Chapter 19: Windows 7

S.Lakshmi Priya
AP/CSE

Objectives

- To explore the principles upon which Windows 7 is designed and the specific components involved in the system
- To understand how Windows 7 can run programs designed for other operating systems
- To provide a detailed explanation of the Windows 7 file system

Windows 7

- 32-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
- Available in six client versions, Starter, Home Basic, Home Premium, Professional, Enterprise and Ultimate. With the exception of Starter edition (32-bit only) all are available in both 32-bit and 64-bit.
- Available in three server versions (all 64-bit only), Standard, Enterprise and Datacenter

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
  - 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 —Windows 7 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)

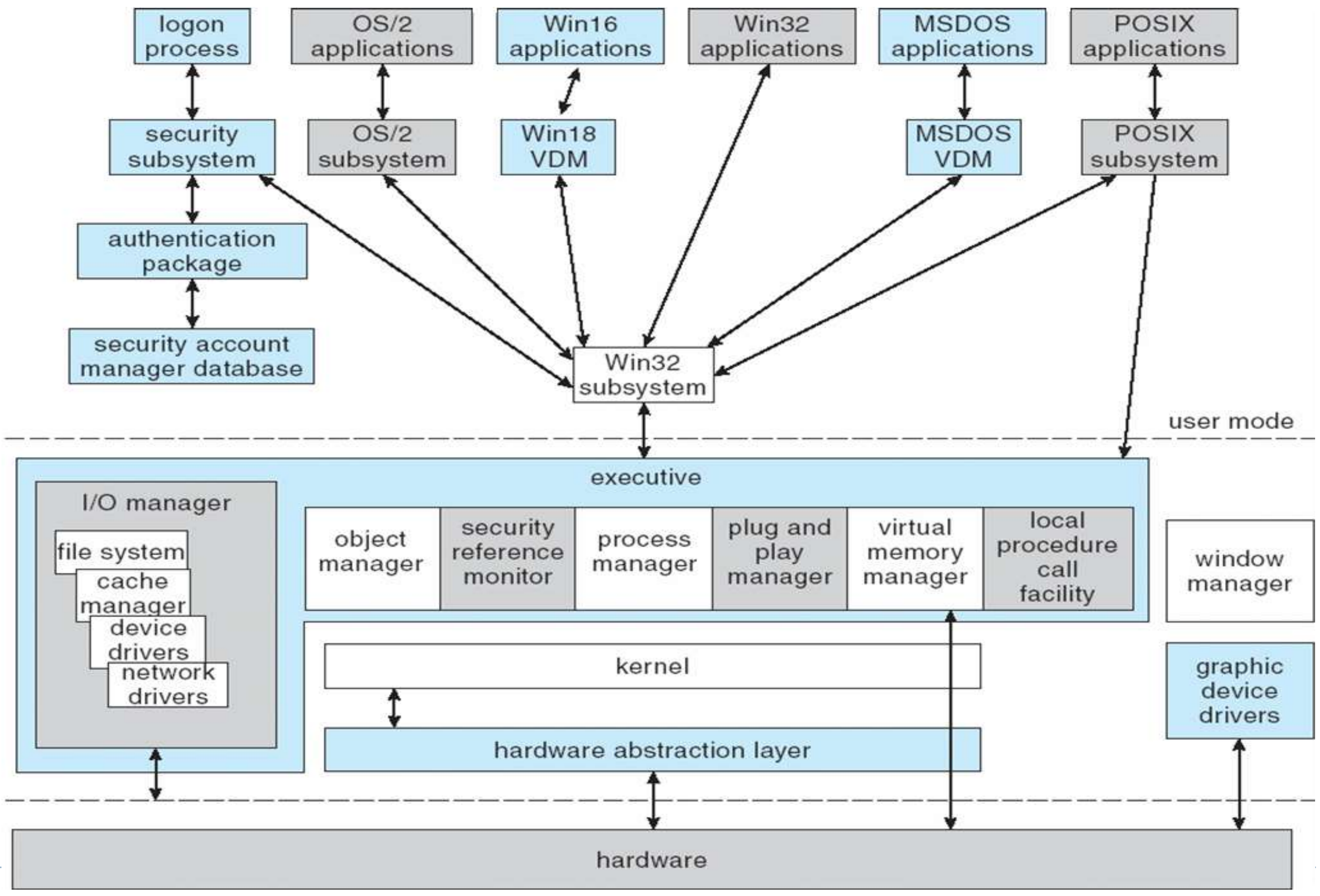
Design Principles (Cont.)

