Windows 2000 / XP

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

32-bit preemptive multitasking operating system for Intel microprocessors

- Key goals for the system:
  - security
  - reliability
  - Windows and POSIX application compatibility
  - high performance
  - extensibility
  - portability
  - international support

- Uses a micro-kernel architecture
- Available in various versions
History

- 1988 @ Microsoft
  - "new technology" (NT) OS
  - OS/2 and POSIX API
- Windows 3.0 popularity
  - NT uses Win32 native API
- Windows XP
  - Part of NT family
  - Goal: unite NT and 95/98 Windows families

Design Principles

- Extensibility — layered architecture
  - Protective mode Executive provides the basic system services
  - User mode server subsystems
  - Modular structure allows additional environmental subsystems to be added without affecting the executive
- Portability — OS 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
  - hardware protection for virtual memory
  - software protection mechanisms for operating system resources
- Compatibility
  - applications that follow the IEEE 1003.1 (POSIX) standard can be complied to run without changing the source code
- Performance
  - subsystems communicate 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
  - different locales via the national language support (NLS) API

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
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 (1/2)

- **Process**
  - has a virtual memory address space
  - information (such as a base priority)
  - affinity for one or more processors

- **Thread**
  - Each process can have one or more threads
  - Threads are the unit of execution scheduled by the kernel's dispatcher
  - Thread has its own state, including a priority, processor affinity, and accounting information

Kernel — Process and Threads (2/2)

- **Thread states:**
  - Ready: waiting to run
  - Standby: highest priority ready thread
  - Running: runs until preempted or end of quorum
  - Waiting: for signal
  - Transition: waiting for resource
  - Terminated
Kernel — Scheduling

- 32-level priority scheme:
  - 0 to 15: variable class
  - 16 to 31: real-time class
  - One queue for each priority

- Priority strategy:
  - Priority is lowered after thread is run
  - Priority is increased at release from wait state
  - Small boost for disk wait
  - Large boost for mouse/keyboard/windows wait
  - 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 — Interrupts

- Software interrupts
  - Asynchronous procedure call
    - Break into executing thread
  - Deferred procedure call
    - Queued

- Hardware interrupts
  - Simple: trap handler based on interrupt dispatch table
  - Complex: exception dispatcher looks for handler

If no handler found then “blue screen of death”
Executive — Object Manager (1/2)

- OS uses objects for all its services and entities
- 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 manipulation via:
  - create
  - open
  - close
  - delete
  - query name: returns name
  - parse: to search for object via name
  - security: allows to check and change access rights

Executive — Object Manager (2/2)

- Object can have permanent or temporary name
- Object names are pathnames
- Symbolic link object allows aliases
- Process gets an object handle
  - by creating an object
  - by opening an existing one
  - by receiving a duplicated handle from another process
  - by inheriting a handle from a parent process
- Each object is protected by an access control list.
Executive — Virtual Memory Manager

- assumes hardware support for
  - virtual to physical mapping
  - paging mechanism
  - transparent cache coherence on multiprocessor systems
  - virtual address aliasing
  - (multiple addresses may refer to same physical memory)
- page-based management scheme with a page size of 4 KB
- two step process to allocate memory:
  - 1. reserves a portion of the process's address space
  - 2. commits the allocation by assigning space in the paging file

2-level Virtual-Memory Layout per process
Executive — Process Manager

- create and delete
  - thread
  - process
  - job
  - set of processes:
    - limits, sizes, processor affinity
- thread assignment to processes

- no parent / child knowledge
- no scheduling

Executive — Local Procedure Call Facility

- client / server model
  - client issues LPC to server
- 3 types of message passing for LPC channel:
  - 1. small messages (< 256 bytes)
    - message is copied
  - 2. large messages
    - sender places message into shared memory section
  - 3. direct read/write operations of the Win32 subsystem
Executive — I/O Manager

- 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 cache manager, which handles caching for the entire I/O system

File I/O

Diagram shows the flow of I/O operations involving a process, cache manager, file system, disk driver, VM manager, and I/O manager. It includes paths for cached I/O, noncached I/O, data copy, and page fault.
Executive — other elements

- Security Reference Monitor
  - Validation and audit checks for every entity in the system
- Plug-and-Play and Power Manager
  - To accommodate device variety
- Registry
  - Database of all system configuration
- System restore utility

Environmental Subsystems

- Win32 subsystem is main operating environment
  - Win32 is used to start all processes
  - Provides all keyboard, mouse and graphical display capabilities
- MS-DOS environment is provided by a Win32 application called the virtual dos machine (VDM)
- 16-Bit Windows Environment
  - Provided by a VDM that incorporates Windows on Windows
  - Provides the Windows 3.1 kernel routines and subroutines for window manager and GDI functions
- Services for UNIX
  - Interix subsystem
  - Designed to run UNIX applications.
File System

