Windows 2000 comes in several product levels: Professional, Server, Advanced Server, and Datacenter server. The differences between these versions are very small and same executable binary used for all versions. The various versions of Windows 2000 are show in Table 1:

Table 1: Versions of Windows 2000

Windows 2000 Version	Max RAM	CPUs	Max clients	Cluster size	Optimized for
Professional	4GB	2	10	0	Response Time
Server	4GB	4	Unlimited	0	Throughput
Advanced Server	8GB	8	Unlimited	2	Throughput
Datacenter Server	64GB	32	Unlimited	4	Throughput

When the system is installed, version is recorded in the registry (internal database). At boot time, the operating system checks the registry to see the version. The details of various versions are given in Table 1 above. The cluster size relates to capability of Windows 2000 to make two or four servers look like a single server to the outside world.

4.3 WINDOWS 2000 PROGRAMMING

In this section we will see the details of programming interface and the registry, a small in-memory database.

4.3.1 Application Programming Interface

Windows 2000 has a set of system calls, which was not made public by Microsoft. Instead Microsoft has defined a set of function calls called the Win32 API (Application Programming Interface,) shown in Figure 1, which are publicly known and well documented. These are a set of library procedures that make system calls to perform certain jobs or do the work in the user space. These Win32 API calls do not change with new releases of Windows, although new API calls are added.

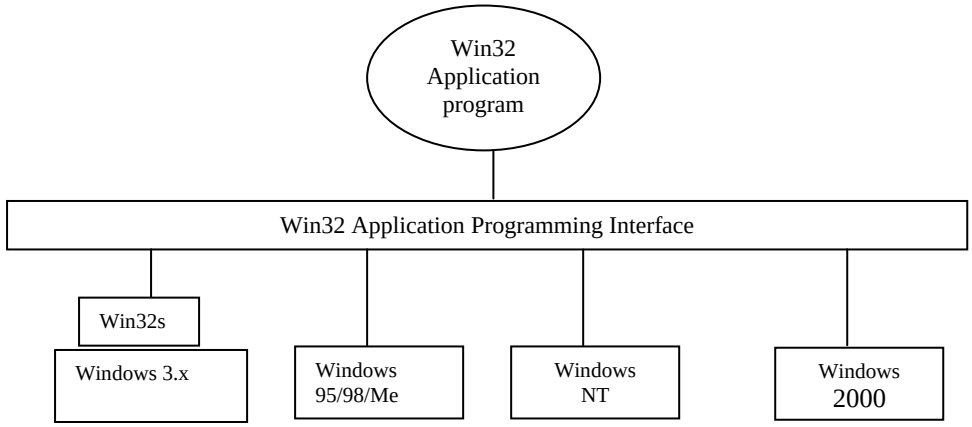

Figure 1: The Win32 Application Programming Interface

- Binary programs for the Intel x86 that conforms to Win32 API interface will run unmodified on all versions of Windows operating system. An extra library is required for Windows 3.x to match a subset of the 32-bit API calls to the 16-bit operating system. Windows 2000 has additional API calls, which will not function on older versions of Windows operating system.

- The Win32 API concept is totally different from the UNIX philosophy. In the case of UNIX, the system calls are all publicly known and form a minimal operating system and removing any of these would affect the functionally of the operating system. Whereas, Win32 provides a very comprehensive interface and the same thing can be performed in a number of ways.
- Most of Win32 API calls creates kernel objects including files, processes, threads, pipes, etc and returns a result called a handle to the caller. Not all system-created data structures are objects and not all objects are kernel objects. True kernel objects are those that need to be named, protected, or shared in some way, has system defined type, well-defined operations on it and occupies storage in kernel memory. This handle is used to perform operations on the object and does the handle refer to specific to the process that created the object. Handles cannot be passed directly to another process (in the same manner as UNIX file descriptors cannot be passed to another process and used there). But is possible to duplicate a handle and then it can be passed to another processes in a protected manner, allowing them controlled access to the objects of there process. Every object has a security descriptor, containing information regarding who may and may not perform what kinds of operations on the object. Windows 2000 can also create and use objects.
- Windows 2000 is partially object oriented because the only way to manipulate objects is by invoking operations on their handles by invoking Win32 calls. But there is not concept of inheritance and polymorphism.
- Memory management system is invisible to programmer (demand paging), but a significant feature is visible, that is the ability of a process to map a file onto a region of its virtual memory.
- File I/O: A file is a linear sequence of bytes, in Win32 environment. Win32 environment provides over 60 calls for creating and destroying files/directories, opening, closing, reading, writing, requesting/setting attributes of files, etc.
- Every process has a process ID telling who it is and every object has an access control list describing who can access it and which operation are allowed, thus providing a fine-grained security.
- Windows 2000 file names are case sensitive and use the Unicode character set.
- On Windows 2000, all the screen coordinates given in the graphic function are true 32-bit numbers.

4.3.2 The Registry

All the information needed for booting and configuring the system and tailoring it to the current user was gathered in a big central database called the registry. It consists of a collection of directories, each of which contains either subdirectories or entries (the files). A directory is called a key and all top level key (directories) starts with the string HKEY (handle to key). At the bottom of the hierarchy are the entries, called values. Each value consists of three parts: a name, a type, and the data. At the top level, the Windows 2000 registry has six keys, called root keys, as shown below. One can view it using one of the registry editors (regedit or regedt32).

- HKEY_LOCAL_MACHINE
- HKEY-USERS
- HKEY_PERFORMANCE_DATA
- HKEY_CLASS_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER

4.4 WINDOWS 2000 OS STRUCTURE

In this section we will see, how the system is organized internally, what are the various components and how these components interact with each others and with user programs.

Windows 2000 consists of two major components:

- Operating system itself, which runs in kernel mode

The kernel handles the process management, memory management, file systems, and so on. The lowest two software layers, the HAL (hardware abstraction layer) and the kernel, are developed in C and in assembly language and are partly machine dependent. The upper layers are written in C and are almost machine independent. The drivers are written in C, or in a few cases C++.

There are two main components: (a) Kernel Mode: the operating system itself, which runs in kernel mode, and (b) User Mode: the environment subsystems. The kernel handles process management, memory management, file systems, and so on. The environment subsystem is separate processes which help user program carry out certain specific system functions. Windows 2000 follows modular structure. The simplified of Windows 2000 is given below in *Figure 2*.