- Reliability — Windows 7 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 7 without changing the source code
- Performance — Windows 7 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

Windows 7 Architecture

- Layered system of module
- Protected mode — hardware abstraction layer (HAL), kernel, executive runs in this mode
- User mode — collection of subsystems and services run in this mode
  - Environmental subsystems emulate different operating systems
  - Protection subsystems provide security functions
- HAL
  - layer of software that hides hardware chipset differences from upper levels of the operating system.
  - chipset-specific details of mapping memory, configuring I/O buses, setting up DMA, and coping with motherboard-specific facilities are all provided by the HAL interfaces.
  - single set of kernel and driver binaries for a particular CPU to be used with different chipsets simply by loading a different version of the HAL.

Depiction of 7 Architecture



System Components — Kernel

- Four main responsibilities:
  - thread scheduling
  - interrupt and exception handling
  - low-level processor synchronization
  - switching between user mode and kernel mode.
- 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

- A 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 six states: ready, standby, running, waiting, transition, and terminated.
  - The highest-priority ready thread is moved to the standby state, which means it is the next thread to run.
  - A thread is in the transition state while it waits for resources necessary for execution;

Kernel — Scheduling

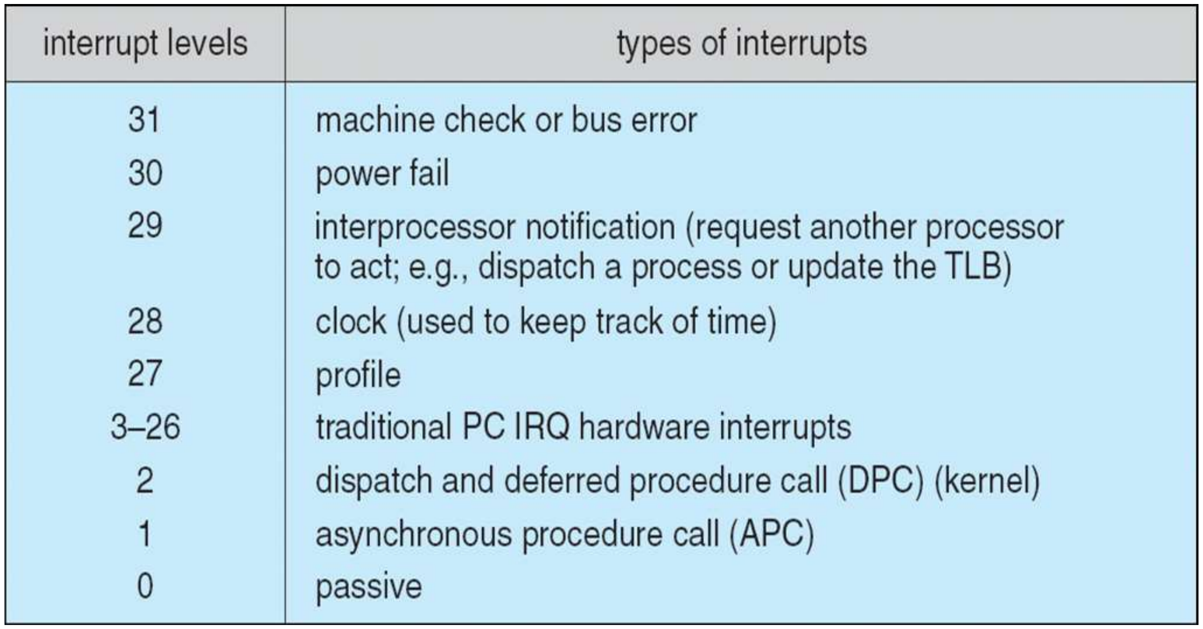
- The dispatcher uses a 32-level priority scheme to determine the order of thread execution.
  - Priorities are divided into two classes
    - The real-time class contains threads with priorities ranging from 16 to 31
    - The variable class contains threads having priorities from 0 to 15
  - Different scheduling queues for each scheduling priority and dispatcher traverses from highest to lowest queue.
  - If thread does not have affinity to the available processor it is skipped and dispatcher looks for other ready threads.
  - If no thread available, dispatcher runs the idle thread

Kernel — Scheduling (Cont.)

