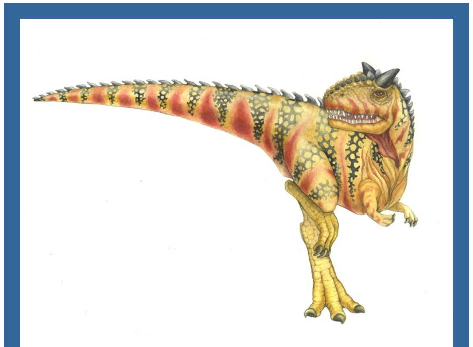
Chapter 21: Windows 10



Chapter 21: Windows

- History
- Design Principles
- System Components
- Terminal Services and Fast User Switching
- File System
- Networking
- Programmer Interface

Objectives

- Explore the principles upon which Windows 10 is designed and the specific components involved in the system
- Provide a detailed explanation of the Windows 10 file system
- Illustrate the networking protocols supported in Windows 10
- Describe the interface available in Windows 10 to system and application programmers
- Describe the important algorithms implemented in Windows 10

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.
- Many versions of Windows along the way, XP, Vista, 7, 8, and now 10
- Many older versions still in use, less secure, some unpatched

Windows 10

- 32- and 64-bit preemptive multitasking operating system for Intel 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
- Several versions with different prices, for devices/laptops/desktops and servers
- Win 10 has an app store, Windows Desktop Bridge to run older binaries, **Pico Providers** replacing the old multiple subsystem concept, **Hyper-V** virtualization, multiuser, distributed services, remote GUI, and other advanced features

Design Principles

- Security -
  - **Access control lists** (**ACLs**) – both attribute-based and claim-based
  - Rudimentary capabilities functionally called **integrity levels**
  - File system and communication encryption
  - Exploit mitigations – **address-space layout randomization** (**ASLR**), **Data Execution Prevention** (**DEP**), **Control-Flow Guard** (**CFG**), and **Arbitrary Code Guard** (**ACG**)
  - Several digital signature facilities
  - **Device Guard** option for fine grain control over what signers code is allowed on the system

Design Principles (Cont.)

- Extensibility — layered architecture
  - **Remote procedure calls** (**RPCs**)
  - **Advanced local procedure calls** (**ALPCs**)
- Portability — Windows 10 can be moved from one hardware architecture to another with relatively few changes
  - Written in C and C++
  - Processor-specific portions are written in assembly language for a given processor architecture (small amount of such code).
  - Platform-dependent code is isolated in a dynamic link library (DLL) called the “hardware abstraction layer” (HAL)

