Chapter 21 – Case Study: Windows XP

Outline

Chapter 21 – Case Study: Windows XP

Outline (continued)

Chapter 21 – Case Study: Windows XP

Outline (continued)

Objectives

- After reading this chapter, you should understand:
  - the history of DOS and Windows operating systems.
  - the Windows XP architecture.
  - the various Windows XP subsystems.
  - asynchronous and deferred procedure calls.
  - how user processes, the executive and the kernel interact.
  - how Windows XP performs process, thread, memory and file management.
  - the Windows XP I/O subsystem.
  - how Windows XP performs interprocess communication.
  - networking and multiprocessing in Windows XP.
  - the Windows XP security model.

21.1 Introduction

- Introducing Windows XP
  - Most recent desktop OS from Microsoft
  - May 2003: Over 1/3 of all Internet users
- 5 Editions, a 6th one soon
  - *Windows XP Home Edition*
  - *Windows XP Professional*
  - *Windows XP Tablet PC Edition*
  - *Windows XP Media Center Edition*
  - *Windows XP 64-Bit Edition*
  - *Windows XP 64-Bit Edition for 64-Bit Extended Systems* (soon)

21.2 History

- 1976 Bill Gates and Paul Allen found Microsoft
  - Bill Gates drops out of Harvard
  - Paul Allen quits day job at Honeywell
  - Move to Albuquerque, NM
- 1981 MS-DOS 1.0
  - 16-bit addressing
  - Real mode
- 1985 Windows 1.0
  - First Microsoft GUI operating system
  - Introduced protected mode

21.2 History

- 1990 Windows 3.1 and Windows for Workgroups 3.1
  - Eliminated real mode, introduced enhanced mode
  - Added network support (LANs)
- 1992 Windows NT 3.1
  - New Technology operating system
  - Created new corporate line
  - Focused on security and stability
  - NTFS
  - Eliminated direct access to memory
  - 32-bit addressing

21.2 History

- 1996 Windows NT 4.0
  - Moved graphics driver into kernel
- 2000 Windows 2000
  - Last purely corporate line desktop OS
  - Active Directory
    - Database of users, computers and services
  - Kerberos
    - Enables single sign-on

21.2 History

- 1995 Windows 95
  - Consumer line
  - 32-bit addressing
  - DirectX
    - Simulates direct access to hardware through API
- 1998 Windows 98
  - Bundled Internet Explorer into operating system
- 2000 Windows ME
  - Last purely consumer line desktop operating system
  - Does not boot in DOS mode

21.2 History

- Windows XP
  - Released 2001
  - Merged consumer and corporate codebases
  - 64-bit support

21.3 Design Goals

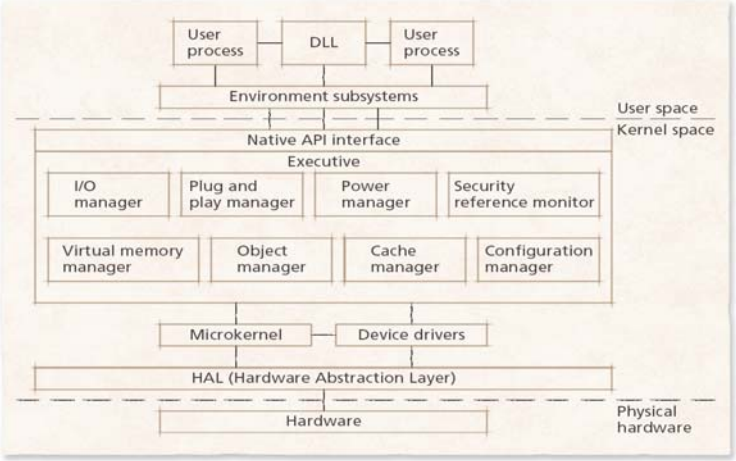
- Merge corporate and consumer codebases
  - Windows XP Professional and Windows XP Home Edition
  - Built on same codebase
- Security
  - Kerberos
  - Access control lists
  - Internet Connection Firewall
- Scalability
  - Symmetric multiprocessing
  - Windows XP 64-Bit Edition
  - Windows XP Embedded

21.3 Design Goals

- User-friendly
  - New GUI
  - More multimedia and network support
- Fast boot
  - 30 seconds after cold boot
  - 20 seconds from hibernation
  - 5 seconds from standby

21.4 System Architecture

Figure 21.1 Windows XP system architecture.



21.4 System Architecture

- Operating system design
  - Modified microkernel
  - Layered
- Components
  - HAL
    - Interacts with hardware, drives device components on mainboard
    - Abstracts hardware specifics that differ between systems of the same architecture
  - Microkernel
    - Basic system mechanisms
    - Thread scheduling, interrupt dispatching, etc.

21.4 System Architecture

- Abstracts hardware specifics that differ between architectures
- Device drivers
  - Control peripheral devices
- Executive
  - Administers the main operating system subsystems, such as the file system, I/O subsystem and system memory
  - Native API
- Environment subsystems
  - Provide a specific computing environment for user-mode processes.
  - Examples: Win32, SFU, WOW64

21.4 System Architecture

- DLLs
  - User-mode modules that processes can dynamically link
  - Environment subsystems' APIs are DLLs
- System services
  - Like Linux daemons: processes that execute in the background at all times.
  - Examples: Task Manager, Computer Browser, etc.

21.5 System Management Mechanisms

- Environment in which Windows XP's components execute.
  - How data is stored and retrieved (registry)
  - Objects
  - Interrupt priority
  - Software interrupts (APCs, DPCs)
  - System threads

21.5.1 Registry

- Central database that stores configuration data accessible to all processes
  - User data
  - System data
  - Hardware data
  - Application data
- Logical structure
  - Tree whose nodes are keys
    - Subkeys and values
    - Predefined keys (e.g., HKEY_LOCAL_MACHINE)
  - Using the registry
    - Navigate the tree structure by going from keys to subkeys
    - Accessing relevant values

21.5.1 Registry

- Windows XP administration of the registry
  - Configuration manager
  - Data stored in hives

21.5.2 Object Manager

- Objects in Windows XP
  - Represents a physical (e.g., peripheral device) or logical resource (e.g., process)
  - Managed by the object manager
  - Represented by a data structure in memory
  - An instance of object type.
    - Defines the object's attributes
    - Defines the object's standard functions (e.g., open and close)
  - Examples: processes, threads, pipes, files, devices, etc.

