Module 23: Windows NT

- History
- Design Principles
- System Components
- Environmental Subsystems
- File System
- Networking
- Programmer Interface

Windows NT

- 32-bit preemptive multitasking operating system for modern microprocessors.
- Key goals for the system:
  - portability
  - security
  - POSIX compliance
  - multiprocessor support
  - extensibility
  - international support
  - compatibility with MS-DOS and MS-Windows applications
- Uses a micro-kernel architecture.
- Available in two versions, Windows NT Workstation and Windows NT Server.
- In 1996, more NT Server licenses were sold than UNIX licenses.

History

- In 1988, Microsoft decided to develop a "new technology" (NT) portable operating system that supported both the OS/2 and POSIX APIs.
- Originally, NT was supposed to use the OS/2 API as its native environment, but during development NT was changed to use the Win32 API, reflecting the popularity of Windows 3.0.

Design Principles

- Extensibility — layered architecture.
  - NT executive, which runs in protected mode, provides the basic system services.
  - On top of the executive, several server subsystems operate in user mode.
  - Modular structure allows additional environmental subsystems to be added without affecting the executive.
- Portability — NT can be moved from one hardware architecture to another with relatively few changes.
  - Written in C and C++.
  - Processor-dependent code is isolated in a dynamic link library (DLL) called the “hardware abstraction layer” (HAL).

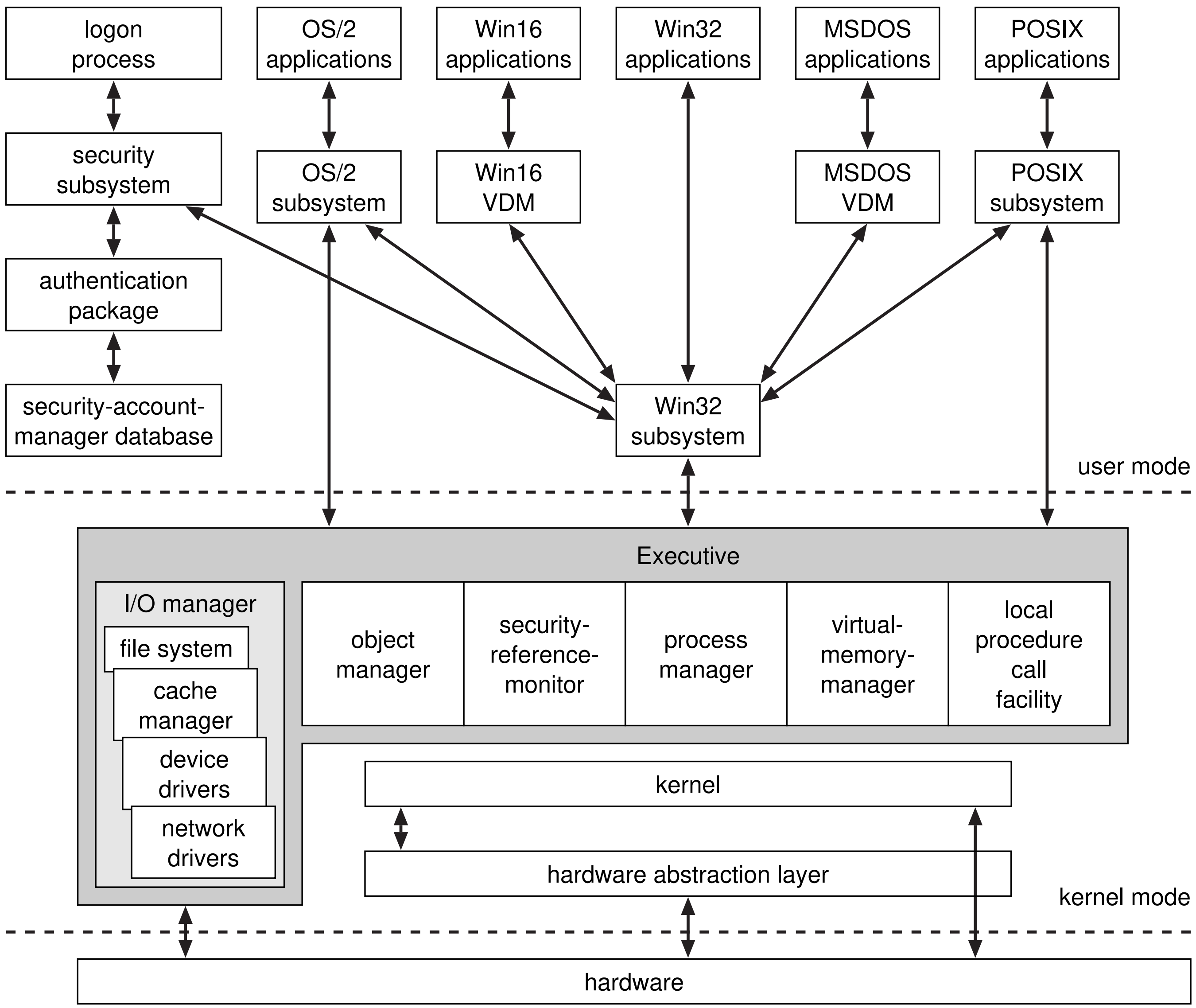
Design Principles (Cont.)