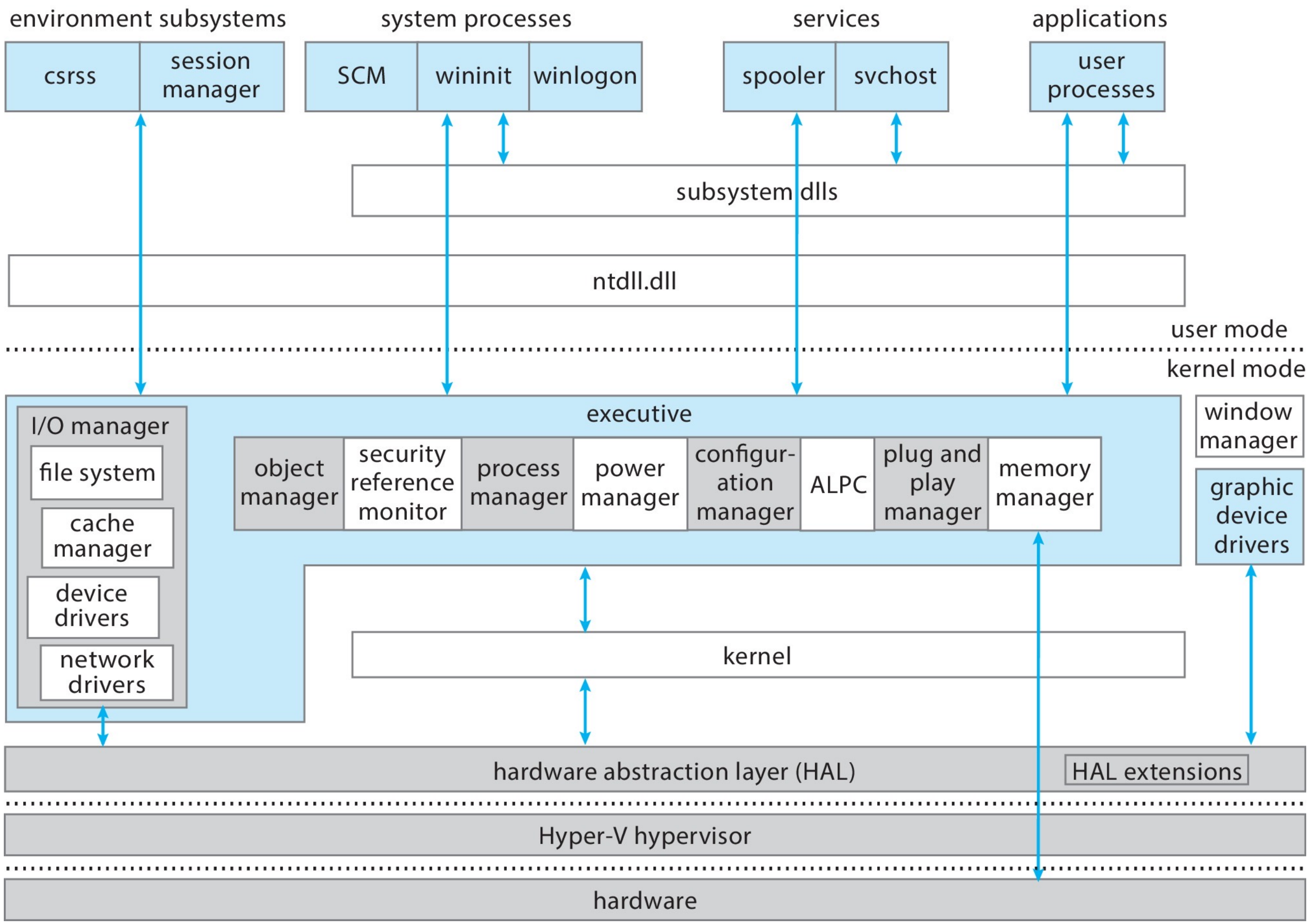
Design Principles (Cont.)

- Reliability — Windows 10 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 complied to run on 10 without changing the source code
- Performance — Windows 10 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
- Energy efficiency – for portable devices, etc, includes dynamic tick feature, process lifetime management, desktop activity monitor, connected standby

Windows 10 Architecture

- Layered system of module
- Protected mode — **hardware abstraction layer** (**HAL**), kernel, executive
- User mode — collection of subsystems
  - Environmental subsystems emulate different operating systems
  - Protection subsystems provide security functions

Depiction of 10 Architecture



System Components

- Layered system of modules operating at specific privilege layers
- Kernel and user mode of course
- **Virtual Trust Levels** (**VTLs**) option implemented by Hyper-V virtualization
  - Enables virtual secure mode
  - **Normal World** (VTL 0) and **Secure World** (VTL 1)
  - Within each world are user and kernel modes
  - Secure world has a secure kernel and executive and a collection of **trustlets**
  - Bottommost layer runs in special processor mode (VMX Root Mode on Intel) including Hyper-V hypervisor, creating hardware-based normal-to-secure-world boundary

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)
- VSM Enclaves allow valid signed third-party code to perform crypto calculations

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 one of eight states: *initializing*, *ready, deferred-ready, standby, running, waiting, transition*, and *terminated*
- Each thread has two modes of execution: **user-mode thread** (**UT**) and **kernel-mode thread** (**KT**)
  - Each has two stacks, one for each mode
  - Kernel layer runs trap handler to switch stacks and change CPU mode

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 Windows 10’s priority strategy
  - Trends 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
  - Complete-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 10 does not guarantee that a real-time thread will start to execute within any particular time limit
  - This is known as *soft realtime*

Windows x86 Interrupt Request Levels

interrupt levels	types of interrupts
31	machine check or bus error
30	power fail
29	interprocessor notification (request another processor to act; e.g., dispatch a process or update the TLB)
28	clock (used to keep track of time)
27	profile
3–26	traditional PC IRQ hardware interrupts
2	dispatch and deferred procedure call (DPC) (kernel)
1	asynchronous procedure call (APC)
0	passive

Kernel — Trap Handling

- The kernel provides trap handling when exceptions and interrupts are generated by hardware of 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

- Windows 7 uses objects for all its services and entities; the object manger 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`.