21.5.2 Object Manager

- Handles and Pointers
  - Pointers
    - Only used by kernel-mode threads
  - Handles
    - Used by user-mode processes and kernel components
    - Like a pointer but allows the system control over what a thread can do to the object.
    - Can be duplicated and passed to other processes

21.5.2 Object Manager

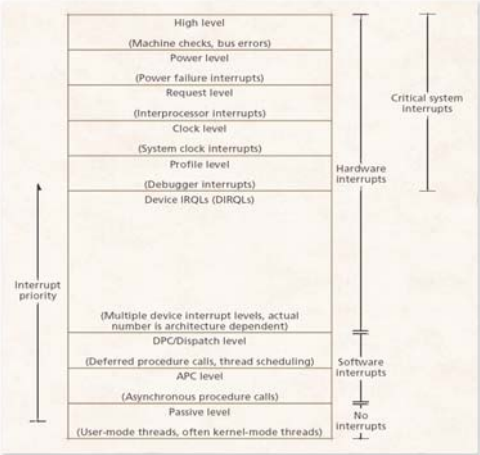
- Object naming
  - Object can be named or unnamed
  - Named objects categorized in object manager's namespace
- Deleting objects
  - No more handles: deleted from namespace (only kernel-mode threads can open a handle to an unnamed objects)
  - No more handles or pointers: deleted from memory

21.5.3 Interrupt Request Levels (IRQLs)

- Interrupt Request Levels (IRQLs)
  - Measure of interrupt priority
  - Processor always executes at an IRQL
  - Processor masks interrupts with IRQL <= current IRQL
- Levels (see figure next slide)
  - Passive: no interrupt being processed
  - APC: asynchronous procedure calls
  - DPC/dispatch: deferred procedure calls, thread scheduling
  - Device IRQL (DIRQL): device interrupts
  - Critical system interrupts: Profile (debugger), Clock, Request (interprocessor), Power, High

21.5.3 Interrupt Request Levels (IRQLs)

Figure 21.2 Interrupt request levels (IRQLs)



21.5.4 Asynchronous Procedure Calls (APCs)

- Asynchronous procedure calls (APCs)
  - Procedure calls, queued to a specific thread
  - Uses
    - Signal I/O completion
    - Communicate between executive and user-mode threads
- Types of APCs
  - Kernel-mode APCs
    - Executed when the target thread obtains a processor
    - Special vs. normal
  - User-mode APCs
    - Executed when the target thread enters an alertable *wait* state
    - After draining queue, thread reenters *wait* state

21.5.5 Deferred Procedure Calls (DPCs)

- Deferred procedure calls (DPCs)
  - Software interrupts queued, executed in the context of the currently running thread
  - Used for general system processing, such as interrupt processing
- Restrictions
  - Scheduling code cannot execute until DPC completes
  - Cannot perform any action that might force scheduling code to execute
    - Cannot access pageable data
    - Cannot block
- Executed when the system returns to the DPC IRQL (unless the DPC is a low-priority DPC)

21.5.6 System Threads

- Used by kernel-mode components that need to do work not in response to a user request
  - Cache manager flushing dirty cache pages
  - Device driver that cannot accomplish all interrupt processing at an elevated IRQL (e.g., because it must access pageable data)
- Kernel threads
  - Created by kernel-mode components
  - Typically belong to the System process
  - In general, behave the same as user-mode threads

21.5.6 System Threads

- System worker threads
  - Provided by the microkernel
  - Sleep until a kernel-mode component queues a work item
  - 3 types: delayed, critical, hypercritical

21.6 Process and Thread Management

- Process
  - Execution context
  - Program code
  - Resources
  - Associated threads
- Threads: units of execution
- Process and threads are objects

21.6.1 Process and Thread Organization

- Data structures that describe processes
  - EPROCESS block (used mainly by executive):
    - Process ID, access token, handle table, etc.
    - Contains a KPROCESS block
    - Points to PEB
    - Stored as a linked list
  - KPROCESS block (used mainly by microkernel):
    - scheduling information
    - synchronization information
  - Process environment block (PEB):
    - Process information available to user threads
    - DLLs linked to process, heap information, etc.

21.6.1 Process and Thread Organization

- Data structures that describe threads
  - ETHREAD block (used mainly by executive)
    - Process ID, pending I/O requests, thread's start address, etc.
    - Contains a KTHREAD block
    - Points to the thread's process's EPROCESS block
  - KTHREAD block (used mainly by the microkernel)
    - Scheduling priority, thread state, etc.
    - Points to thread's TEB
  - Thread environment block
    - Thread information available to user threads
    - Critical sections owned by thread, stack information, etc.

21.6.1 Process and Thread Organization

- Thread local storage (TLS)
  - Area where threads can store local data (e.g., for a DLL)
  - TLS index contains one TLS slot for each of a process's threads
  - Process can have many TLS indexes

21.6.1 Process and Thread Organization

- Creating and terminating processes
  - Creating processes
    - API function call
    - Parent and child independent, except that child can inherit handles and attributes from parent
  - Terminating processes
    - All of process's threads terminate
    - A process's thread directs the process terminate
    - User that owns the process logs off

21.6.1 Process and Thread Organization

- Jobs
  - Group processes together as a unit
  - Manage resources consumed by these processes (e.g., CPU time, memory consumption, etc.)
  - Terminate all processes at once
- Fibers
  - Unit of execution (like a thread)
  - Scheduled by thread that creates them, not microkernel.
  - Thread must convert itself into a fiber to create fibers
  - Fiber local storage (FLS)

21.6.1 Process and Thread Organization

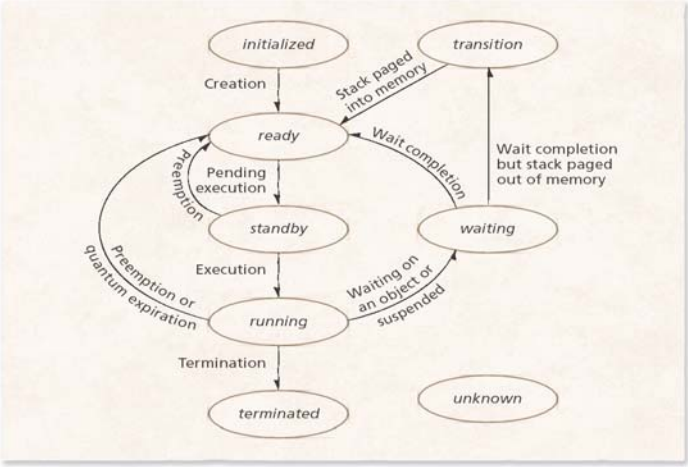
- Thread pools
  - Worker threads that sleep waiting for work items
  - Each process gets a thread pool
  - Useful in certain situations
    - Fulfilling client requests
    - Asynchronous I/O
    - Combining several threads that sleep most of the time
  - Memory overhead and less control for the programmer

21.6.2 Thread Scheduling

- Thread States
  - *Initialized*
  - *Ready*
  - *Standby*
  - *Running*
  - *Waiting*
  - *Transition*
  - *Terminated*
  - *Unknown*
- Transitions (see figure next slide)

21.6.2 Thread Scheduling

Figure 21.3 Thread state-transition diagram.