- When does a scheduling decision take place?
  - 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 7 does not guarantee that a real-time thread will start to execute within any particular time limit .
  - This is known as soft realtime.

Windows 7 Interrupt Request Levels

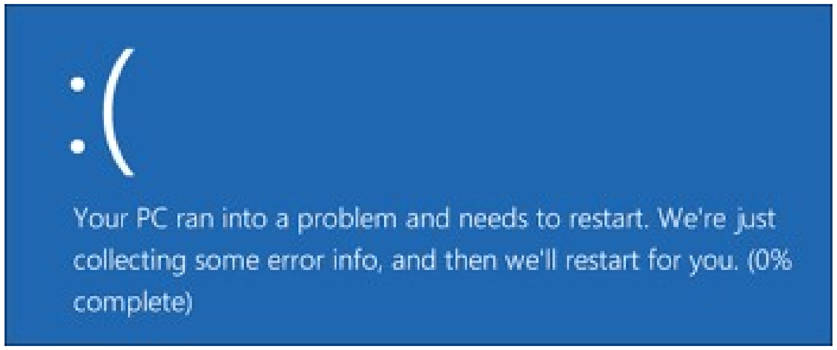
- Interrupts managed through interrupt objects each of which is associated with an ISR.
- Interrupt dispatch table is maintained to bind each interrupt level to a service routine.
- 32 interrupt levels :- 8 reserved for kernel and 24 for hardware interrupts from HAL



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.
- Exceptions that cannot be handled by the trap handler are handled by the kernel's exception dispatcher.
- When an exception occurs in kernel mode, the exception dispatcher simply calls a routine to locate the exception handler.
  - If no handler is found, a fatal system error occurs, and the user is left with the infamous "blue screen of death" that signifies system failure.



:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/blue-screen-data

Executive

- Windows executive provides a set of services that all environmental subsystems use
- Services are grouped as follows: object manager, virtual memory manager, process manager, advanced local procedure call facility, I/O manager, cache manager, security reference monitor, plug-and-play and power managers, registry, and booting

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 – right to use the object (ACL), checks quotas
  - Keeps track of which processes are using each object
- Objects are manipulated by a standard set of methods, namely create, open, close, delete, etc.,
- Eg for objects : semaphores, mutex, files, sections, ports etc.,

- User-mode code accesses these objects using an opaque value called a **handle**
- Each process has a **handle table** containing entries that track the objects used by the process.
- Kernel-mode code can access an object by using either a handle or a **referenced pointer**.

Executive — Object Manager (Contd.,)

- Processes and object handles
  - 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.
  - When a process exits, all its open handles are implicitly closed.
- The object manager keeps track of two counts for each object:
  - the number of handles for the object - number of handles that refer to the object in the handle tables of all processes,and
  - the number of referenced pointers - referenced pointer count is incremented whenever a new pointer is needed by the kernel and decremented when the kernel is done with the pointer
  - The purpose of these reference counts is to ensure that an object is not freed while it is still referenced by either a handle or an internal kernel pointer.

Executive — Virtual Memory Manager

- The VM manager in Windows 7 uses a page-based management scheme with a page size of 4 KB.
- The Windows 7 VM manager uses a two step process to allocate memory
  - The first step reserves a portion of the process's virtual address space
  - The second step commits the allocation by assigning space in the physical memory
- Windows implements shared memory by defining a **section object**
- After getting a handle to a section object, a process maps the memory of the section to a range of addresses, called a **view**
  - A process can establish a view of the entire section or only the portion it needs.
  - Memory protection of pages in the section can be set to read-only, read-write, read-write-execute, execute-only, no access, or copy-on-write.
    - No-access page : raises an exception if accessed (Eg: access virtual addresses that are not committed to memory)
    - Copy-on-write: page shared until one of the processes using it wants to modify

Virtual Memory Manager (Cont.)