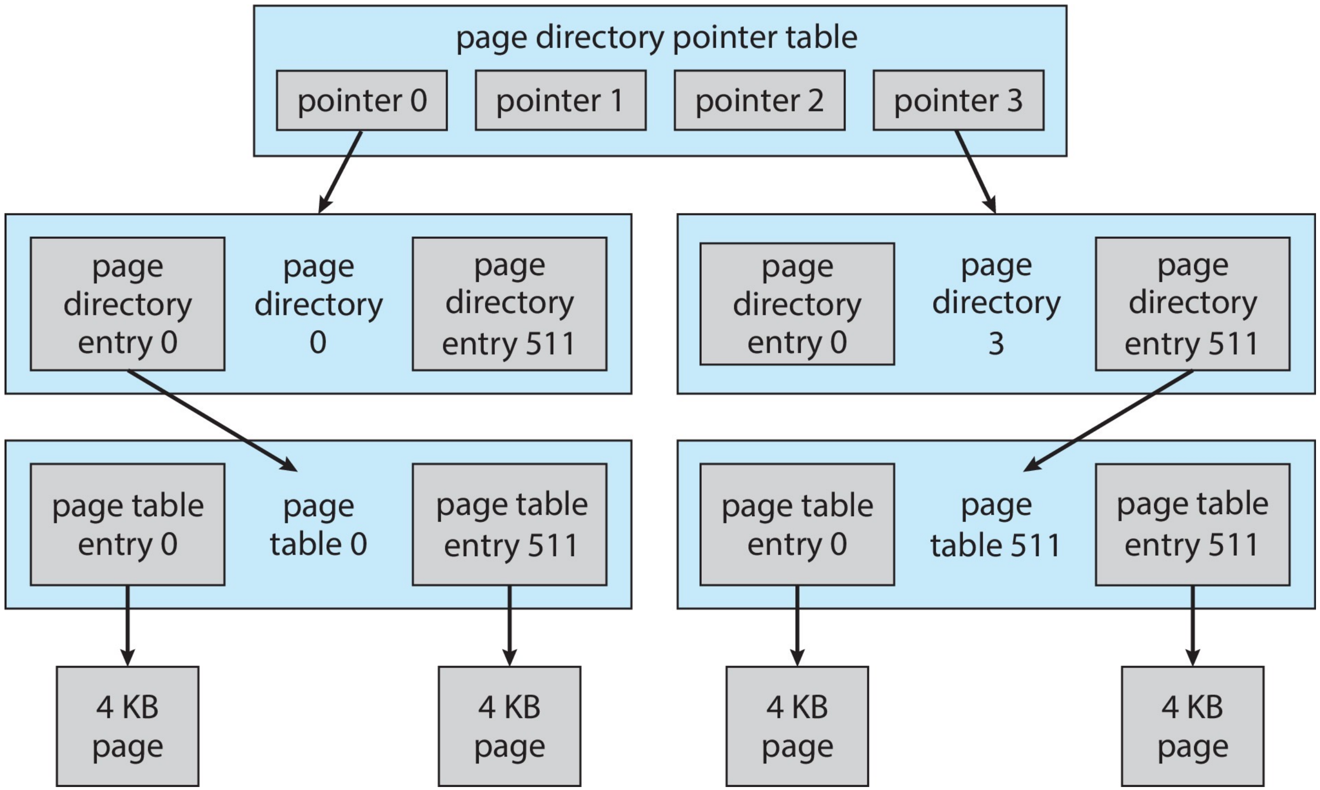
Executive — Naming Objects

- The Windows 7 executive allows almost any object to be given a name, which may be either permanent or temporary
  - Exceptions are process, thread and some others object types
- Object names are structured like file path names in MS-DOS and UNIX
- Windows 7 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 addressing aliasing.
- The VM manager in Windows 10 uses a page-based management scheme with a whatever page sizes are supported by hardware (4 KB, 2MB, 1GB)
- The Windows 10 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 system's paging file(s)