21.6.2 Thread Scheduling

- Threads scheduled without regard to processes
- 32 Priority levels
  - 0 = lowest priority
  - 31 = highest priority
- 32 ready queues
- Round-robin for highest-priority non-empty ready queue

21.6.2 Thread Scheduling

- Preemption
  - Can occur when:
    - Thread's priority changes
    - Thread enters the *ready* state
    - Thread exiting the *running* state
  - Preempted thread back to front of ready queue
  - Real-time threads: get new quantum
  - Other threads: quantum not reset

21.6.2 Thread Scheduling

- Determining priority
  - Dynamic threads: priority levels 0-15
  - Real-time threads: priority levels 16-31
  - Process
    - Base priority class
    - Determines a narrow range for the base priorities of its threads
  - Thread
    - Base priority level
    - Determines the exact base priority within a priority class
    - Base priority class + Base priority level = base priority

21.6.2 Thread Scheduling

- Determining priority (cont.)
  - Priority changes
    - Only for dynamic threads
    - Priority boosts
      - Exiting the wait state
      - Window receives user input
      - Has not executed for a long time
    - Priority reduction
      - Cannot go below base priority
      - Reduced by one if thread executes for entire quantum
      - Priority returns to previous level after one quantum if thread boosted because had not executed for a long time

21.6.2 Thread Scheduling

- Multiprocessor scheduling
  - Affinity mask
  - Last processor = processor on which thread executed last time it ran
  - Ideal processor
    - Maximize parallelism – related threads, different ideal processors
    - Maximize cache hits – related threads, same ideal processor
  - Dispatcher chooses from highest priority non-empty run queue based on time waited, last processor, ideal processor

21.6.3 Thread Synchronization

- Dispatcher objects
  - Event object
    - Signaled when event occurs;
    - unsignaled either when one thread awakens or all threads awaken (choice determined by event's creator)
  - Mutex object
    - One owner
    - Acquire – unsignaled; release – signaled
  - Semaphore object
    - Counting semaphore
    - Signaled while count > 0; unsignaled when count 0
    - Can be acquired multiple times by same thread

21.6.3 Thread Synchronization

Figure 21.4 Dispatcher objects in Windows XP.

Dispatcher Object	Transitions from Unsignaled to Signaled State When
Event	Associated event occurs.
Mutex	Owner of the mutex releases the mutex.
Semaphore	Semaphore's count rises above zero.
Waitable timer	Specified amount of time elapses.

21.6.3 Thread Synchronization

- Dispatcher objects (cont.)
  - Waitable timer object
    - Signaled when time elapses
    - Manual reset vs. auto reset
    - Single user vs. periodic
  - Objects that can act as dispatcher objects: process, thread, console input

21.6.3 Thread Synchronization

- Kernel mode locks
  - Spin lock
  - Queued spin lock
    - More efficient than spin lock
    - Guarantees FIFO ordering of requests

21.6.3 Thread Synchronization

- Kernel mode locks (cont.)
  - Fast mutex
    - Like a mutex, but more efficient
    - Cannot specify maximum wait time
    - Reacquisition by owning thread causes deadlock
    - APC IRQL
    - Variant at passive IRQL
  - Executive resource lock
    - One lock holder in exclusive mode
    - Many lock holders in shared mode
    - Good for readers and writers

21.6.3 Thread Synchronization

- Other synchronization tools
  - Critical section object
    - Like a mutex, but only for threads of the same process
    - Faster than a mutex
    - No maximum wait time
  - Timer-queue timer
    - Waitable timer objects combined with a thread pool
  - Interlocked variable access
    - Atomic operations on variables
  - Interlocked singly-linked lists
    - Atomic insertion and deletion
  - APCs and IPC

21.7 Memory Management

- Virtual memory manager (VMM)
  - Executive component responsible for managing memory
- Lazy allocation
  - Avoid allocating memory until necessary
- Prefetching
  - Move pages from disk to main memory before they are needed
- Pagefile
  - Stores pages that do not fit in main memory
  - Windows XP supports up to 16 pagefiles

21.7.1 Memory Organization

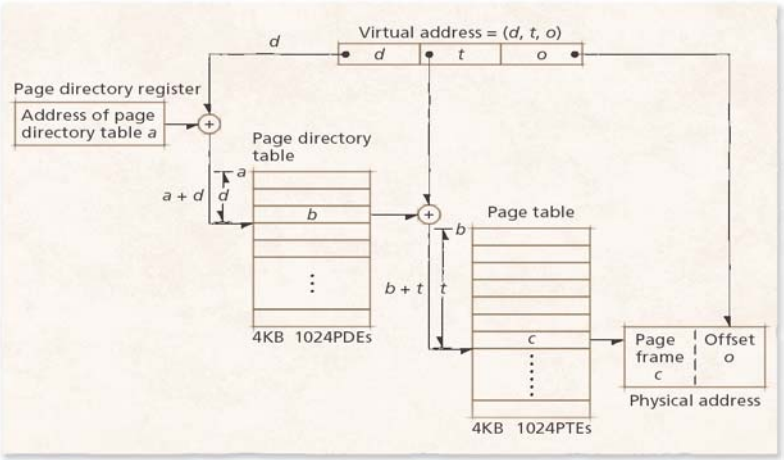
- 32-bit virtual address space
  - Windows 64-Bit Edition has 64-bit address space.
  - 4GB virtual address space per process
- User space vs. System space
  - Process can access only user space
  - VMM stores page tables and other data in system space
  - 2GB user space, 2GB system space
- 4KB pages

21.7.1 Memory Organization

- 2-level hierarchical memory map
  - Page directory table
    - Page directory entries (PDEs) point to page table
    - One page directory table per process
    - Location in page directory register
  - Page table
    - Page table entries (PTEs) point to page frames
  - Page frame
    - Contains page of data
- TLB accelerates address translation

21.7.1 Memory Organization

Figure 21.5 Virtual address translation.



21.7.1 Memory Organization

- PTE can point to data in pagefile
  - 4 bits determine which pagefile
  - 20 bits indicate offset in pagefile
- PTE has five protection bits
  - Read
  - Write
  - Execute
  - Copy-on-write
  - Raise exception on access
- PTE has three state bits

21.7.1 Memory Organization

Figure 21.6 Page states.

Page state bit	Definition
Valid	PTE is valid—it points to a page of data.
Modified	Page in memory is no longer consistent with the version on disk.
Transition	VMM is in the process of moving the page to or from disk. Pages in transition are always invalid.

21.7.1 Memory Organization

- Large pages
  - VMM treats several pages as one page
- Restrictions
  - Pages must be consecutive in virtual and main memory
  - Large pages allow read and write access
  - Minimum size usually 2MB
  - Size must be multiple of (platform specific) minimum

21.7.1 Memory Organization

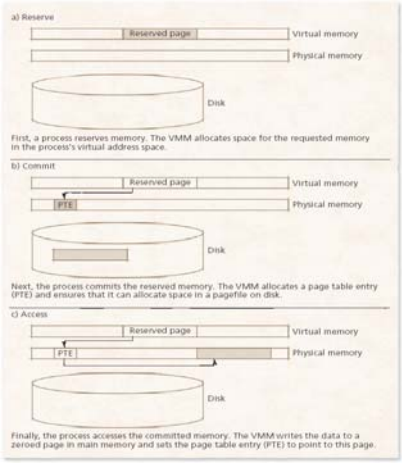
- Copy-on-write pages
  - Process share page frames transparently
- Prototype page tables
  - Enable copy-on-write pages
  - PTEs point to prototype page tables
  - Prototype page table entries (PPTEs) point to page frames that hold copy-on-write pages

21.7.2 Memory Allocation

- 3 stages of memory allocation
  - Reserve
  - Commit
  - Access

21.7.2 Memory Allocation

Figure 21.7 Memory allocation stages.



21.7.2 Memory Allocation

- Optimizations
  - No more must-succeed requests
  - I/O throttling
- Page frame database
  - Table lists state of each page frame

21.7.2 Memory Allocation

Figure 21.8 Page frame states.