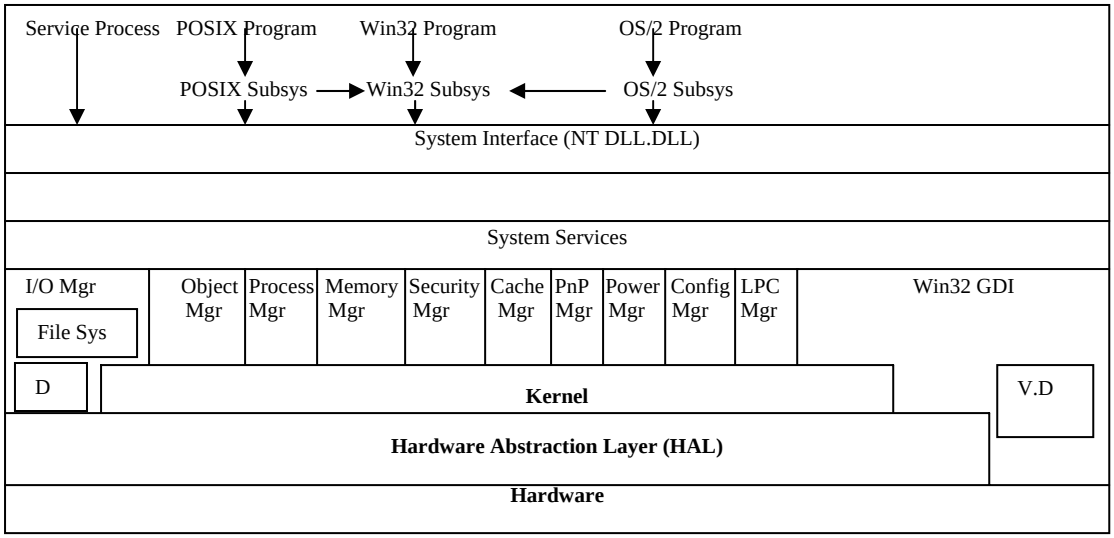

Figure 2: The structure of Windows 2000. D – Device Driver, V.D – Video Driver

12.4.1 Hardware Abstraction Layer

To make the operating system portable across platforms, Microsoft attempts to hide many of the machine dependencies in a HAL (Hardware Abstraction Layer). The job of the HAL is to present the rest of the operating system with abstract hardware devices. These devices are available in the form of machine-independent services (the procedure calls and macros). By not addressing the hardware directly, drivers and the kernel require fewer changes when being ported to a new hardware. Furthermore, HAL services are identical on all Windows 2000 systems irrespective of the underlying hardware and porting HAL is simple because all the machine dependent code is located in one place and the goals of the port are well defined. The HAL address those services which relate to the chipset on the parent board and which vary from system to system. This hides the differences between one vendor's parent board and another one's, but not the differences between a Pentium and an Alpha. The various HAL services are given below, but HAL does not provide abstraction or services for specific I/O devices such as mouse, keyboard, or disk or for memory management unit:

- Access to device registers
- Bus-independent device addressing
- Interrupt handling
- Resetting
- DMA transfer
- Control of the times and real time clock
- Low-level spin locks and multiprocessor synchronization

- Interfacing with the BIOS and its CMOS configuration memory

4.4.2 Kernel Layer

The purpose of the kernel is to make the rest of the operating system hardware-independent. It accesses the hardware via the HAL and builds upon the extremely low level HAL services to build higher-level abstractions. In addition to providing a higher-level abstraction of the hardware and handling thread switches, the kernel also has another key function: providing low level support for control objects and dispatcher objects.

Executive

The executive is the upper portion of the operating system. It is written in C and is architecture independent. The executive consists of ten components, each of which is just a collection of procedures that work together to accomplish a certain task. These components are: object manager, I/O manager, Process Manager, Memory Manager, Security Manager, Cache Manager, Plug-and-Play manager, Power Manager, Configuration Manager, Local Procedure call manager.

On booting, Windows 2000 is loaded into memory as a collection of files and the description of important files is given below in *Table 2*:

Table 2: Important files in Windows 2000 Operating System

File Name	Consists of
ntoskrnl.exe	Kernel and executive
hal.dll	HALL
Win32k.sys	Win32 and GDI
*.sys	Driver files

Device Drivers

Device drivers are not part of *ntoskrnl.exe* binary and when a drive is installed on the system, it is added to a list of the registry and is loaded dynamically on booting. A device driver can control one or more I/O devices and can perform encryption of a data stream or just providing access to kernel data structures. The largest device drivers are Win32 GDI and video, handles system calls and most of the graphics.

Objects

Objects are most important concept in Windows 2000. Objects provide a uniform and consistent interface to all system resources and data structures such as threads, processes, semaphores, etc. An object is a data structure in RAM. A file on disk is an object, but an object is created for file when it is opened. When a system boots, there are no objects present at all (except for the idle and system processes, whose objects are hardwired into the ntoskrnl.exe file). All objects are created on the fly as the system boots up and various initialization programs run. Some of the common executive object types are shown in *Table 3* below:

Table 3: Some of the Common Executive Object Types Managed by Object Manager

Object Type	Description
Process	User Process
Thread	Thread within a process
Semaphore	Counting semaphore used for interprocess synchronization
Mutex	Binary semaphore used to enter a critical region
Event	Synchronization object with persistent state (signaled/not)
Port	Mechanism for interprocess message passing
Timer	Object allowing a thread to sleep for a fixed time interval

Queue	Object used for completion notification on asynchronous I/O
Open File	Object associated with an open file
Access token	Security descriptor for some object
Profile	Data structure used for profiling CPU usage
Section	Structure used for mapping files into virtual address space
Key	Registry key
Object directory	Directory for grouping objects within the object manager
Symbolic link	Pointer to another object by name
Device	I/O device object
Device driver	Each loaded device has its own object.

As objects are created and deleted during execution, the object manager maintains a name space, in which all objects in the system are located. A process uses this name space to locate and open a handle for some other process' object, provided it has been granted permission to do so. Windows 2000 maintains three name spaces: (1) object name space, (2) file system name space, and (3) the registry name space. All three-name space follows a hierarchical model with multiple levels of directories for organizing entries. The object name space is not visible to users without special viewing tools. One freely downloadable viewing tool 'winobj' is available at www.sysinternals.com.