Virtual-Memory Layout

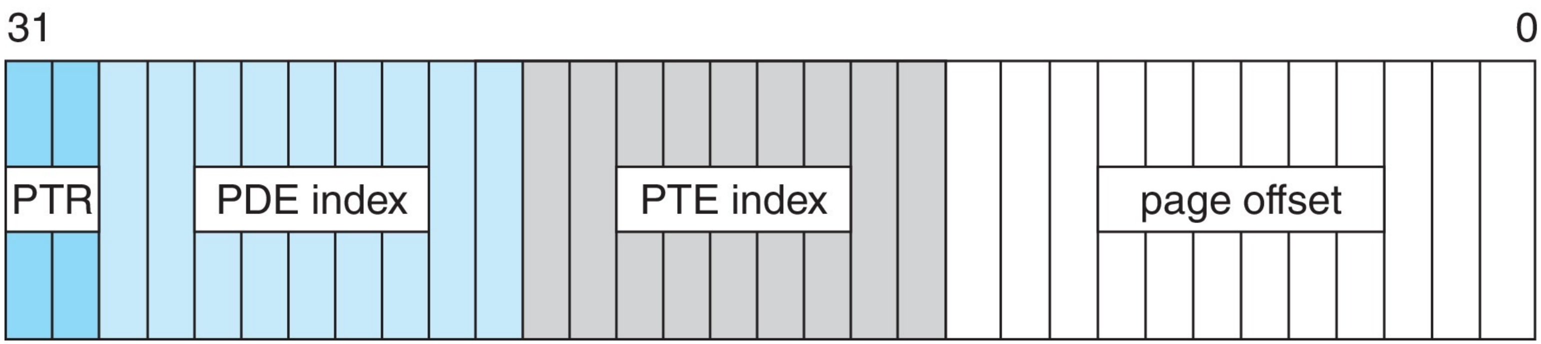


Virtual Memory Manager (Cont.)

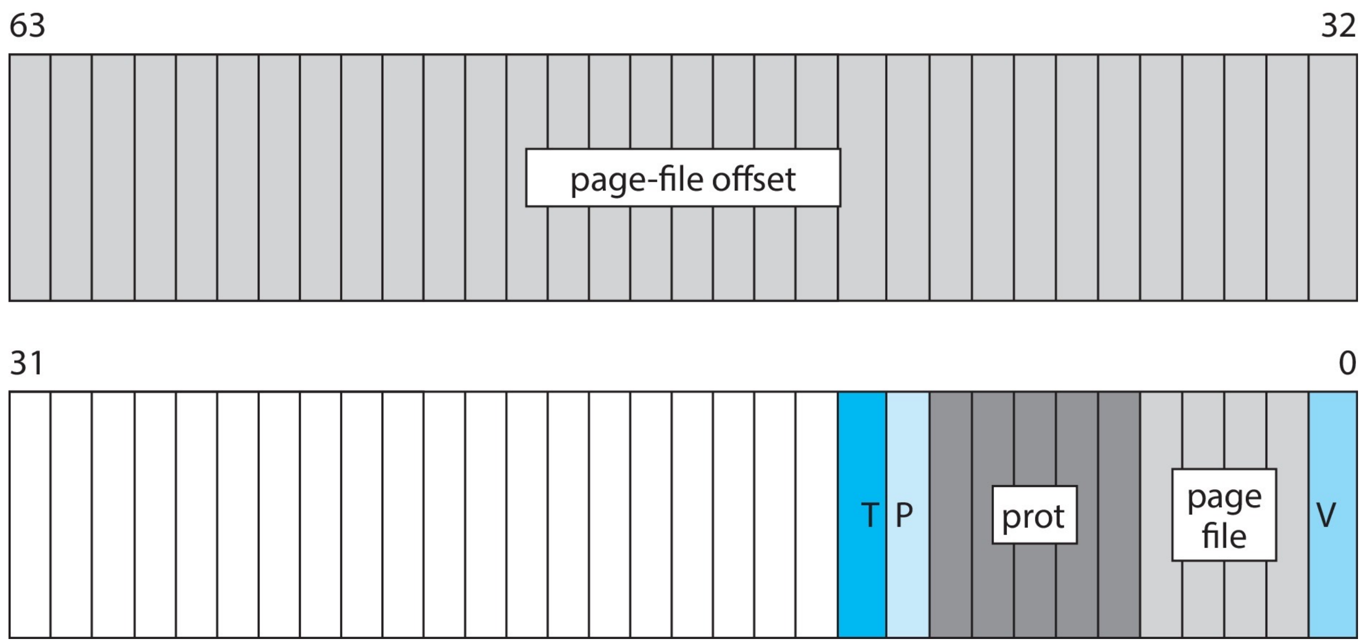
- The virtual address translation in Windows 10 uses several data structures
  - Each process has a *page directory* that contains 1024 *page directory entries* of size 4 bytes
  - Each page directory entry points to a *page table* which contains 1024 *page table entries* (PTEs) of size 4 bytes
  - Each PTE points to a 4 KB *page frame* in physical memory
- A 10-bit integer can represent all the values form 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 bye address in physical memory
- A page can be in one of six states: valid, zeroed, free standby, modified and bad