Frame State	Definition
Valid	Page belongs to a process's working set and its PTE is set to valid.
Transition	Page is in the process of being transferred to or from disk.
Standby	Page has just been removed from a process's working set; its PTE is set to invalid and in transition.
Modified	Page has just been removed from a process's working set; it is not consistent with the on-disk version. The VMM must write this page to disk before freeing this page. The PTE of this page is set to invalid and in transition.
Modified No-Write	Page has just been removed from a process's working set; it is not consistent with the on-disk version. The VMM must write an entry to the log file before freeing this page. The PTE of this page is set to invalid and in transition.
Free	Page frame does not contain a valid page; however it might contain an invalid page that has no PTE and is not part of any working set.
Zeroed	Page frame is not part of any working set and all of its bits have been set to zero. For security reasons, only zeroed page frames are allocated to processes.
Bad	Page frame has generated a hardware error and should not be used.

21.7.2 Memory Allocation

- Page lists
  - One page list per possible page frame status
  - Linked list
  - Fast access
- Virtual Address Descriptor (VAD)
  - Each VAD describes range of allocated virtual memory
- Clusters
  - Disk divided into groups of pages called clusters
  - At most one file per cluster
- Clustered demand paging
  - Windows XP prefetches entire cluster when process asks for one page

21.7.2 Memory Allocation

- Prefetching
  - Logical Prefetcher
    - Responsible for prefetching
  - Records all memory access during
    - Windows boot time
    - Application start up
  - Next time prefetch needed pages
  - Usually done during device initialization
  - Faster start up

21.7.3 Page Replacement

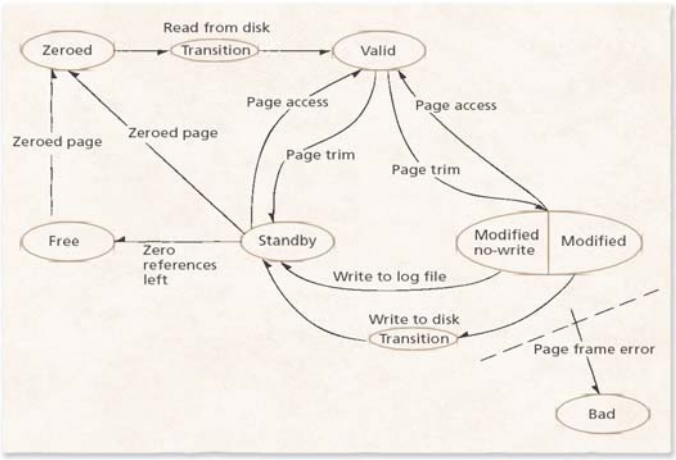
- Working set
  - Pages a process currently has in main memory
- Balance set manager
  - Responsible for trimming working sets
- Localized Least Recently Used
  - Similar to LRU
  - Localized by process

21.7.3 Page Replacement

- Localized LRU
  - Trim all pages above working set maximum
  - Place trimmed pages into
    - Standby Page List
    - Modified Page List
    - Modified No-Write Page List
  - Return to Valid Page List if process access page
  - Otherwise place on Free Page List
  - Zero bits of free pages
  - Move pages to Zeroed Page List
  - Allocate zeroed page to process that requests new page

21.7.3 Page Replacement

Figure 21.9 Page replacement process.



21.7.3 Page Replacement

- Two segments of main memory
  - Paged pool
    - VMM can move pages to pagefile
  - Nonpaged pool
    - VMM never moves pages to pagefile
    - Limited space
    - Used for
      - Device driver code
      - VMM code
      - Interrupt handler code
    - Code in nonpaged pool may not access paged pool

21.8 File Systems

- Three driver layers
  - Volume drivers
    - Low level drivers
    - Interact with data storage hardware devices
  - File system drivers
    - NTFS
    - FAT16 and FAT32
    - CDFS and UDF
  - File system filter drivers
    - Perform high-level functions
    - Virus scanning
    - Encryption

21.8.1 File System Drivers

- Implementing a file system
  - Local file system driver
    - Interacts with hardware devices connected to computer
    - Hard disk drive
    - DVD drive
  - Remote file system driver
    - Interacts with file system driver on remote computer
    - Uses network protocols

21.8.1 File System Drivers

- Typical Disk I/O
  - User-mode thread passes file handle to object manager
  - Object manager passes file pointer to file system driver
  - File system driver passes request to device driver stack
  - Eventually request reaches disk
  - Disk performs requested I/O

21.8.2 NTFS

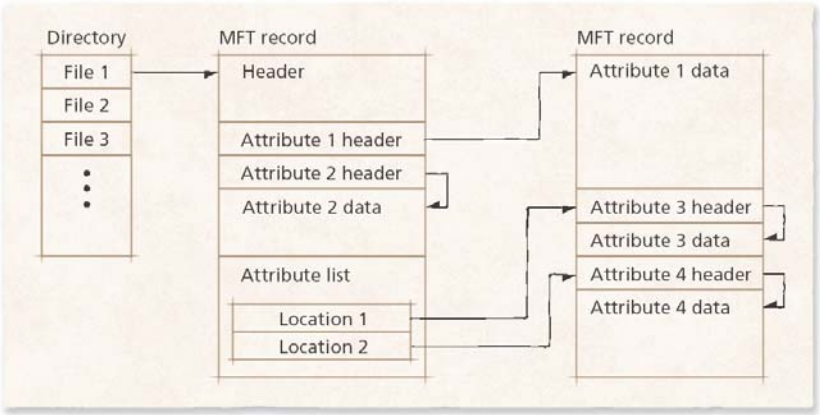
- NTFS overview
  - Native Windows XP file system
  - More secure than FAT
  - Scales well to large disks
    - Cluster size depends on disk size
    - 64-bit file pointers
    - Can address up to 16 exabytes of disk
  - Multiple data streams
  - Compression and encryption
  - Reparse points

21.8.2 NTFS

- Master File Table (MFT)
  - Each file has an MFT entry
  - MFT entry composed of one or more records
  - All data stored as attributes
- Resident attribute
  - Stored entirely within MFT entry
- Non-resident attribute
  - Header stored within MFT entry
  - Data stored elsewhere on disk
  - MFT record contains list of VCN, LCN, Run

21.8.2 NTFS

Figure 21.10 Master File Table (MFT) entry for a sample file.



21.8.2 NTFS

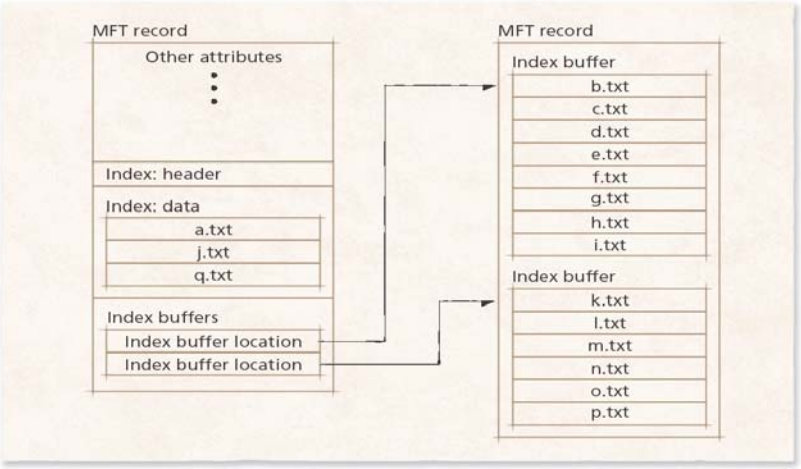
- VCN
  - Virtual cluster number
  - File divided into clusters
  - VCN indicates which number cluster in file
- LCN
  - Logical cluster number
  - Disk divided into clusters
  - LCN indicates on which cluster on disk file portion stored
- Run
  - Indicates number of cluster file portion spans

21.8.2 NTFS

- Directories
  - Files on disk
  - Contain alphabetical list of files
  - Stored as B-trees
  - Each file in directory points to its MFT record

21.8.2 NTFS

Figure 21.11 Directory contents are stored in B-trees.