- Reliability — NT uses hardware protection for virtual memory, and software protection mechanisms for operating system resources.
- Compatibility — applications that follow the IEEE 1003.1 (POSIX) standard can be compiled to run on NT without changing the source code.
- Performance — NT subsystems can communicate with one another via high-performance message passing.
  - Preemption of low priority threads enables the system to respond quickly to external events.
  - Designed for symmetrical multiprocessing.
- International support — supports different locales via the national language support (NLS) API.

NT Architecture

- Layered system of modules.
- Protected mode — HAL, kernel, executive.
- User mode — collection of subsystems
  - Environmental subsystems emulate different operating systems.
  - Protection subsystems provide security functions.

Depiction of NT Architecture



System Components — Kernel

- Foundation for the executive and the subsystems.
- Never paged out of memory; execution is never preempted.
- Four main responsibilities:
  - thread scheduling
  - interrupt and exception handling
  - low-level processor synchronization
  - recovery after a power failure
- Kernel is object-oriented; uses two sets of objects.
  - *dispatcher objects* control dispatching and synchronization (events, mutants, mutexes, semaphores, threads, and timers).
  - *control objects* (asynchronous procedure calls, interrupts, power notify, power status, process and profile objects).

Kernel — Process and Threads

- The process has a virtual memory address space, information (such as a base priority), and an affinity for one or more processors.
- Threads are the unit of execution scheduled by the kernel's dispatcher.
- Each thread has its own state, including a priority, processor affinity, and accounting information.
- A thread can be in one of six states: ready, standby, running, waiting, transition, and terminated.

Kernel — Scheduling

- The dispatcher uses a 32-level priority scheme to determine the order of thread execution. Priorities are divided into two classes.
  - The real-time class contains threads with priorities ranging from 16 to 31.
  - The variable class contains threads having priorities from 0 to 15.
- Characteristics of NT's priority strategy:
  - Tends to give very good response times to interactive threads that are using the mouse and windows.
  - Enables I/O-bound threads to keep the I/O devices busy.
  - Compute-bound threads soak up the spare CPU cycles in the background.

Kernel — Scheduling (Cont.)

- Scheduling can occur when a thread enters the ready or wait state, when a thread terminates, or when an application changes a thread's priority or processor affinity.
- Real-time threads are given preferential access to the CPU; but NT does not guarantee that a real-time thread will start to execute within any particular time limit.

Kernel — Trap Handling

- The kernel provides trap handling when exceptions and interrupts are generated by hardware or software.
- Exceptions that cannot be handled by the trap handler are handled by the kernel's *exception dispatcher*.
- The interrupt dispatcher in the kernel handles interrupts by calling either an interrupt service routine (such as in a device driver) or an internal kernel routine.
- The kernel uses spin locks that reside in global memory to achieve multiprocessor mutual exclusion.

Executive — Object Manager

- NT uses objects for all its services and entities; the object manager supervises the use of all the objects.
  - Generates an object *handle*.
  - Checks security.
  - Keeps track of which processes are using each object.
- Objects are manipulated by a standard set of methods, namely `create`, `open`, `close`, `delete`, `query name`, `parse` and `security`.

Executive — Naming Objects

- The NT executive allows any object to be given a name, which may be either permanent or temporary.
- Object names are structured like file path names in MS-DOS and UNIX.
- NT implements a *symbolic link object*, which is similar to *symbolic links* in UNIX that allow multiple nicknames or aliases to refer to the same file.
- A process gets an object handle by creating an object, by opening an existing one, by receiving a duplicated handle from another process, or by inheriting a handle from a parent process.
- Each object is protected by an access control list.