Virtual-to-Physical Address Translation on IA-32

- 2 bits index into four PDEs, 9 bits for page directory entry, 9 bits for page table entry, and 12 bits for byte offset in page



Page File Page-Table Entry



V=0, T=0, P=0

5 bits for page protection, 32 bits for page-file offset, frame address, 4 bits to select a paging file, and 20 bits for additional bookkeeping

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
- Processes represented by job object
- Docker container support via job objects called **silos**

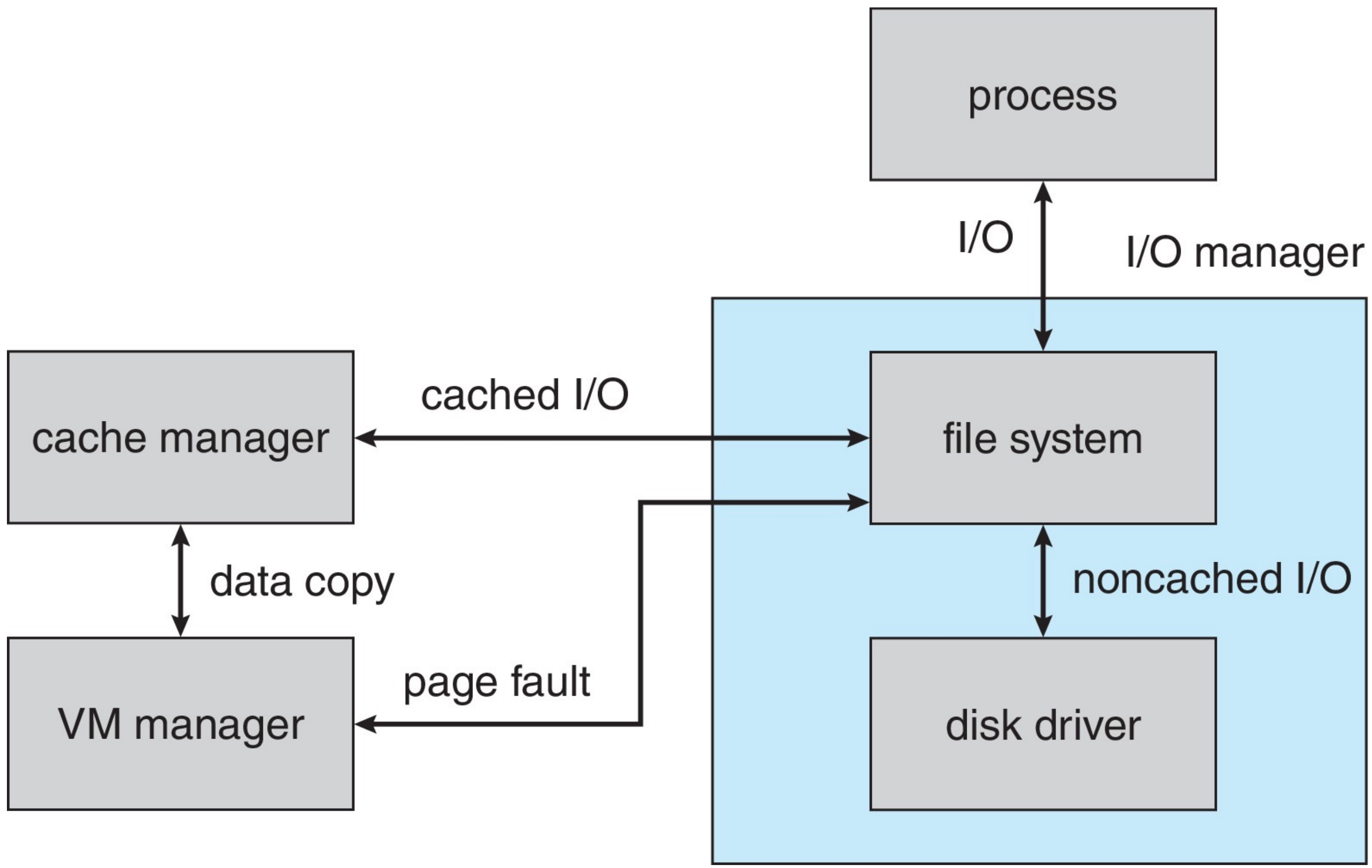
Executive —
Local Procedure Call Facility