21.8.2 NTFS

- Multiple directory entries can point to same file
- Hard links always accurate
  - Point to correct file even if original file deleted or moved
  - NFTS treats hard links the same way as original path name
- file_name
  - MFT file attribute
  - One for each path name for file
- hard_link
  - MFT file attribute
  - Number of file_name attributes
  - File deleted only when hard_link drops to zero

21.8.2 NTFS

- Data stream
  - Contents of file
- Default data stream
  - Unnamed data stream
  - Primary contents of file
    - Text of word document
    - Image of bitmap
- Alternate data stream
  - Named data stream
  - Used to store metadata
    - Author, summary, copyright
    - Back up versions

21.8.2 NTFS

- Data streams are MFT attributes
- Data streams can be resident attributes
  - Fast access
  - Shortcut files .link
  - Small text files

21.8.2 NTFS

- File compression
  - Transforms file to take less space on disk
  - Lempel-Ziv Compression Algorithm
  - Transparent
    - Applications access files using standard API calls
    - System compresses and decompresses files
    - Applications unaware if file compressed

21.8.2 NTFS

- Compression unit
  - 16 clusters
  - NTFS compresses each compression unit individually
  - If compressed version still takes 16 clusters, compression unit stored uncompressed
  - Enables random I/O to file
  - NTFS caches recently accessed compression units
    - Likely to be accessed again
    - Lazy-write thread compresses cached units and writes to disk

21.8.2 NTFS

- Encryption
  - Protects files from illicit access
  - Encryption performed in compression units
  - Keys
    - Public key / private key encryption
    - Recovery key given to system administrator
      - In case user forgets password
    - Encrypted versions of keys stored on disk
    - Decrypted keys stored in non-paged pool

21.8.2 NTFS

- Sparse files
  - Examples
    - Bitmap with large white areas
    - Sparse matrix
  - Store large runs of zeroed bits in zero block list
    - Saves space
    - Less overhead than compression
  - Users can specify which blocks are zeroed
  - NTFS automatically detects zeroed compression units
    - Creates entries in zero block list

21.8.2 NTFS

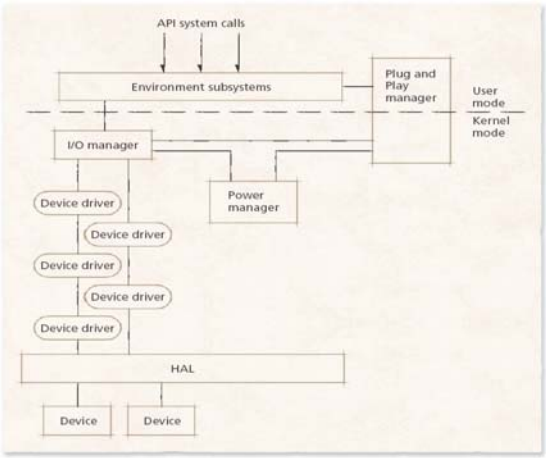
- Reparse point
  - Definition
    - Tag and 16KB data associated with file
    - NTFS uses tag to call file system filter when file accessed
  - Directory junction
    - Directory points to another directory on same volume
    - Makes navigating file system easier
  - Mounted volumes
    - Directory points to root directory on another volume
    - Single directory structure permits access to multiple volumes
    - Access disk drives, floppy drives, DVD drives and etc.

21.9 Input/Output Management

- Environment subsystems
- I/O manager
- Device drivers
- Plug and Play manager
  - Dynamically recognizes new devices
  - Allocates resources to devices: I/O ports, DMA channels
- Power manager
  - Determines each device's and system's power state

21.9 Input/Output Management

Figure 21.12 Windows XP I/O support components.



21.9.1 Device Drivers

- Device driver
  - Controls device
  - Organized into device driver stack
- Low level driver
  - The driver that interacts directly with HAL
  - Example: bus driver
- High level driver
  - Abstracts hardware specifics
  - NTFS
- Intermediate driver
  - Between high level driver and low level driver

21.9.1 Device Drivers

- Device Object
  - Stores data for device
  - Example: type of device
- Driver Object
  - Stores device objects for all devices a driver services
  - Pointers to common driver routines

21.9.1 Device Drivers

- User-mode driver
  - Executes in user-space
  - Specific to environment subsystem
- Kernel-mode driver
  - Execute in context of current thread
  - Device extensions
    - Store data for kernel-mode drivers
    - Part of device object
    - Located in nonpaged memory pool

21.9.1 Device Drivers

- Plug and Play (PnP)
  - Dynamically add new hardware devices
  - Redistribute resources (I/O ports, DMA channels)
- Hardware support
  - Recognize new components
  - Notify system
- Software support
  - Windows XP PnP manager
    - Kernel-mode: configures devices, allocates resources
    - User-mode: interacts with setup programs, notifies user-mode processes of events
  - PnP I/O requests: sent to device drivers

21.9.1 Device Drivers

- Power manager
  - Administers power policy
  - Conservation vs. performance
  - Device power states: D0, D1, D2 and D3
    - D0 full power
    - D3 powered off
  - System power states: S0, S1, S2, S3, S4, S5
    - S0 full power
    - S5 powered off

21.9.1 Device Drivers

- Windows Driver Model (WDM)
  - Bus driver
    - Interact directly with bus (PCI, SCSI)
    - Mandatory one bus driver per bus
    - Provide generic functions for devices on bus
    - Handle PnP I/O requests
  - Filter driver
    - Optional
    - Modify device actions: mouse acceleration
    - Add features: security checks
    - Sort I/O requests among devices on same bus

21.9.1 Device Drivers

- Windows Driver Model (cont.)
  - Function driver
    - Implements main function of device
    - I/O processing
    - Class/miniclass pair implementation
      - Windows XP provides class driver
        - Generic functions
        - E.g. printer
      - Developer provides miniclass driver
        - Device specific function
        - E.g. specific printer model

21.9.1 Device Drivers

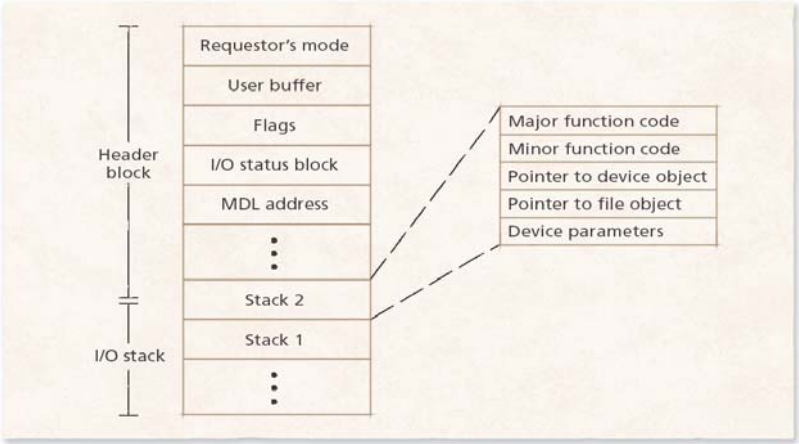
- Windows Driver Model (cont.)
  - All WDM drivers must:
    - Be a bus, filter or function driver
    - Implement
      - Power Management
      - PnP
      - Windows Measurement Instrumentation (WMI)
  - WMI
    - Provides data about hardware to user processes
      - Configuration data
      - Diagnostic data
      - Custom data

21.9.2 Input/Output Processing

- I/O request packet (IRP)
  - Contains all information necessary to complete I/O request
  - I/O status block
    - Success or error code
  - I/O stack
    - I/O stack location
      - One per device driver in stack
      - Major function code: general description of I/O request
      - Minor function code: specific I/O request function
      - Pointer to device object
      - Pointer to file object
      - Device specific parameters

21.9.2 Input/Output Processing

Figure 21.13 I/O request packet (IRP).