4.4.3 Environment Subsystem

User mode components are of three kinds:

(1) DLLs,
(2) Environment subsystems, and
(3) Service processes.

These components provide each user process with an interface that is different from the Windows 2000 system call interface. Windows 2000 supports the following APIs: Win32, POSIX, and OS2. The job of the DLLs and environment subsystems is to implement the functionality of the published interface, thus hiding the true system call interface from application programs. The Win32 interface is the official interface for Windows 2000, by using DLLs and Win32 environment subsystem, a program can be coded in Win32 specifications and run unmodified on all versions of Windows, even though the system calls are not the same on different systems. Windows 2000 uses DLLs extensively for all aspects of the system. A user process generally links with a number of DLLs that collectively implements the Win32 interface.

4.5 PROCESS AND THREADS

Each process in Windows 2000 operating system 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.

The following sections explain the basics of process management and also introduce the basic synchronization operations.

4.5.1 Windows Processes and Threads

Every process consists of one or more threads, and the Windows thread is the basic executable unit. Threads are scheduled on the basis of the following factors: (a) availability of resources such as CPUs and physical memory, (b) priority, (c) fairness, and so on. Windows has supported symmetric multiprocessing (*SMP*) since NT4, so threads can be allocated to separate processors within a system. Therefore, each 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 registers values.

Windows 2000 has a number of concepts for managing the CPU and grouping resources together. In the following section we will study these concepts.

Windows 2000 supports traditional process that communicates and synchronizes with each other. Each process contains at least one thread, which contains at least one fiber (a lightweight thread). Processes combine to form a job for certain resource management purposes. And jobs, processes, threads, and fibers provides a very general set of tools for managing parallelism and resources.

- **Job:** collection of processes that share quotas and limits
- **Process:** container for holding resources
- **Thread:** Entity scheduled by the kernel
- **Fiber:** Lightweight thread managed entirely in user space.

A job in Windows 2000 is a collection of one or more processes. There are quotas (maximum number of processes, total CPU time, memory usage) and resource limits associated with each job, store in the corresponding job object.

Every process has a 4-GB address space, with the user occupying the bottom 2 GB (3 GB on advanced server and Datacenter Server) and operating system occupying the rest. Process is created using Win32 call and consist of a process ID, one or more threads, a list of handles, and an access token containing security related information. Each process starts out with one thread and new threads can be created dynamically using a win32 call. Operating system selects a thread to run for CPU scheduling. Every thread has a state like ready, running, blocked, etc. Every thread has a thread ID, which is taken from the same space as the process IDs, so an ID can never be in use for both a process and a thread at the same time. Processes and thread can be used as byte indices into kernel tables, as they are in multiple of four. A thread is normally run in user mode, but when it invokes a system call it switches to kernel mode. There are two stacks for each thread, one for use when it is in user mode and one for use when it is in kernel mode. A thread consists of the following

- Thread ID
- Two stacks
- A context (to save its register when not running)
- A private are for its own local variables and
- Possibly its own access token. If a thread has its own access token, this variable overrides the process access token in order to let client threads pass their access rights to server threads who are doing work for them.

Threads are a scheduling concept and not a resource ownership concept. In addition to normal threads that run in user processes, Windows 2000 provides a number of daemon threads (associated with the special system or idle processes) that runs in kernel space and are not linked or associated with any user process. Switching threads in Windows 2000 are relatively expensive, as they require entering and exiting kernel mode. Further, Windows 2000 provides fibers, which are similar to threads, but are scheduled in user space by the originating program (or its run time system). A thread can have multiple fibers. There are no executive objects relating to fibers, as there are for jobs, processes, and threads. In fact, the operating system knows nothing about the fibers and there are no true system calls for managing fibers. Windows 2000 is capable of running on a symmetric multiprocessor system. The relationship between jobs, processes, and threads is illustrated in *Figure 3*.

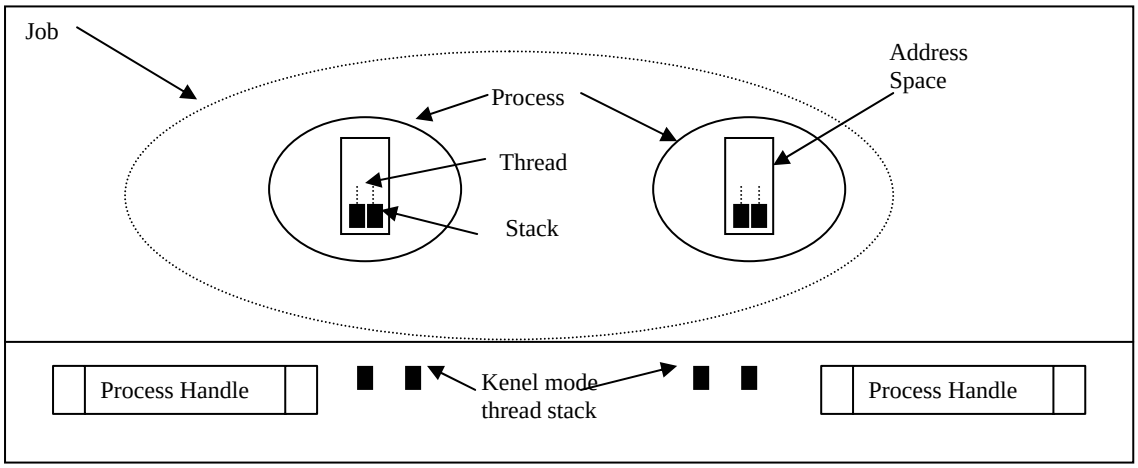

Figure 3: Relationship between jobs, processes, and threads

API Calls

Win32 API function Create Process is used to create a new process. This function consists of 10 parameters. The design is complicated that UNIX as in UNIX 'fork' has no parameters, and 'exec' has just three (pointers to the name of the file to execute, command line parameter array, and the environment strings). The ten parameters of Create Process are:

1) Pointer to the name of the executable file.
2) Command line itself
3) Pointer to a security descriptor for the process
4) Pointer to a security descriptor for the initial thread
5) A bit telling whether the new process inherits the creator's handles.
6) Miscellaneous flag (error mode, priority, debugging, consoles etc.)
7) A pointer to the environment string.
8) Pointer to the name of the new process current working directory.
9) Pointer for initial window on the screen.
10) Pointer to a structure that returns 18 values.