- The LPC passes requests and results between client and server processes within a single machine
- When a 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 was 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 Windows 10 **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

File I/O


process
I/O
I/O manager
cache manager
cached I/O
file system
data copy
noncached I/O
VM manager
page fault
disk driver

Executive — Security Reference Monitor

- The object-oriented nature of Windows 10 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

Executive – Plug-and-Play Manager

- Plug-and-Play (PnP) manager is used to recognize and adapt to changes in the hardware configuration
- When new devices are added (for example, PCI or USB), the PnP manager loads the appropriate driver
- The manager also keeps track of the resources used by each device

Executive – Power Manager

- Detects current power conditions
- Puts system to sleep, hibernation
- Processor power manager for core parking, CPU throttling and boosting, etc.
- Device state management

Executive – Registry

- Configuration information kept in internal repositories called **hives**
- Managed by configuration manager called **registry**
- Separate hives for system information, each user’s preferences, software information, security and boot options
- Windows creates **system restore point** before making registry changes to be able to recover previous registry if something goes wrong

Executive – Booting

- Windows starts booting via BIOS firmware
- UEFI includes **Secure Boot feature** that provides integrity checking via digital signature verification of all firmware and boot-time components
- Firmware does **Power-on self-test** (**POST**)
- If machine is hibernating, state restored via `winresume.efi` program
- Kernel initializes, starts several processes including **idle**, **system process**, **secure system process** and **session manager subsystem** (**SMSS**)

File System

- The fundamental structure of the Windows 10 file system (NTFS) is a *volume*
  - Created by the Windows 10 disk administrator utility
  - Based on a logical disk partition
  - May occupy a portions 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 48-bit file number and a 16-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 that are logged
  - 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 Windows 7 *log file service.*

File System — Security

- Security of an NTFS volume is derived from the Windows 7 object model
- Each file object has a security descriptor attribute stored in this 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 Windows 7, 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 tow 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

File System — Reparse Points

- A reparse point returns an error code when accessed
- The reparse data tells the I/O manager what to do next
- Reparse points can be used to provide the functionality of UNIX *mounts*
- Reparse points can also be used to access files that have been moved to offline storage

Networking

- Windows 10 supports both peer-to-peer and client/server networking; it also has facilities for network management
- To describe networking in Windows 10, 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
- Windows 7 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:
  1. `Session control`
  2. `File`
  3. `Printer`
  4. `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 of sessions between two logical names on the network
    - Support reliable data transfer for a session via NetBIOS requests or *SMBs*

Networking — Protocols (Cont.)

- Windows uses the TCP/IP Internet protocol version 4 and version 6 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 Windows machines that are connected over the Internet
- The Data Link Control protocol (DLC) is used to access IBM mainframes and HP printers that are directly connected to the network (possible on 32-bit only versions using unsigned drivers)

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 with 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

- Windows uses the concept of a domain to manage global access rights within groups
- A domain is a group of machines running Windows that share a common security policy and user database
- Windows 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

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

- 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.
- Windows 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 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

- 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:
  1. `IDLE_PRIORITY_CLASS` (priority level 4)
  2. `BELOW_NORMAL_PRIORITY CLASS` (NT priority level 6)
  3. `NORMAL_PRIORITY_CLASS` (level 8 — typical for most processes
  4. `ABOVE_NORMAL_PRIORITY_CLASS` (level 10)
  5. `HIGH_PRIORITY_CLASS` (level 13)
  6. `REALTIME_PRIORITY_CLASS` (level 24)
- To provide performance levels needed for interactive programs, Windows has a special scheduling rule for processes in the `NORMAL_PRIORITY_CLASS`
  - 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, increases the scheduling quantum by some factor, typically 3