21.9.2 Input/Output Processing

Figure 21.14 Major function code examples in Windows XP.

Major function code	Typical reason to send an IRP with this major function code
IRP_MJ_READ	User-mode process requests to read from a file.
IRP_MJ_WRITE	User-mode process requests to write to a file.
IRP_MJ_CREATE	User-mode process requests a handle to a file object.
IRP_MJ_CLOSE	All handles to a file object have been released and all outstanding I/O requests have been completed.
IRP_MJ_POWER	Power manager queries a driver or directs a driver to change the power state of a device.

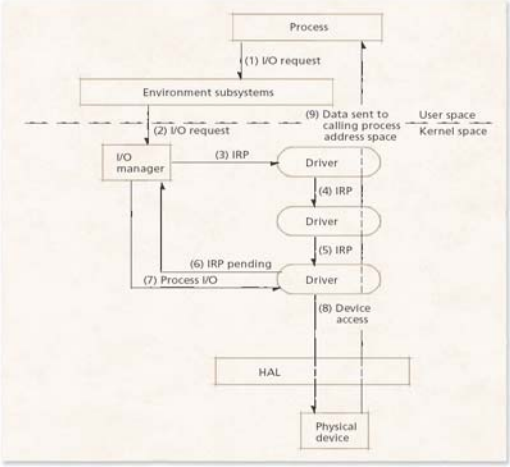
Major function code	Typical reason to send an IRP with this major function code
IRP_MJ_PNP	PnP manager queries a driver, allocates resources to a device or directs a driver to perform some operation.
IRP_MJ_DEVICE_CONTROL	User-mode process calls a device I/O control function to retrieve information about a device or direct a device to perform some operation (e.g., format a disk).

21.9.2 Input/Output Processing

- I/O completion routines
  - Drivers register I/O completion routines with I/O manager
    - Low-level drivers may not
    - I/O manager calls I/O completion routines when IRP processed
  - Drivers specify whether to call routine if IRP
    - Completed successfully
    - Produced and error
    - Cancelled
  - Used to retry failed IRP
  - Dispose of completed IRP

21.9.2 Input/Output Processing

Figure 21.15 Servicing an IRP.



21.9.2 Input/Output Processing

- Synchronous I/O
  - Thread enters *wait* state until I/O completes
- Overlapped I/O
  - Thread stays in ready state during some of the I/O
  - Polling
    - Thread periodically polls device on I/O status
  - Signaling
    - Thread performs some processing
    - Enters *wait* state until an event object enters *signaled* state
  - Alertable I/O
    - System queues APC to signal completed I/O
    - Thread processes APC next time it is in alertable *wait* state

21.9.2 Input/Output Processing

- Overlapped I/O (cont.)
  - I/O completion ports
    - Associate file handles with I/O completion ports
    - Send I/O completion packet to I/O completion port when file I/O complete
    - Several threads register with I/O completion port and block
    - System wakes one thread up to finish processing I/O

21.9.2 Input/Output Processing

- Data transfer techniques
  - Buffered I/O
    - I/O manager creates system buffer
    - All I/O requests done through buffer
    - Does not lock pages in main memory
    - Useful for small data transfers
  - Direct I/O
    - Data transferred directly to/from process virtual address space
    - Memory descriptor list (MDL)
      - Maps virtual addresses to main memory pages
      - Locks pages into main memory
    - Useful for large data transfers

21.9.2 Input/Output Processing

- Data transfer techniques (cont.)
  - Neither I/O
    - Executes in context of calling thread
    - Driver decides whether to use buffered I/O or direct I/O
    - Only high level drives may use neither I/O

21.9.3 Interrupt Handling

- Interrupt Service Routine (ISR)
  - Associated with each device
  - Called by processor on an interrupt
    - Returns false if device not interrupting
    - Returns true and process interrupt otherwise
  - Must execute quickly to avoid masking other interrupts
  - Queues DPC for non-critical aspects
- Interrupt object
  - One per ISR
  - Stores: Interrupt DIRQL, ISR location, interrupt vector
- Interrupt dispatch table
  - Maps hardware interrupts to interrupt vectors

21.9.4 File Cache Management

- Cache manager
  - Maps recently used pages to system virtual memory
  - Single system wide cache for all volumes
  - Dynamically changes size
  - Memory manager moves pages between main memory and disk
- Fast I/O
  - Performing I/O without accessing disk
  - Generally implies using cache
- Clean cache
  - Memory manager moves dirty pages to disk
  - Lazy writer thread flushes dirty pages to disk

21.10 Interprocess Communication

- Data oriented
  - Pipes
  - Mailslots (message queues)
  - Shared memory
- Procedure oriented / object oriented
  - Remote procedure calls
  - Microsoft COM objects
  - Clipboard
  - GUI drag-and-drop capability

21.10.1 Pipes

- Manipulated with file system calls
  - Read
  - Write
  - Open
- Pipe server
  - Process that creates pipe
- Pipe clients
  - Processes that connect to pipe
- Modes
  - Read: pipe server receives data from pipe clients
  - Write: pipe server sends data to pipe clients
  - Duplex: pipe server sends and receives data

21.10.1 Pipes

- Anonymous Pipes
  - Unidirectional
  - Between local processes
  - Synchronous
  - Pipe handles, usually passed through inheritance
- Named Pipes
  - Unidirectional or bidirectional
  - Between local or remote processes
  - Synchronous or asynchronous
  - Opened by name
  - Byte stream vs. message stream
  - Default mode vs. write-through mode

21.10.2 Mailslots

- Mailslot server: creates mailslot
- Mailslot clients: send messages to mailslot
- Communication
  - Unidirectional
  - No acknowledgement of receipt
  - Local or remote communication
  - Implemented as files
  - Two modes
    - Datagram: for small messages
    - Server Message Block (SMB): for large messages

21.10.3 Shared Memory

- File mapping
  - Processes map their virtual memory to same page frames in physical memory
  - Multiple processes access same file
  - No synchronization guaranteed
- File mapping object
  - Maps file to main memory
- File view
  - Maps a process's virtual memory to main memory mapped by file mapping object