There is no concept of inheritance in Windows 2000 and it does not enforce any kind of parent-child or other hierarchy. All processes are created equal. But there is an implicit hierarchy in terms of who has a handle to whom. One of the eighteen parameters returns to the creating process is a handle to the new process, thus allowing control over the new process.

Initially each process is created with a one thread and process can create further threads, which is simpler than the process creation. The kernel performs the thread creation function and is not implemented purely in user space as is the case in some other operating systems. CreateThread has only six parameters:

1) Security descriptor (optional)
2) initial stack size
3) starting address
4) user defined parameter
5) initial state of the thread (ready/blocked)
6) thread ID.

4.5.2 Interprocess Communication

Threads can communicate in many ways including: pipes, named pipes, mailslots, sockets, remote procedure calls, and shared files. The various communications modes are shown in *Table 4* below:

Table 4: Thread Communication Mode

Thread communication	Mode
Pipes	byte and message, selected at creation time.
Name Pipes	Byte and message
Mailslots	A feature of Windows 2000 not present in UNIX. They are one-way, whereas pipes are two-way. They can also be used over a network but do not provide guaranteed delivery. They allow the sending process to broadcast a message to many receivers, instead of to just one receiver.
Sockets	Are like pipes, except that they normally connect processes on different machines. For example, one process writes to a socket and another one on a remote system reads from it.
Remote Procedures Calls	Are a mechanism for a process A to have process B call a procedure in B's address space on A's behalf and return the result to A.
Shared Files	Processes can share memory by mapping onto the same file at the same time

Besides Windows 2000 providing numerous interprocess communication mechanism, it also provides numerous synchronization mechanisms, including semaphores, mutexes, critical regions, and events. All of these mechanisms functions on threads, not process.

Semaphore

A semaphore is created using function 'CreateSemaphore' API function. As Semaphores are kernel objects and thus have security descriptors and handles. The handle for a semaphore can be duplicated and as a result multiple process can be synchronised on the same semaphore. Calls for up (ReleaseSemaphore) and down (WaitForSingleObject) are present. A calling thread can be released eventually using WaitForSingleObject, even if the semaphore remains at 0.

Mutexes

Mutexes are kernel objects used for synchronization and are simpler than semaphores as they do not have counters. They are locks, with API functions for locking (WaitForSingleObject) and unlocking (releaseMutex). Like Semaphores, mutex handles can be duplicated.

Critical Sections or Critical Regions

This is the third synchronization mechanism which is similar to mutexes. It is pertinent to note that Critical Section are not kernel objects, they do not have handles or security descriptors and cannot be passed between the processes. Locking and

unlocking is done using EnterCriticalSection and LeaveCriticalSection, respectively. As these API functions are performed initially in user space and only make kernel calls when blocking is needed, they are faster than mutexes.

Events

This synchronization mechanism uses kernel objects called events, which are of two types: manual-reset events and auto-reset events.

The number of Win32 API calls dealing with processes, threads, and fibres is nearly 100. Windows 2000 knows nothing about fibres and fibres are implemented in user space. As a result, the CreateFibre call does its work entirely in user space without making 12-13 system calls.

Scheduling

There is no central scheduling thread in Windows 2000 operating system. But when a thread cannot run any more, the thread moves to kernel mode and runs the scheduler itself to see which thread to switch to and the concurrency control is reached through the following conditions

- Thread blocks on a semaphore, mutex, event, I/O, etc.: In this situation the thread is already running in kernel mode to carry out the operation on the dispatcher or I/O object. It cannot possibly continue, so it must save its own context, run the scheduler to pick its successor, and load that threat's context to start it.

- It signals an object – In this situation, thread is running in kernel mode and after signaling some object it can continue as signaling an object never blocks. Thread runs the scheduler to verify if the result of its action has created a higher priority thread. If so, a thread switch occurs because Windows 2000 is fully pre-emptive.

- The running thread's quantum expires: In this case, a trap to kernel mode occurs, thread executes the scheduler to see who runs next. The same thread may be selected again and gets a new quantum and continue running, else a thread switch takes place.

The scheduler is also called under two other conditions:

1) An I/O operation completes: In this case, a thread may have been waiting on this I/O and is now released to run. A check is to be made to verify if it should pre-empt the running thread since there is no guaranteed minimum run time. The win32 API provides two hooks (SetPriorityClass, SetThreadPriority) for process to influence thread scheduling.

2) A timed wait expires.

For SetPriorityClass, the values are: realtime, high, above normal, normal, below normal, and idle. For SetThreadPriority, the values are: time critical, highest, above normal, normal, below normal, lowest, and idle. With six process classes and seven thread classes, a thread gets any one of 42 combinations, which is an input to the scheduling algorithm.

The scheduler has 32 priorities, numbered from 0 to 31. The 42 combinations are mapped onto the 32 priority classes.

4.6 BOOTING WINDOWS 2000

To start Windows 2000 system, it must be booted first. The boot process generates the initial processes that start the system. The process is as follows:

- Boot process consists of reading in the first sector of the first disk , the master boot record, and jumping to it.
- The assembly language program reads the partition table to check which partition table contains the bootable operating system.
- On finding the O.S partition, it reads the first sector of the partition, called the boot sector.
- The program in the boot sector reads its partition's root directory, searching for a file called ntldr.
- *Ntldr* is loaded into memory and executed. Ntldr loads Windows 2000.
- *Ntldr* reads *Boot.ini*, the only configuration information that is not available in the registry. It lists of various versions of hal.dll and ntoskrnl.exe available for booting in this partition, parameters no of CPUs, RAM size, space for user processes (2 GB or 3 GB), and rate of real time clock.
- *Ntldr* then selects and loads hal.dll and ntoskrnl.exe and bootvid.dll (default video driver).
- *Ntldr* next reads the registry to find out which drivers are needed to complete the boot. It reads all drivers and passes the control to ntoskrnl.exe.
- The Operating system does some general initialization and then calls the executive components to do their own initialization. The object manager prepares its name space to allow other components call it to insert objects into the name space. Memory manager set up initial page tables and the plug-and-play manager finding out which I/O devices are present and loading their drivers. The last step is creating the first true user process, the session manager, smss.exe (*The session manager is a native Windows 2000 process, it makes true system calls and does not use the Win32 environment subsystem. It starts csrss.exe and reads registry hives from disk. Its job includes entering many objects in the object manager's name space, creating required paging files, and opening important DLLs. After this, it creates winlogon.exe).* Once this process is up and running, booting of the system is complete.
- After the service processes (user space daemons) start and allow user to log in. *Winlogon.exe* first creates the authentication manager (*lsass.exe*), and then the parent process of the services (*services.exe*). *Winlogon.exe* is responsible for user logins. A separate program in *msgina.dll* handles the actual login dialog.