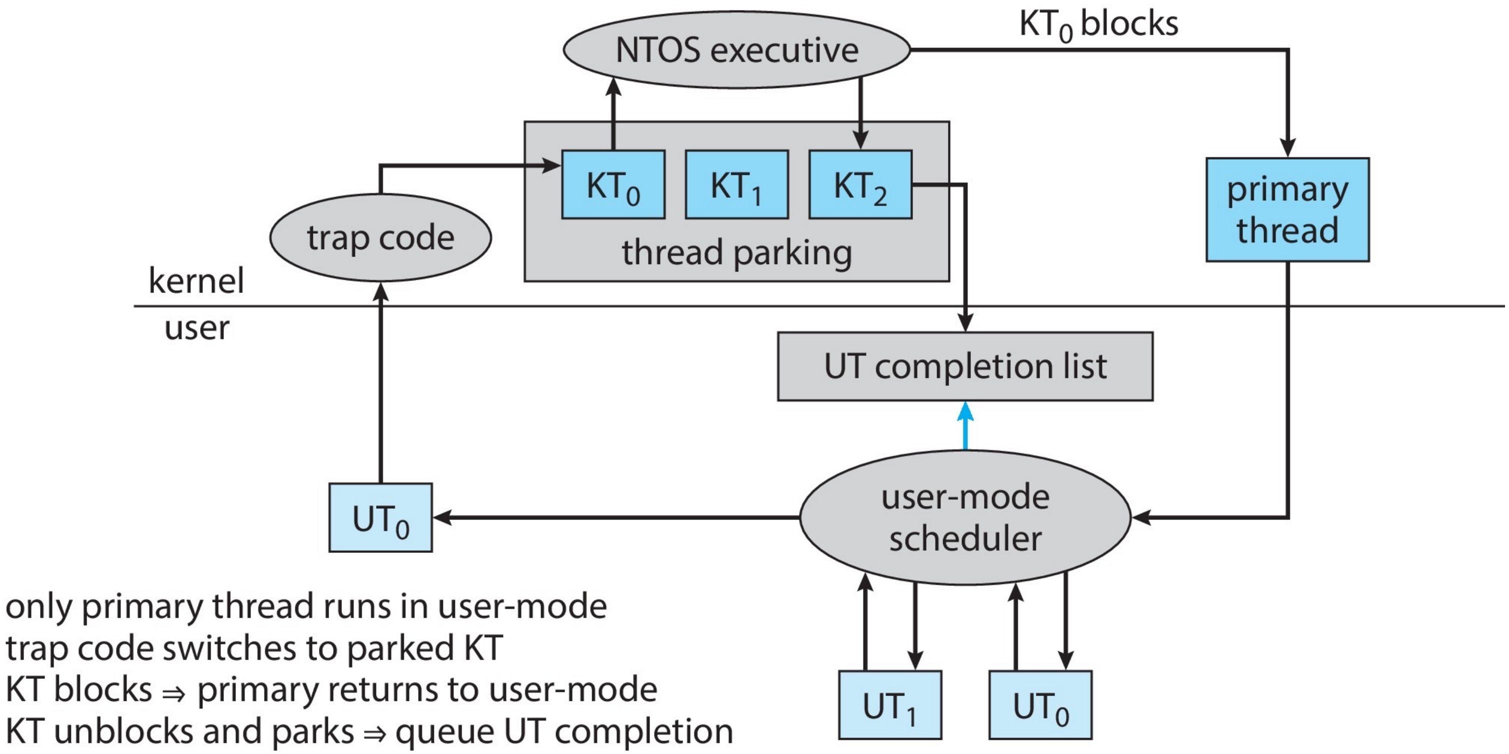
Process Management (Cont.)

- 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
  - Windows includes fibers to facilitate the porting of legacy UNIX applications that are written for a fiber execution model
- Windows also introduced user-mode scheduling for 64-bit systems which allows finer grained control of scheduling work without requiring kernel transitions

User-mode Scheduling



Programmer Interface — Interprocess Communication

- Win32 applications can have interprocess communication by sharing kernel objects
- An alternate means of interprocess communications 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 Win 32 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

End of Chapter 21