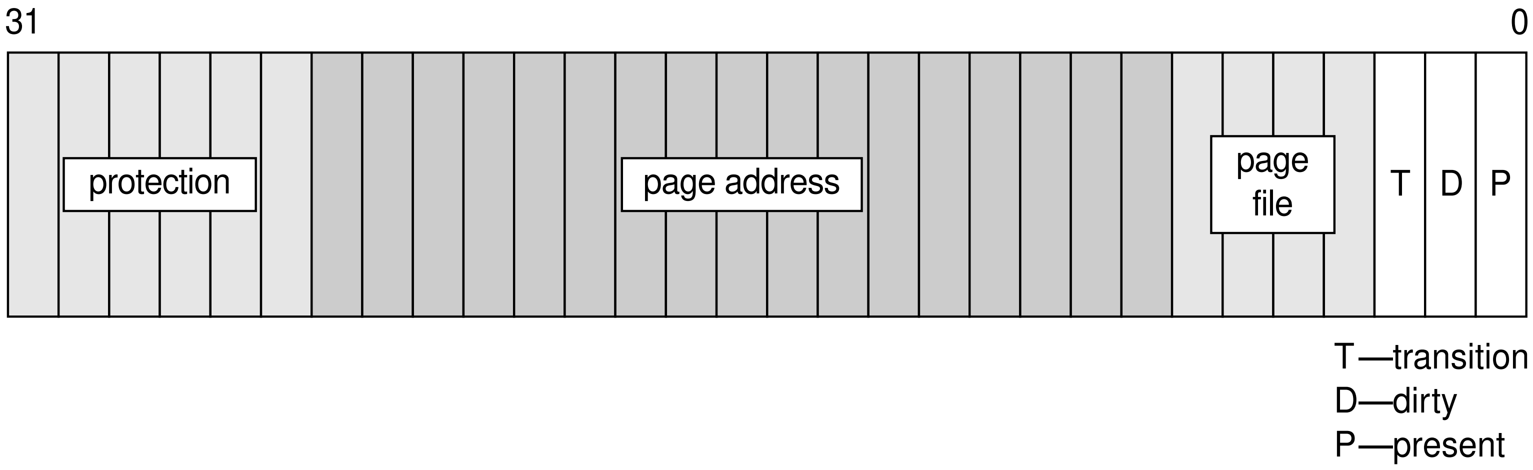
Executive — Virtual Memory Manager

- The design of the VM manager assumes that the underlying hardware supports virtual to physical mapping, a paging mechanism, transparent cache coherence on multiprocessor systems, and virtual address aliasing.
- The VM manager in NT uses a page-based management scheme with a page size of 4 KB.
- The NT VM manager uses a two step process to allocate memory.
  - The first step *reserves* a portion of the process's address space.
  - The second step *commits* the allocation by assigning space in the NT paging file.

Virtual Memory Manager (Cont.)

- The virtual address translation in NT uses several data structures.
  - Each process has a *page directory* that contains 1024 *page directory entries* of size 4 bytes.
  - Each page directory entry points to a *page table* which contains 1024 *page table entries* (PTEs) of size 4 bytes.
  - Each PTE points to a 4 KB *page frame* in physical memory.
- A 10-bit integer can represent all the values from 0 to 1023, therefore, can select any entry in the page directory, or in a page table.
- This property is used when translating a virtual address pointer to a byte address in physical memory.
- A page can be in one of six states: valid, zeroed, free, standby, modified and bad.

The PTE Structure



- 5 bits for page protection, 20 bits for page frame address, 4 bits to select a paging file, and 3 bits that describe the page state.

Executive — Process Manager

- Provides services for creating, deleting, and using threads and processes.
- Issues such as parent/child relationships or process hierarchies are left to the particular environmental subsystem that owns the process.

Executive — Local Procedure Call Facility

- The LPC passes requests and results between client and server processes within a single machine.
- In particular, it is used to request services from the various NT subsystems.
- When an LPC channel is created, one of three types of message passing techniques must be specified.
  - First type is suitable for small messages, up to 256 bytes; port's message queue is used as intermediate storage, and the messages are copied from one process to the other.
  - Second type avoids copying large messages by pointing to a shared memory section object created for the channel.
  - Third method, called *quick* LPC, is used by graphical display portions of the Win32 subsystem.