Table 5: The processes starting up during system booting.

Process	Details
Idle System	Home to the idle thread
Smss.exe	Creates smss.exe & paging files; reads registry, open DLLs
Csrss.exe	First real process, creates csrss & winlogon
Winlogon.exe	Win32 process
Lsass.exe	Login daemon
Services.exe	Authentication manager
	Looks in registry and starts services
Printer server	Remote job printing
File server	Local files requests
Telnet server	Remote logins
Incoming mail handler	Accepts and stores inbound email
Incoming fax handler	Accepts and prints inbound faxes
DNS Resolver	Internet domain name system server
Event logger	Logs various system events
Plug-and-play manager	Monitors hardware

4.7 MEMORY MANAGEMENT

Windows 2000 consists of an advanced virtual memory management system. It provides a number of Win32 functions for using it and part of the executives and six dedicated kernel threads for managing it. In Windows 2000, each user process has its

own virtual address space, which is 32 bit long (or 4 GB of virtual address space). The lower 2 GB minus approx 256 MB are reserved for process's code and data; the upper 2 GB map onto to kernel memory in a protected way. The virtual address space is demand paged with fixed pages size. Virtual address space for a user is shown in *Figure 4.4* below:

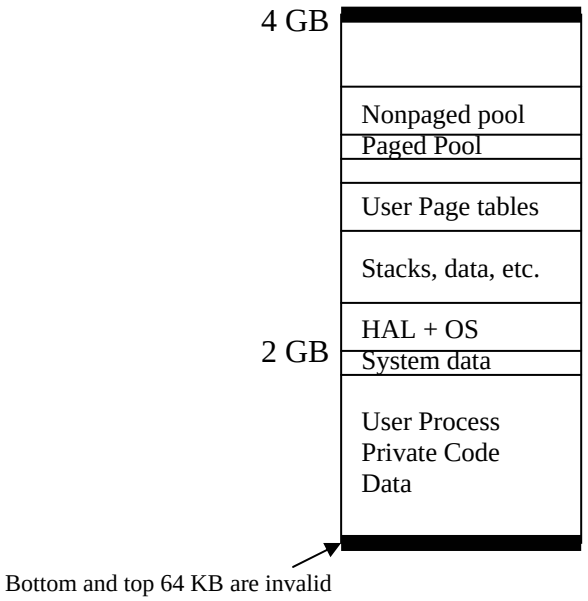

Figure 4.4: Virtual Address space layout for user processes

The bottom and top 64 Kb process virtual space is normally unmapped. This is done intentionally to catch programming errors as invalid pointers are often 0 or –1. From 64 KB onward comes the user's private code and data, which extends upto 2 GB. Some system counters and timers are available at high end of 2 GB. This makes them visible and allows processes to access them without the overhead of a system call. The next 2 GB of virtual address space contains the operating system, code, data, paged and non-paged pools. The upper 2 GB is shared by all user processes, except page tables, which are exclusively reserved for each process' own page table. The upper 2 GB of memory is not writable and mostly not even readable by user-mode processes. When a thread makes a system call, it traps into kernel mode and keeps on running in the same thread. This makes the whole OS and all of its data structure including user process visible within thread's address space by switching to thread's kernel stack. This results in less private address space per process in return for faster system calls. Large database server feel short of address space, and that is why 3-GB user space option is available on Advanced Server and Datacenter server.

There are three states available for each virtual space: free, reserved, or committed. A free page is not currently in use and reference to it results in page fault. When a process is initiated, all pages are in free state till the program and initial data are loaded onto the address space. A page is said to be committed when code or data is mapped onto a page. A committed page is referenced using virtual memory hardware and succeeds if the page is available in the main memory. A virtual page in reserved state cannot be mapped until the reservation is explicitly removed. In addition to the free, reserved, and committed states, pages also have other states like readable, writable, and executable.

In order to avoid wasting disk space, Windows 2000 committed pages that are not permanent on disk (e.g. stack pages) are not assigned a disk page till they have to be paged out. This strategy makes system more complex, but on the other hand no disk space need be allocated for pages that are never paged out.

Free and reserved pages do not consist of shadow pages on disk and a reference to these pages cause page fault. The shadow pages on the disk are arranged into one or more paging files. It is possible to have 16 paging files, spread over 16 separate disks, for higher I/O bandwidth, each created with an initial and maximum size. These files

can be created at the maximum size at the time of system installation to avoid fragmentation.

Like many versions of UNIX, Windows 2000 allows files to be mapped directly onto the virtual address spaces. A mapped file can be read or written using ordinary memory references. Memory-mapped files are implemented in the same way as other committed pages, only the shadow pages are in the user's file instead of in the paging file. Due to recent writes to the virtual memory, the version in memory may not be identical to the disk version. On unmapping or explicitly flushing, the disk version is brought up to date. Windows 2000 permits two or more processes to map onto to the same portion of a file at the same time, at different virtual addresses as depicted in *Figure 5*. The processes can communicate with each and pass data back and forth at very high bandwidth, since no copying is required. The processes may have different access permissions. The change made by one process is immediately visible to all the others, even if the disk file has not yet been updated.