- NTFS file system concept: volume
  - Created by disk administrator utility
  - Based on a logical disk partition
  - May occupy portions of a disk, an entire disk, or span across several disks
- All information about volume is stored in a metadata file
- NTFS cluster is the underlying unit of disk allocation
  - A cluster is a number of disk sectors that is a power of two
  - Small cluster size reduces internal fragmentation

File System — Internal Layout

- NTFS logical cluster number (LCN): disk address
- NTFS file is a structured object consisting of attributes
- NTFS file is described by records in the Master File Table (MFT)
- Each file in NTFS volume has a unique ID
  - Called a file reference:
    - 64-bit quantity that consists of a 48-bit file number and a 16-bit sequence number
- NTFS name space is a hierarchy of directories
  - Index root contains the top level of the B+ tree
File System — Recovery

- File system data structure updates are performed as a transaction:
  - 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.

- Logging functionality is provided by a log file service.

Volume Management and Fault Tolerance

- **FtDisk** (fault tolerant disk driver) provides several ways to combine multiple 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* (RAID level 0).
  - Variation: *stripe set with parity*, or RAID level 5.
  - Disk mirroring, (RAID level 1).

- **Bad sector handling**
  - In hardware: *sector sparing*.
  - In software: *cluster remapping*. 
File System — Compression

- 2 methods:
  - Regular file is divided into compression units, which are blocks of 16 contiguous clusters of compressed data
  - Sparse file:
    - Clusters that contain all zeros are not actually stored
    - Instead, gaps are left in the sequence of virtual cluster numbers stored in the meta data 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

- peer-to-peer and client/server networking
- facilities for network management
- internal networking interfaces
  - NDIS (Network Device Interface Specification)
    - Separates network adapters from 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
- implements transport protocols as drivers that can be loaded and unloaded from the system dynamically
Networking — Protocols

- server message block (SMB) protocol
- network basic Input/Output system (NetBIOS)
- NetBEUI (NetBIOS Extended User Interface)
- TCP/IP Internet protocol
- PPTP (Point-to-Point Tunneling Protocol)
- Novell NetWare protocol
- Web Distributed Authoring and Versioning (WebDAV)
- Appletalk

Networking — Dist. Processing Mechanisms

- supports distributed applications via:
  - named NetBIOS
    - communicate done via NetBEUI, NWLink, or TCP/IP
  - named pipes
    - connection-oriented messaging mechanism that are named via the uniform naming convention (UNC)
  - named mail slots
    - connectionless messaging mechanism used for broadcast applications
  - Windows Sockets
    - windows sockets API
  - Network Dynamic Data Exchange (NetDDE).
Distributed Processing Mechanisms (Cont.)

- Remote Procedure Call (RPC)
  - follows Distributed Computing Environment (DCE) standard
  - RPC messages are sent using NetBIOS, or Winsock on TCP/IP networks, or named pipes on LAN Manager networks
  - Uses Microsoft *Interface Definition Language* (IDL)

Networking — Domains

- A domain is a group of machines that share a common security policy and user database
- provides three models of setting up trust relationships:
  - One way: A trusts B
  - Two way transitive:
    - A trusts B, B trusts C so A, B, C trust each other
  - Crosslink:
    - allows authentication to bypass hierarchy to cut down on authentication traffic
Name Resolution in TCP/IP Networks

- Name resolution is the process of converting a computer name to an IP address:
  - e.g., www.cs.fiu.edu resolves to 131.94.125.219

- XP provides several methods of name resolution:
  - Windows Internet Name Service (WINS)
  - Broadcast name resolution
  - Domain name system (DNS)
  - A host file
  - An LMHOSTS file

Programmer Interface — Access to Kernel Objects

- Concept: handle to named kernel object
  - CreateXXX function to open a handle to XXX
  - CloseHandle function disposes of handle (deletes object if the use count drops to 0)

- 3 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: creates duplicate handle to same object, can be passed to another process.
Programmer Interface — Process Management

- 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

Programmer Interface — Memory Management

- Application can allocate memory
- Application can map a file into its address space
  - Dummy file handle be used to share memory among processes
- Application can use heap memory:
  - A Win 32 process is created with a 1 MB default heap
  - Thread can lock heap to prevent access by concurrent threads
- Application can request thread-local memory
  - Used for functions that fail to work properly in a multithreaded environment