Executive — I/O Manager

- The I/O Manager is responsible for
  - file systems
  - cache management
  - device drivers
  - network drivers
- Keeps track of which installable file systems are loaded, and manages buffers for I/O requests.
- Works with VM Manager to provide memory-mapped file I/O.
- Controls the NT cache manager, which handles caching for the entire I/O system.
- Supports both synchronous and asynchronous operations, provides time outs for drivers, and has mechanisms for one driver to call another.

Executive — Security Reference Manager

- The object-oriented nature of NT enables the use of a uniform mechanism to perform runtime access validation and audit checks for every entity in the system.
- Whenever a process opens a handle to an object, the security reference monitor checks the process's security token and the object's access control list to see whether the process has the necessary rights.

Environmental Subsystems

- User-mode processes layered over the native NT executive services to enable NT to run programs developed for other operating systems.
- NT uses the Win32 subsystem as the main operating environment; Win32 is used to start all processes. It also provides all the keyboard, mouse and graphical display capabilities.
- MS-DOS environment is provided by a Win32 application called the *virtual dos machine* (VDM), a user-mode process that is paged and dispatched like any other NT thread.

Environmental Subsystems (Cont.)

- 16-Bit Windows Environment:
  - Provided by a VDM that incorporates *Windows on Windows*.
  - Provides the Windows 3.1 kernel routines and stub routines for window manager and GDI functions.
- The POSIX subsystem is designed to run POSIX applications following the POSIX.1 standard which is based on the UNIX model.

File System

- The fundamental structure of the NT file system (NTFS) is a *volume*.
  - Created by the NT disk administrator utility.
  - Based on a logical disk partition.
  - May occupy a portion of a disk, an entire disk, or span across several disks.
- All *metadata*, such as information about the volume, is stored in a regular file.
- NTFS uses *clusters* as the underlying unit of disk allocation.
  - A cluster is a number of disk sectors that is a power of two.
  - Because the cluster size is smaller than for the 16-bit FAT file system, the amount of internal fragmentation is reduced.

File System — Internal Layout

- NTFS uses *logical cluster numbers* (LCNs) as disk addresses.
- A file in NTFS is not a simple byte stream, as in MS-DOS or UNIX, rather, it is a structured object consisting of *attributes*.
- Every file in NTFS is described by one or more records in an array stored in a special file called the Master File Table (MFT).
- Each file on an NTFS volume has a unique ID called a *file reference*.
  - 64-bit quantity that consists of a 16-bit file number and a 48-bit sequence number.
  - Can be used to perform internal consistency checks.
- The NTFS name space is organized by a hierarchy of directories; the *index root* contains the top level of the B+ tree.

File System — Recovery

- All file system data structure updates are performed inside transactions.
  - Before a data structure is altered, the transaction writes a log record that contains redo and undo information.
  - After the data structure has been changed, a commit record is written to the log to signify that the transaction succeeded.
  - After a crash, the file system data structures can be restored to a consistent state by processing the log records.

File System — Recovery (Cont.)

- This scheme does not guarantee that all the user file data can be recovered after a crash, just that the file system data structures (the metadata files) are undamaged and reflect some consistent state prior to the crash.
- The log is stored in the third metadata file at the beginning of the volume.
- The logging functionality is provided by the NT *log file service*.

File System — Security

- Security of an NTFS volume is derived from the NT object model.
- Each file object has a security descriptor attribute stored in its MFT record.
- This attribute contains the access token of the owner of the file, and an access control list that states the access privileges that are granted to each user that has access to the file.

Volume Management and Fault Tolerance

- `FtDisk`, the fault tolerant disk driver for NT, provides several ways to combine multiple SCSI disk drives into one logical volume.
- Logically concatenate multiple disks to form a large logical volume, a *volume set*.
- Interleave multiple physical partitions in round-robin fashion to form a *stripe set* (also called RAID level 0, or "disk striping").
  - Variation: *stripe set with parity*, or RAID level 5.
- Disk mirroring, or RAID level 1, is a robust scheme that uses a *mirror set* — two equally sized partitions on two disks with identical data contents.
- To deal with disk sectors that go bad, `FtDisk` uses a hardware technique called *sector sparing*, and NTFS uses a software technique called *cluster remapping*.

File System — Compression

- To compress a file, NTFS divides the file's data into *compression units*, which are blocks of 16 contiguous clusters.
- For sparse files, NTFS uses another technique to save space.
  - Clusters that contain all zeros are not actually allocated or stored on disk.
  - Instead, gaps are left in the sequence of virtual cluster numbers stored in the MFT entry for the file.
  - When reading a file, if a gap in the virtual cluster numbers is found, NTFS just zero-fills that portion of the caller's buffer.