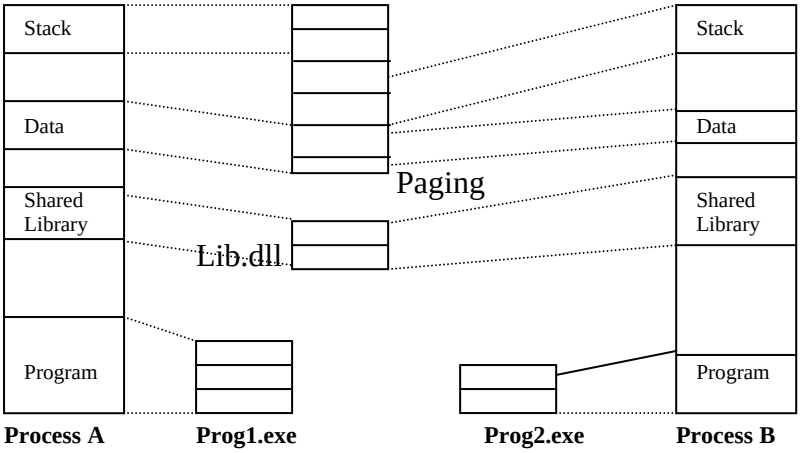

Figure 5: Two or more processes to map onto the same part of the same file at the same time, possibly at different virtual addresses in Windows 2000.

There is another technique called copy-on-write, where data can be safely written without affecting other users or the original copy on disk. This is to handle a situation where two programs share a DLL file and one of them changes the file's static data.

To allow the programs to use more physical memory than fit in the address space, there is technique to place a piece of code at any virtual address without relocation is called position independent code. There is a technique called bank switching in which a program, in which program could substitute some block of memory above the 16-bit or 20-bit limit for a block of its own memory. When 32- bit machine came, it was thought that there would be enough address space forever. Now large programs often need more than the 2 GB or 3 GB of user address space Windows 2000 allocates to them, so bank switching is back, now called address windowing extensions. This technique is only used on servers with more than 2 GB of physical memory.

System Calls

Win32 API contains a number of functions that allow a process to manage its virtual memory explicitly. Some of important functions are listed below in *Figure 6*.

Table 6: Important Win32 API functions for managing virtual memory in Windows 2000.

Win32 API function	Description
VirtualAlloc	Reserve or commit a region
VirtualFree	Release or decommit a region
VirtualProtect	Change the read/write/execute protection on a region
VirtualQuery	Inquire about the status of a region
VirtualLock	Make a region memory resident (i.e., disable paging for it.
VirtualUnlock	Make a region pageable in the usual way
CreateFileMapping	Create a file mapping object and (optionally) assign it a name
MapViewOfFile	Map (part of) a file into the address space

UnmapViewOfFile	Remove a mapped file from the address space
OpenFileMapping	Open a previously created file mapping object

These functions operate on a region having one or a sequence of pages that are consecutive in the virtual address space. To minimize porting problems to upcoming architectures with pages larger than existing ones, allocated regions always begin on 64-KB boundaries. Actual address space is in multiple of 64 KB. Windows 2000 has a API function to allow a process to access the virtual memory of a different process over which it has been given control or handle.

Windows 2000 supports a single linear 4-GB demand-paged address space per process and segmentation is not supported in any form. Pages size is shown in *Table 7* below:

Table 7: Windows 2000 Page Size

Processor	**Page Size**
Theoretically	Any power of 2 upto 64 KB
Pentium	4 KB
Itanium	8KB or 16 KB

The operating system can use 4-MB pages to reduce page table space consumed. Whereas the scheduler selects individual threads to run and does not care much about the processes, the memory manager deals with processes and does not care much about threads. As and when virtual address space is allocated for a process, the memory manager creates a VAD (Virtual Address Descriptor). VAD consists of a range of addresses mapped, the backing store file and offset where it is mapped, and the protection code. On touching the first page, the directory of page tables is created and pointer to it added in the VAD. The address space is completely defined by the list of its VAD's.

There is no concept of pre-paging in Windows 2000. All pages are brought dynamically to the memory as page fault occurs. On each page fault:

- A trap to the kernel occurs
- Kernel builds a machine independent descriptor and passes this to memory manager
- Memory manager checks its validity
- If the faulted page falls within a committed or reserved region, then it looks for address in the list of VAD, finds/create the page table, and looks up the relevant entry.

The page-table structure is different for different architectures. Entry for a Pentium system for a mapped page is given below in *Figure 6*:

Bits	**20**					**3**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Page frame	Not used	G	L	D	A	C	Wt	U	W	V				

G: Page is global to all processes
L: Large (4-MB) page
D: Page is dirty
A: Page has been accessed
C: Caching enabled/disabled
Wt: Write through (no caching)
U: Page is accessible in user mode
W: Writing to the page permitted
V: Valid page table entry

Figure 6: A page table entry for a mapped

A page fault occurs in five of the below listed categories:

1) The page referenced is not committed.

2) A protection violation occurred

3) A shared page has been written.

4) The stack needs to grow.

5) The page referenced is committed but not currently mapped in.

It is pertinent to note that Windows 2000 does not read in isolated pages from the disk, but reads in runs of consecutive pages to minimize disk transfers. The run size is larger for code pages than for data pages.

In Windows 2000 entire paging system makes heavy use of the working set concept. Every process has a working set and this set consists of the mapped-in pages that are in memory. The size and composition of the working set fluctuates as the process' thread run. Working set is described by two parameters: minimum size and the maximum size and there are hard bounds. The default minimum is in the range 20-50 and the default initial maximum is in the range 45-345 depending on the total amount of RAM.

Windows 2000 uses a local algorithm, to prevent one process from obstructing other by hogging memory. The page is added if the working set is smaller than the minimum on occurrence of a page fault. If the working set is larger than the maximum, a page is evicted from the working set to make room for the new page. System also tries to tune to the situation and algorithm may be a mix of local and global. The last 512 pages are left for some slack for new processes.

In general, Windows 2000 resolves various conflicts through complex heuristics, guesswork, historical precedent, rules of thumb, and administrator-controlled parameter setting. Memory management is a highly complex subsystem with many data structures, algorithms, and heuristics.

4.8 INPUT/OUTPUT IN WINDOWS 2000

The goal of the Windows 2000 I/O system is to provide a framework for efficiently handle various I/O devices. These includes keyboards, mice, touch pads, joysticks, scanners, still cameras, television cameras, bar code readers, microphones, monitors, printers, plotters, beamers, CD-records, sound cards, clocks, networks, telephones, and camcorders.

The I/O manager works in conjunction with the plug-and-play manager and closely connected with the plug and play manager. The power manager can put the computer into six different states: (1) fully operational, (2) Sleep-1: CPU power reduced, RAM and cache on; instant wake up, (3) Sleep-2: CPU and RAM on; CPU cache off; continue from current PC; (4) Sleep-3: CPU and cache off; RAM on; restart from fixed address; (5) Hibernate: CPU, cache, and RAM off; restart from saved disk file; (6) off: Everything off; full reboot required. Similarly I/O devices can be in various power states using power manager. It is pertinent to note that all the file systems are technically I/O drivers and request for any data block is initially sent to the cache manager and on failure it is directed to the I/O manager, which calls the proper file system driver to get the required block from the disk.