21.10.4 Local and Remote Procedure Calls

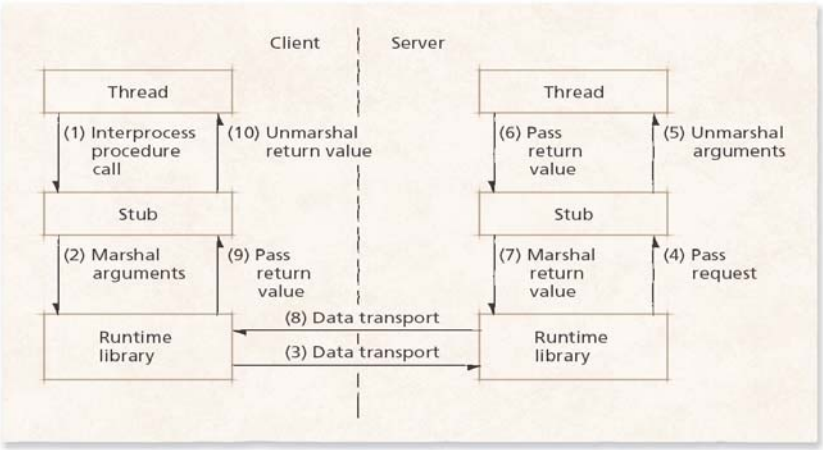
- Server Process
  - Executes procedures
- Client process
  - Calls procedures on the server process
- Local procedure call (LPC)
  - Client process and server process on same machine
  - Only kernel-mode threads may expose LPCs
- Remote procedure call (RPC)
  - Client process and server process on different machines
- Local remote procedure call (LRPC)
  - Client process and server process on same machine
  - Client process uses RPC protocol

21.10.4 Local and Remote Procedure Calls

- Synchronous or Asynchronous communication
- Stub
  - Maps between client process procedure calls and server process procedures
  - Marshals and unmarshals arguments
- Run-time library
  - Passes data across network
  - Any network protocol
    - TCP/IP
    - IPX/SPX

21.10.4 Local and Remote Procedure Calls

Figure 21.16 Flow of an LPC or RPC call.



21.10.4 Local and Remote Procedure Calls

- Microsoft Interface Definition Language (MIDL) file
  - Exposes server process procedures to client processes
  - Header
    - Data global to all interfaces
    - Universally unique identifier (UUID)
    - RPC version number
  - Body
    - Data unique to this interface
    - Variable declarations
    - Function declarations
- Application configuration file
  - Platform specific attributes

21.10.4 Local and Remote Procedure Calls

- Endpoint
  - Server must create an endpoint
    - Network specific address of server process
    - Hardware port
    - Named pipe
  - Client-side runtime library functions establish binding to endpoint

21.10.4 Local and Remote Procedure Calls

- Binding handle
  - Stores necessary information to establish binding
    - Server name
    - Server address
    - RPC protocol sequence
  - Automatic
    - Client calls remote process
    - Runtime library handles all communication
  - Implicit
    - Client determines server to process RPC
    - Runtime library manages binding handle
  - Explicit
    - Client uses binding handle to establish binding
    - Manages all communication
    - May connect to multiple server processes simultaneously

21.10.5 Component Object Model

- COM
  - Transparent location of communicating components
  - Components written in different languages
  - Compiled binaries follow COM protocol
- COM interface
  - COM objects communicate exclusively through interfaces
  - COM object extended by adding new interface
  - Globally Unique Identifier (GUID)
    - 128-bit integer
    - Unique in the world, for all practical purposes
    - Class identifier (CLSID): GUID for object class
    - Interface identifier (IID): GUID for interfaces

21.10.5 Component Object Model

- COM communication
  - COM server registers objects in registry
  - Client uses CLSID to obtain object pointer from registry
  - Client obtains interface
    - Uses this interface to find other interfaces exposed by the COM object
  - Client uses interface to call COM procedures
  - Procedure executes
    - Directly for in-process communication
    - LRPC for cross-process communication
    - DCOM for cross-network communication

21.10.5 Component Object Model

- COM threading models
  - Apartment model
    - One thread acts as server for one object
  - Free thread model
    - Multiple threads act as server for same object
  - Mixed model
    - Both apartment model and free thread model used in same process
- COM extensions
  - COM+: advanced resource management
  - Distributed COM (DCOM): cross-network support
  - ActiveX Controls: self-registering COM with low overhead

21.10.6 Drag and Drop and Compound Documents

- Clipboard
  - Stores data and data format information
  - Accessible by all processes
  - Windows XP defines standard formats: text, bitmap, etc.
  - Processes may define custom formats
- Compound Documents
  - Example: Word document with JPEG images
  - Object Linking and Embedding (OLE)
    - Built on COM
    - Defines standard interfaces
  - Linking vs. embedding objects

21.11 Networking

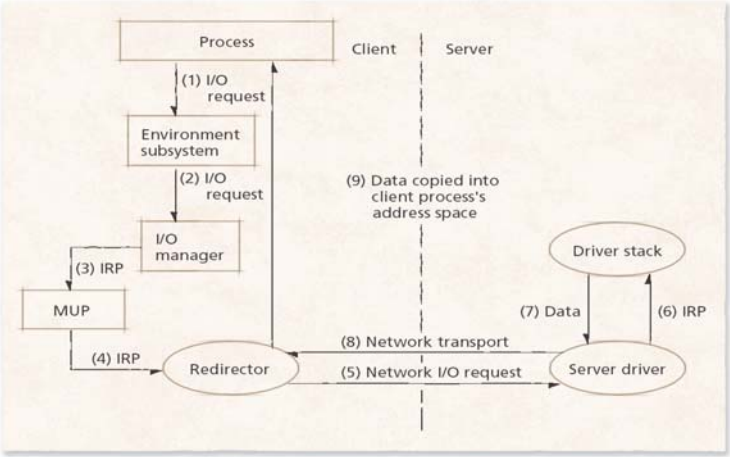
- Network I/O
- Driver architecture
- Protocols
  - Network
  - Transport
  - Application
- Services
  - Active Directory
  - .NET

21.11.1 Network Input/Output

- Transparent to programmers
  - Network I/O done with same API calls as local I/O
- Multiple UNC Provider (MUP)
  - Determines location of remote file
  - Uses Uniform Naming Convention (UNC) format
- Redirectors
  - Kernel-mode file system drivers
  - Interact with server drivers on remote machines to perform I/O

21.11.1 Network Input/Output

Figure 21.17 Handling a network I/O request.



21.11.1 Network Input/Output

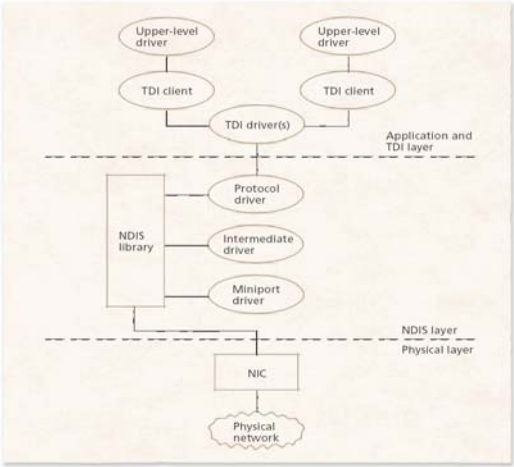
- File sharing protocols
  - Common Internet File System (CIFS)
    - Windows XP native file sharing system
    - Complements HTTP
    - Supports authentication
    - Opportunistic locks (oplocks)
      - Breaking an oplock
  - Web-based Distributed Authoring and Versioning (WebDAV)
    - For collaborative authoring between groups in remote locations
    - Writes data directly to HTTP servers

21.11.2 Network Driver Architecture

- Transport Driver Interface (TDI)
  - Application layer
  - TDI client: low-level application driver
  - TDI driver: interface between TDI client and NDIS
- Network Driver Interface Specification (NDIS)
  - Link layer and some network layer functionality
  - NDIS library: interface between NDIS drivers
  - NDIS protocol driver: place data in packets
  - NDIS intermediate driver: optional, packet filtering
  - NDIS miniport driver: interface between NDIS and NIC
- NIC
  - Physical layer

21.11.2 Network Driver Architecture

Figure 21.18 Network driver architecture.