Networking

- NT supports both peer-to-peer and client/server networking; it also has facilities for network management.
- To describe networking in NT, we refer to two of the internal networking interfaces:
  - NDIS (Network Device Interface Specification) — Separates network adapters from the transport protocols so that either can be changed without affecting the other.
  - TDI (Transport Driver Interface) — Enables any session layer component to use any available transport mechanism.
- NT implements transport protocols as drivers that can be loaded and unloaded from the system dynamically.

Networking — Protocols

- The *server message block* (SMB) protocol is used to send I/O requests over the network. It has four message types:
  - `Session control`
  - `File`
  - `Printer`
  - `Message`
- The Network Basic Input/Output System (NetBIOS) is a hardware abstraction interface for networks. Used to:
  - Establish logical names on the network.
  - Establish logical connections or *sessions* between two logical names on the network.
  - Support reliable data transfer for a session via NetBIOS requests or *SMB*s.

Networking — Protocols (Cont.)

- NetBEUI (NetBIOS Extended User Interface): default protocol for Windows 95 peer networking and Windows for Workgroups; used when NT wants to share resources with these networks.
- NT uses the TCP/IP Internet protocol to connect to a wide variety of operating systems and hardware platforms.
- PPTP (Point-to-Point Tunneling Protocol) is used to communicate between Remote Access Server modules running on NT machines that are connected over the Internet.
- The NT NWLink protocol connects the NetBIOS to Novell NetWare networks.

Networking — Protocols (Cont.)

- The Data Link Control protocol (DLC) is used to access IBM mainframes and HP printers that are directly connected to the network.
- NT systems can communicate with Macintosh computers via the AppleTalk protocol if an NT Server on the network is running the Windows NT Services for Macintosh package.

Networking — Dist. Processing Mechanisms

- NT supports distributed applications via named NetBIOS, named pipes and mailslots, Windows Sockets, Remote Procedure Calls (RPC), and Network Dynamic Data Exchange (NetDDE).
- NetBIOS applications can communicate over the network using NetBEUI, NWLink, or TCP/IP.
- Named pipes are a connection-oriented messaging mechanism that are named via the *uniform naming convention* (UNC).
- Mailslots are a connectionless messaging mechanism that are used for broadcast applications, such as for finding components on the network.
- Winsock, the windows sockets API, is a session-layer interface that provides a standardized interface to many transport protocols that may have different addressing schemes.

Distributed Processing Mechanisms (Cont.)

- The NT RPC mechanism follows the widely-used Distributed Computing Environment standard for RPC messages, so programs written to use NT RPCs are very portable.
  - RPC messages are sent using NetBIOS, or Winsock on TCP/IP networks, or named pipes on Lan Manager networks.
  - NT provides the *Microsoft Interface Definition Language* to describe the remote procedure names, arguments, and results.

Networking — Redirectors and Servers

- In NT, an application can use the NT I/O API to access files from a remote computer as if they were local, provided that the remote computer is running an MS-NET server.
- A *redirector* is the client-side object that forwards I/O requests to remote files, where they are satisfied by a *server*.
- For performance and security, the redirectors and servers run in kernel mode.

Access to a Remote File

- The application calls the I/O manager to request that a file be opened (we assume that the file name is in the standard UNC format).
- The I/O manager builds an I/O request packet.
- The I/O manager recognizes that the access is for a remote file, and calls a driver called a Multiple Universal Naming Convention Provider (MUP).
- The MUP sends the I/O request packet asynchronously to all registered redirectors.
- A redirector that can satisfy the request responds to the MUP.
  - To avoid asking all the redirectors the same question in the future, the MUP uses a cache to remember which redirector can handle this file.

Access to a Remote File (Cont.)

- The redirector sends the network request to the remote system.
- The remote system network drivers receive the request and pass it to the server driver.
- The server driver hands the request to the proper local file system driver.
- The proper device driver is called to access the data.
- The results are returned to the server driver, which sends the data back to the requesting redirector.

Networking — Domains

- NT uses the concept of a *domain* to manage global access rights within groups.
- A domain is a group of machines running NT Server that share a common security policy and user database.
- NT provides four domain models to manage multiple domains within a single organization.
  - *Single domain model*, domains are isolated.
  - *Master domain model*, one of the domains is designated the master domain.
  - *Multiple master domain model*, there is more than one master domain, and they all trust each other.
  - *Multiple trust model*, there is no master domain. All domains manage their own users, but they also all trust each other.