Windows 2000 supports dynamic disk, which spans multiple partitions and even multiple disks and may be reconfigured on fly without rebooting the system. Windows 2000 also provide support for asynchronous I/O and it is possible for a thread to start an I/O operation and then continue executing in parallel with the I/O. This feature is very important on servers.

4.8.1 Implementation of I/O

I/O manager creates a framework where different I/O devices can operate. This framework consists of a set of device-independent procedures for certain aspects of I/O plus a set of loaded device drivers for communicating with the devices.

4.8.2 Device Drivers

In Windows 2000 device driver must conform to a Windows Driver Model (defined for Microsoft). It also provides a toolkit for writing conformant drivers. The drivers must meet the following requirements:

- handle incoming I/O requests, which arrive in a standard format

- Be as object based as the rest of Windows 2000.
- Allow plug-and-play devices to be dynamically added / removed.
- Permit power management, where applicable.
- Be configurable in terms of resource usage.
- Be reentrant for use on multiprocessors.
- Be portable across Windows 98 and Windows 2000.

In UNIX, drivers are located by using their major device numbers. In Windows 2000, at boot time, or when a new hot pluggable plug-and-play device is attached to the computer, the system detects it and calls the plug-and-play manager. The plug-and-play manager detects the manufacturer and model number from the device. After that it looks for the corresponding driver in the hard disk or prompts the user to insert a floppy disk or CD-ROM with driver. Once detected the driver is loaded into the memory.

A driver may or may not be stacked as shown in *Figure 7* below. In stacked mode the request passes through a sequence of drivers. The use of driver stacking is: (a) separate out the bus management from the functional work of actually controlling device; (b) to be able to insert filter drivers into the stack.

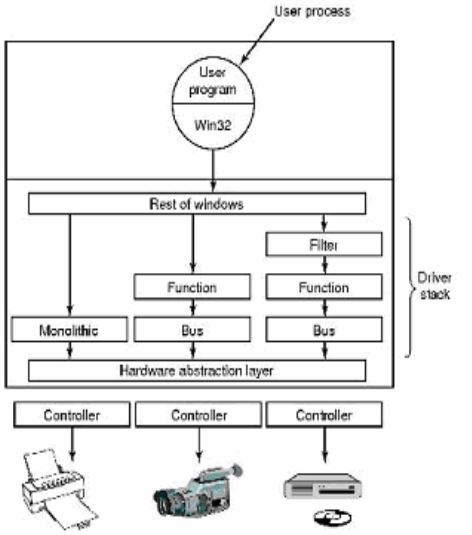

Figure 4.7: Windows 2000 allows drivers to be stacked

4.9 FILE SYSTEM MANAGEMENT

Windows 2000 supports several file systems like FAT-16, FAT-32, and NTFS. Windows 2000 also supports read-only file systems for CD-ROMs and DVDs. It is possible to have access to multiple file system types on the same running system.

4.9.1 New Technology File System (NTFS)

NTFS stands for **N**ew **T**echnology **F**ile **S**ystem. Microsoft created NTFS to compensate for the features it felt FAT was lacking. These features include increased fault tolerance and enhanced security. Individual File names in NTFS are limited to 255 characters; full paths are limited to 32,767 characters. Files are in Unicode. But Win32 API does not fully support case-sensitivity for file names. NTFS consists of multiple attributes, each of which is represented by a stream of bytes.

NTFS is a highly complex and sophisticated file system. Each NTFS volume contains files, directories, bitmaps, and other data structures. These volumes are organized as a linear sequence of blocks (512 bytes to 64 KB size), which is called clusters in Microsoft's terminology. The main data structure in each volume is master file table (MFT), which is a linear sequence of fixed size 1-KB records. Each MFT describes one file or directory.

4.9.2 Fault Tolerance

NTFS repairs hard disk errors automatically without displaying an error message. When Windows 2000 writes a file to an NTFS partition, it keeps a copy of the file in memory. It then checks the file to make sure it matches the copy stored in memory. If the copies don't match, Windows marks that section of the hard disk as bad and won't use it again (Cluster Remapping). Windows then uses the copy of the file stored in memory to rewrite the file to an alternate location on the hard disk. If the error occurred during a **read**, NTFS returns a read error to the calling program, and the data is lost.

4.9.3 Security

NTFS has many security options. You can grant various permissions to directories and to individual files. These permissions protect files and directories locally and remotely. NT operating system was designed to meet the U.S. Department of Defense's C2 (the orange book security requirements). Windows 2000 was not designed for C2 compliance, but it inherits many security properties from NT, including the following: (1) secure login with anti-spoofing mechanism, (2) Discretionary access controls, (3) Privileged access controls, (4) Address space protection per process, (5) New pages must be zeroed before being mapped in, (6) security auditing.

Every user and group in Windows 2000 is identified by a SID (security ID), which includes a binary numbers with a short header followed by a long random component. The purpose is to get a unique SID worldwide. The process and its threads run under the users' SID. And only threads with authorized SIDS can access objects. Each process has an access token which possess its SID and other information.

NTFS also includes the Encrypting File System (EFS). EFS uses public key security to encrypt files on an NTFS volume, preventing unauthorized users from accessing those files. This feature comes in quite handy on a portable compute, for example. Lose a portable, and the files on its disk are fair game to anyone who knows how to get to them. EFS use 128-bit (40-bit internationally) Data Encryption Standard (DES) encryption to encrypt individual files and folders. Encryption keys are implemented on a Windows 2000 domain or in the case of a standalone computer or locally. The operating system generates a recovery key so administrators can recover encrypted data in the event that users lose their encryption key.

