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).

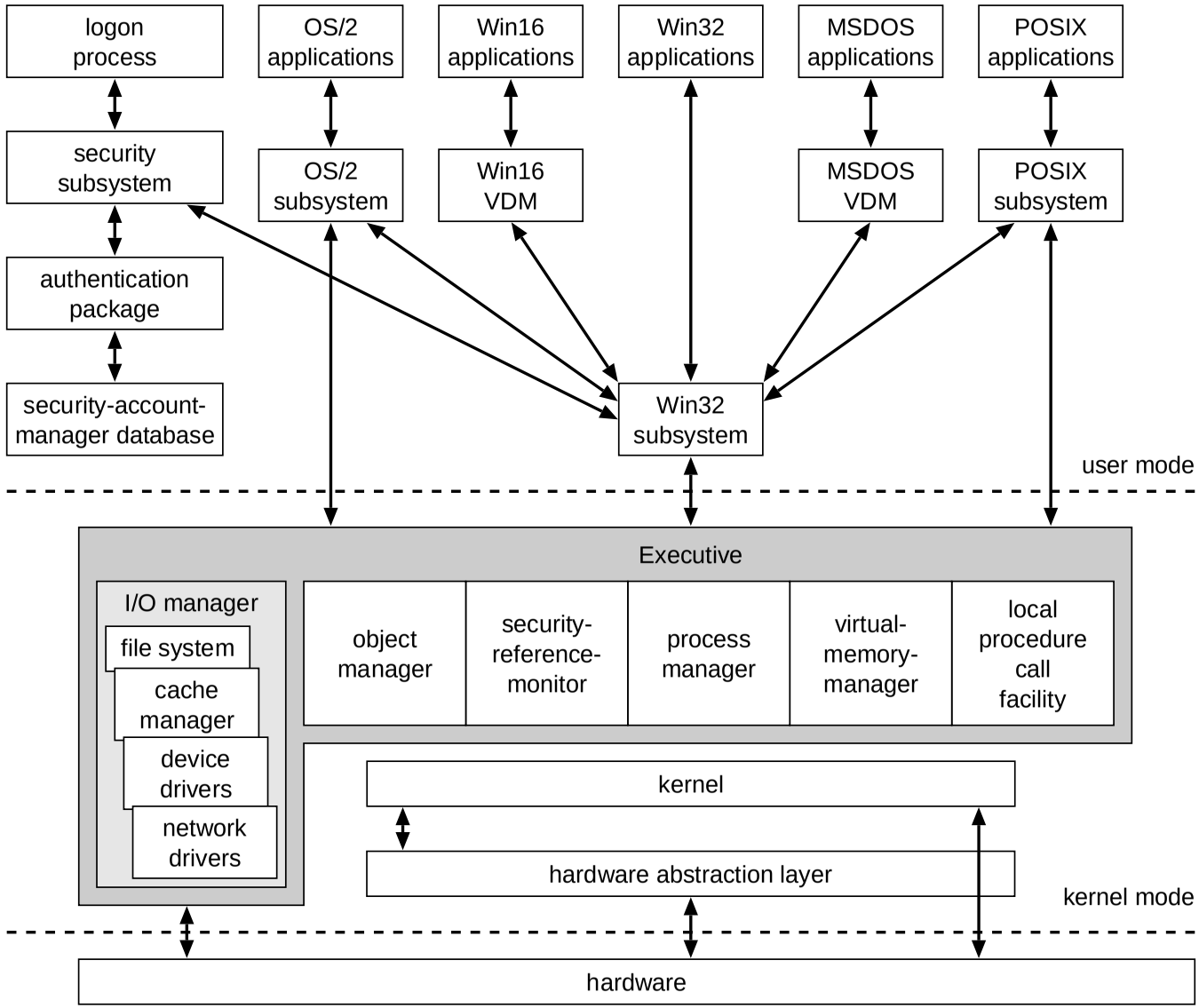
Design Principles (Cont.)

- 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.

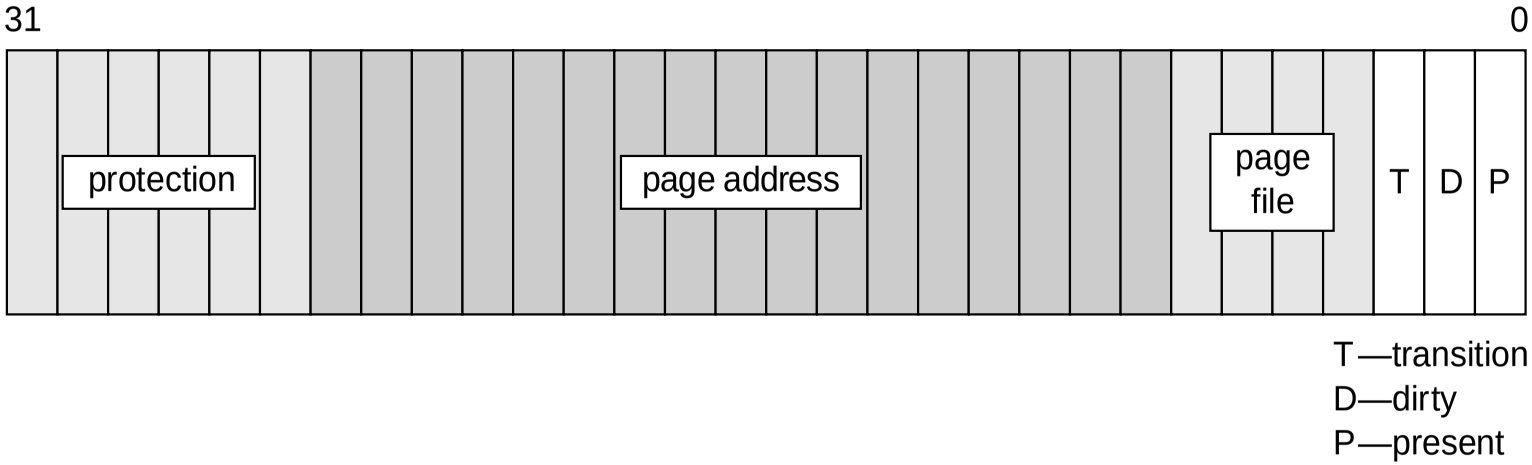
Executive — Virtual Memory Manager

- 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.

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.