Name Resolution in TCP/IP Networks

- On an IP network, *name resolution* is the process of converting a computer name to an IP address.

  e.g., `www.bell-labs.com` resolves to `135.104.1.14`

- NT provides several methods of name resolution:
  - Windows Internet Name Service (WINS)
  - broadcast name resolution
  - domain name system (DNS)
  - a hosts file
  - an LMHOSTS file

Name Resolution (Cont.)

- WINS consists of two or more WINS servers that maintain a dynamic database of name to IP address bindings, and client software to query the servers.
- WINS uses the Dynamic Host Configuration Protocol (DHCP), which automatically updates address configurations in the WINS database, without user or administrator intervention.

Programmer Interface — Access to Kernel Obj.

- A process gains access to a kernel object named `XXX` by calling the `CreateXXX` function to open a *handle* to `XXX`; the handle is unique to that process.
- A handle can be closed by calling the `CloseHandle` function; the system may delete the object if the count of processes using the object drops to 0.
- NT provides three ways to share objects between processes.
  - A child process inherits a handle to the object.
  - One process gives the object a name when it is created, and the second process opens that name.
  - `DuplicateHandle` function:
    - Given a handle to a process and the handle's value, a second process can get a handle to the same object, and thus share it.

Programmer Interface — Process Management

- A process is started via the `CreateProcess` routine which loads any dynamic link libraries that are used by the process, and creates a *primary thread*.
- Additional threads can be created by the `CreateThread` function.
- Every dynamic link library or executable file that is loaded into the address space of a process is identified by an *instance handle*.

Process Management (Cont.)

- Scheduling in Win32 utilizes four priority classes:
  - `IDLE_PRIORITY_CLASS` (priority level 4)
  - `NORMAL_PRIORITY_CLASS` (level 8) – typical for most processes
  - `HIGH_PRIORITY_CLASS` (level 13)
  - `REALTIME_PRIORITY_CLASS` (level 24)
- To provide performance levels needed for interactive programs, NT has a special scheduling rule for processes in the `NORMAL_PRIORITY_CLASS`.
  - NT distinguishes between the *foreground process* that is currently selected on the screen, and the *background processes* that are not currently selected.
  - When a process moves into the foreground, NT increases the scheduling quantum by some factor, typically 3.

Process Management (Cont.)

- The kernel dynamically adjusts the priority of a thread depending on whether it is I/O-bound or CPU-bound.
- To synchronize the concurrent access to shared objects by threads, the kernel provides synchronization objects, such as semaphores and mutexes.
  - In addition, threads can synchronize by using the `WaitForSingleObject` or `WaitForMultipleObjects` functions.
  - Another method of synchronization in the Win32 API is the critical section.

Process Management (Cont.)

- A *fiber* is user-mode code that gets scheduled according to a user-defined scheduling algorithm.
  - Only one fiber at a time is permitted to execute, even on multiprocessor hardware.
  - NT includes fibers to facilitate the porting of legacy UNIX applications that are written for a fiber execution model.

Programmer Interface — Interprocess Comm.

- Win32 applications can have interprocess communication by sharing kernel objects.
- An alternate means of interprocess communication is message passing, which is particularly popular for Windows GUI applications.
  - One thread sends a message to another thread or to a window.
  - A thread can also send data with the message.
- Every Win32 thread has its own input queue from which the thread receives messages.
- This is more reliable than the shared input queue of 16-bit windows, because with separate queues, one stuck application cannot block input to the other applications.

Programmer Interface — Memory Management

- Virtual memory:
  - `VirtualAlloc` reserves or commits virtual memory.
  - `VirtualFree` decommits or releases the memory.
  - These functions enable the application to determine the virtual address at which the memory is allocated.
- An application can use memory by memory mapping a file into its address space.
  - Multistage process.
  - Two processes share memory by mapping the same file into their virtual memory.

Memory Management (Cont.)

- A heap in the Win32 environment is a region of reserved address space.
  - A Win32 process is created with a 1 MB *default heap*.
  - Access is synchronized to protect the heap's space allocation data structures from damage by concurrent updates by multiple threads.
- Because functions that rely on global or static data typically fail to work properly in a multithreaded environment, the thread-local storage mechanism allocates global storage on a per-thread basis.
  - The mechanism provides both dynamic and static methods of creating thread-local storage.