4.9.4 File Compression

Another advantage to NTFS is native support for file compression. The NTFS compression offers you the chance to compress individual files and folders of your choice. An example of file compression in Windows 2000 is shown in Figure 8:

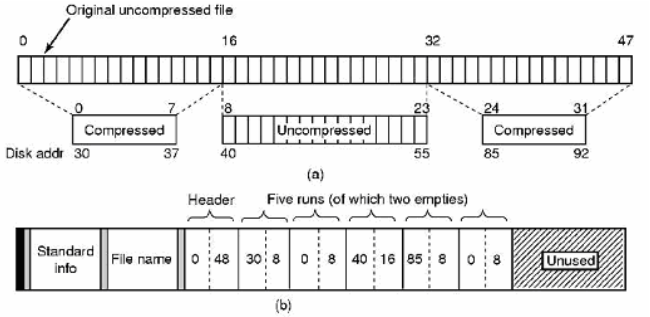

Figure 8: (a) 48-block file being compressed into 32 blocks, (b) MFT record for the file after compression

4.9.5 Disk Quotas

Disk quotas allow administrators to manage the amount of disk space allotted to individual users, charging users only for the files they own. Windows 2000 enforces quotas on a per-user and per-volume basis.

4.9.6 Caching in Windows 2000

The Windows 2000 cache manager does caching and conceptually it is similar to other systems. However, its design has some unusual properties.

Windows 2000 has single integrated cache that works for all file systems in use. Separate file systems do not need to maintain their respective cache. The operation of cache manager is shown in Figure below: operates at a higher level than the file systems.

Windows 2000 cache is organized by virtual block, not physical blocks. The traditional file caches keep track of blocks by two-part addresses of the form (partition (denotes device and partition), block (block no.)). In Windows 2000, cache manager uses (file, offset) to refers to a block.

☞ **Check Your Progress 1:**

1. Using Windows 2000 server, perform the following activities:

   a) Create a logon warning message
   b) Create an user account
   c) Create two new local groups and assign permissions
   d) Obtain a list of users without authentication

   .....

   .....

4.10 SUMMARY

Windows 2000 structure consists of the HAL, the kernel, the executive, and a thin system services layer that catches incoming system calls. There are also a variety of device drivers, file systems and GDI. The HAL layer provides hardware independence and hides differences in hardware from the upper layer. And the kernel hides the remaining differences from the executive to make it almost machine independent.

The user processes create Windows 2000 executive who is based on memory objects. User processes get back handles to manipulate them at later. Objects can be created by the executive components and object manager possess a name space where objects can be inserted for subsequent references.

Windows 2000 operating system supports processes, jobs, threads, and fibers. Processes consist of virtual address spaces, which are container for resources. Threads are the unit of execution, which are scheduled by the operating system. Lightweight threads or fibers are scheduled entirely in user space. Resources quotas are assigned by jobs, which is a collection of processes. Priority algorithm is used for scheduling.

Windows 2000 has a demand-paged virtual memory and the paging algorithm is based on the working set concept. There several lists of free pages. On occurrence of page fault, a free page is made available. Pages that have not been used for a long time are fed by trimming the working set using complex formulas.

Device drivers do I/O, which is based on the windows device model. Initializing a driver object those posses the addresses of the procedures that the system can call to add devices or perform I/O. Further activates a device driver; drivers can be stacked to act as filters.

The Windows 2000 NTFS file system is based on a master file table with one record per file or directory. Multiple attributes of a file can be in the MFT record or nonresident, on disk. NTFS also supports compression and encryption.

Security in Windows 2000 Operating system is based on access control lists. Each process has an access control token; each object has a security descriptor.

Windows 2000 maintains a single system wide cache for all the file systems. This is a virtual cache not the physical one.

4.11 SOLUTIONS / ANSWERS

Check Your Progress 1

1) (i) **Creating a Logon Warning Message**

- In the run box, type Regedit to open the Registry editor.
- In the Registry Editor navigate to: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\ CurrentVersion\WinIogo.
- Double-click the LegalNoticeCaption value.
- In the Value Data entry field, enter Unauthorized Access Warning!! Then press OK.
- Double-click the LegalNoticeText value.
- Type Unauthorized access to this system is punishable by a fine of Rs fifty thousands and/or one years in prison. Access to this system indicates that you have read and agree to this warning in the Value Data entry field, then press OK.
- Close the Registry Editor.
- Log off and back on again (no need to restart) to verify the changes. Verify that you warning message is present.

(ii) **Creating an User Account**

- Log on to your NT Server as an Administrator.
- Navigate to: User Manager for Domains.
- Select User > New User.
- Type TestUser in the Username Field.
- Type your first name in the Password Field. Keep in mind that passwords are case sensitive.
- Type the exact same password in the Confirm Password Field.
- Add this user to the Administrators Group.

(iii) **Assigning Permissions**

- Logon to your Windows NT server as Administrator.
- Navigate to User Manager for Domains.
- Create two New local groups, GROUP1 and GROUP2
- Create a user, Hari, and make this user belong to both the GROUP1 and GROUP2 groups.
- Close User Manager for Domains.
- Open My Computer.
- Navigate to an NTFS partition (Right-click to check properties if unsure)
- Create a new Folder called Check
- Right-click on Check, and select properties.
- Select the Sharing tab.
- Remove the Everyone group and give GROUP1 Full Control
- Select the Security tab
- Remove the Everyone group, and give GROUP2 Read
- Close the properties of the Check folder.

(iv) **Obtaining a List of users without authentication**

- Logon to your Windows 2000 machine as Administrator.

- Navigate to Active Users and Computers. Create a Global Group (Group types will be discussed shortly) named Global1.
- Create Domain five domain users.
- Add the new users to the Global1 group.
- On the second computer, logon to Windows NT as Administrator
- Navigate to User Manager for Domains
- Select Policies, and then Trust Relationships.
- Press the Add button, next to Trusted Domains.
- Enter the NetBIOS domain name of the other computer, you may leave the password blank, and press OK.
    - (In the event the name is not found, add an entry to the Lmhosts file of the NT computer with the hostname and IP address of the Windows 2000 computer)
    - Do not verify the trust.
- Stay in User Manager for Domains, and create a Local group called Local1.
- Double-click the new group to add members.
- Select the domain of the other computer as the source in the List Names From scroll box.
- Select the Global1 Group.
- Press the Members button to reveal the usernames of the remote domain.

4.12 FURTHER READINGS

1) Operating Systems, Milan Milenkovic, TMGH, 2004.
2) Operating Systems, William Stallings, Pearson Education, 2003.
3) www.microsoft.com/Windows**2000**/
4) www.**windows**itpro.com/**windows**nt**2000**2003faq/