21.11.3 Network Protocols

- Network layer
  - IP (installed by default)
  - Internetwork Packet eXchange (IPX) (for LANs)
- Transport layer
  - TCP (installed by default)
  - Sequenced Packet eXchange (SPX) for IPX packets

21.11.3 Network Protocols

- Application layer
  - WinHTTP
    - Synchronous and asynchronous communication
    - HTTP, SSL, Kerberos
    - Servers
  - WinINet
    - Interaction with HTTP, FTP, Gopher protocols
    - Clients
  - CIFS
    - Extension of Server Message Block (SMB)
    - Improves access control and handles announcing and naming of networked resources
    - Works over a virtual LAN
    - Used in My Network Places program

21.11.3 Network Protocols

- Winsock
  - Windows sockets 2 (Winsock 2)
  - Can port from BSD sockets with minimal changes
  - Not a transport protocol
    - Socket not required at both ends
    - Runs with a variety of network/transport protocols such as TCP/IP or IPX/SPX
  - Stream or datagram socket
  - Synchronous or asynchronous
  - Protocol transparency
  - Quality of Service (QOS) capability

21.11.4 Network Services

- Active Directory
  - Internet directory
  - Stores information about all shared objects
  - Security features
- Lightweight Data Access Protocol (LDAP)
  - Protocol for accessing an Internet Directory
  - Client/server model
- Remote Access Service (RAS)
  - Remotely connect to LAN over WAN or VPN
- DCOM
  - Distributed computing

21.11.5 .NET

- .NET
  - Replaces DCOM
  - Local applications talk to remote applications to execute user requests
  - Web services
    - Applications accessible over Web
    - Data transported in XML over HTTP
    - Interoperable
  - .NET framework programming model
    - Extended Windows API supports Web Services
    - Visual Basic .NET, Visual C++ .NET, C#
  - .NET component servers
    - Websites, Windows 2003 Server, and etc.

21.12 Scalability

- Five editions of Windows XP (soon a sixth)
- Symmetric multiprocessing
- 64-bit address space
- Windows XP Embedded

21.12.1 Symmetric Multiprocessing (SMP)

- Scheduling
  - Ideal processor and last processor attributes
  - Affinity masks
- Locks
  - Queued spin locks
  - Other fine grained locks (e.g. dispatcher lock)
  - Reduced critical section sizes
- Windows XP 64-Bit Edition
  - 7152 GB of virtual memory
  - 1 terabyte cache memory
  - 12 gigabytes non-paged pool

21.12.1 Symmetric Multiprocessing (SMP)

- Programming features for large scale applications
  - Job objects
  - I/O completion ports
  - Thread pools

21.12.2 Windows XP Embedded

- Components
  - Windows XP divided into over 10,000 components
    - User-space components, such as Notepad
    - Kernel-space components, such as CD-ROM interface or power management tools
  - Developers choose components they need
  - Include Windows XP microkernel
- Uses
  - Embedded devices: printers, routers, etc.

21.12.2 Windows XP Embedded

- Venturcom's RTX extension
  - For Windows XP and Windows XP Embedded
  - Hard real-time guarantees
  - Never mask real-time interrupts
  - 128 levels of scheduling priority
  - Wake threads in priority order, not FIFO

21.13 Security

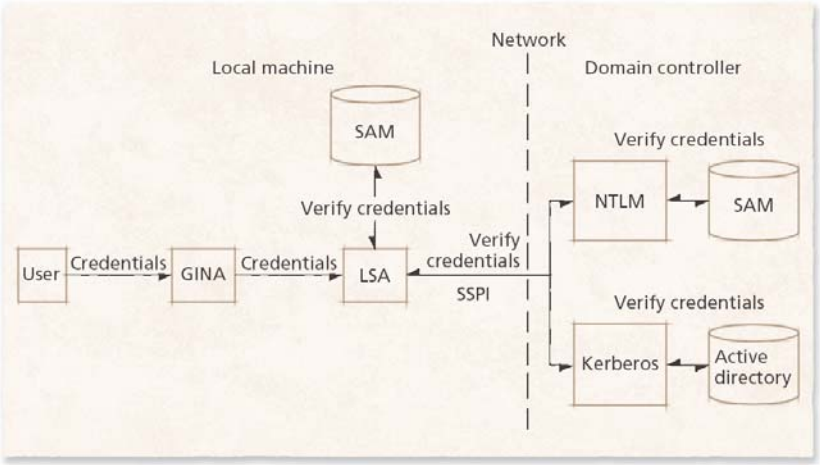
- Windows XP Professional security oriented
- Authentication
- Authorization
- Internet Connection Firewall

21.13.1 Authentication

- Credentials
  - Identity (username)
  - Proof of identity (password)
  - Graphical Identification and Authentication (GINA) DLL
- Authentication
  - Managed by Local Security Authority (LSA)
  - Security Accounts Manager (SAM) database (local)
  - Kerberos version 5 + Active Directory (remote)
  - Network LanMan (NTLM) + SAM database(remote)
  - Security Support Provider Interface (SSPI)

21.13.1 Authentication

Figure 21.19 Authentication process in Windows XP network.



21.13.2 Authorization

- Security principal
  - Any entity that can perform an action
    - User
    - Group
    - Computer
    - Service
  - Security identifier (SID)
    - Uniquely identifies security principal

21.13.2 Authorization

- Access token
  - Stores in security information about security principal
    - Security principal SID
    - All group SIDs
    - Session ID
  - Enables fast user switching
    - Session ID tells which processes to run in background
  - Inheritance
    - Given to all processes and threads owned by security principal

21.13.2 Authorization

- Security descriptor
  - Protect resources: files, programs, printers, etc.
  - Discretionary Access Control List (DACL)
    - Ordered list of Access Control Entries (ACEs)
      - Stores SID
      - What security principal with that SID may do
    - First ACE to match access token SID determines authorization
    - Inclusive or exclusive security policy (or both)

21.13.3 Internet Connection Firewall

- Internet Connection Firewall (ICF) protects
  - Single computer
  - Network
- Packet filtering
  - Outbound traffic recorded in flow table
  - Inbound packets permitted only if match entry in flow table
- Port mapping
  - Some applications require unsolicited inbound traffic
    - Web servers
    - On-line games
    - LDAP
  - Users may explicitly open a port to unsolicited packets

21.13.4 Other Features

- Encrypting File System
- Cookie management
- Control software execution
  - File hash provides extra protection
- Certificates
- Trusted Internet Zones
- Automatic Update
  - Notifies users of security patches
  - Can download and install patches automatically