- The virtual address translation in Windows 7 uses several data structures
  - Eg: In IA-32 and AMD64 processors,
    - Each process has a page directory that contains 512 page directory entries of size 4 bytes.
    - Each page directory entry points to a page table which contains 512 page table entries (PTEs) of size 4 bytes.
    - Each PTE points to a 4 KB page frame in physical memory.
- A translation look aside buffer (TLB) is also used to cache the frequently accessed pages

Page table layout in IA-32

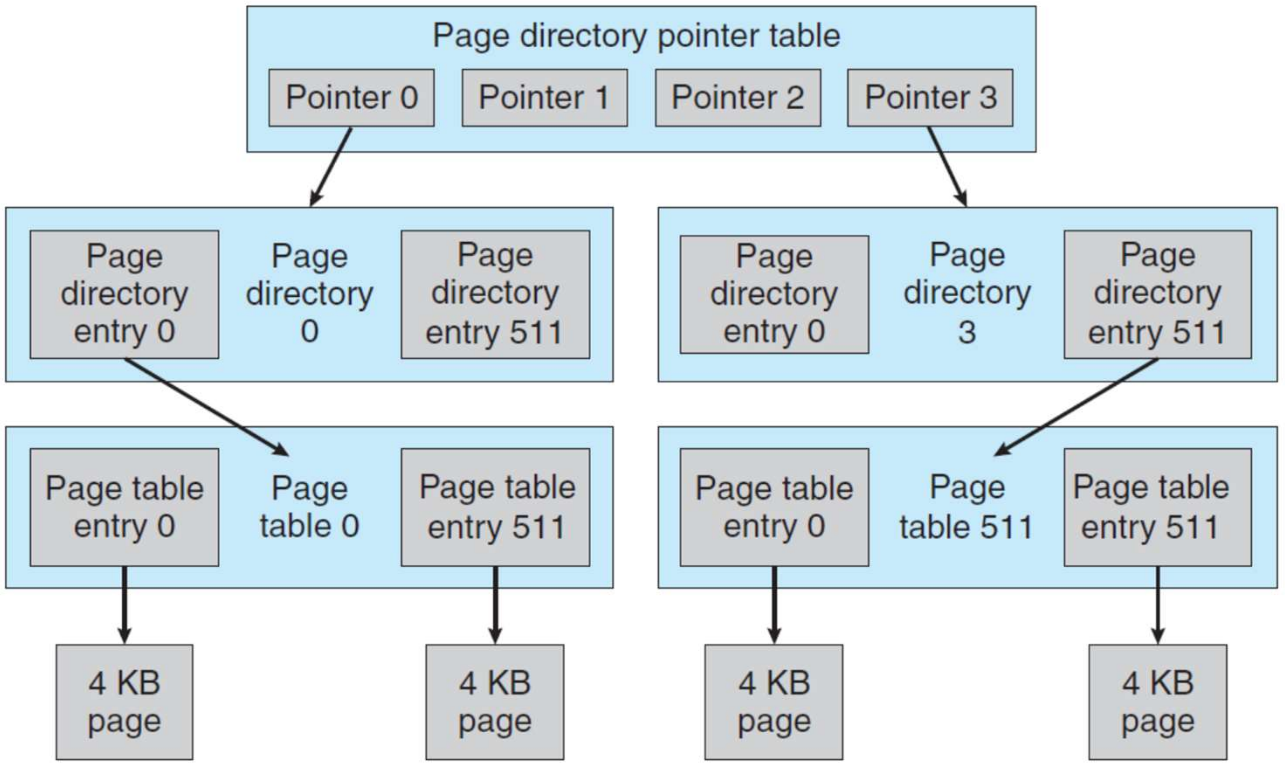


Figure 19.3 Page-table layout.

In IA-32 processors a second level PDE is needed with 4 entries

AMD64 has four such levels

Virtual-to-Physical Address Translation

- 10 bits for page directory entry, 20 bits for page table entry, and 12 bits for byte offset in page

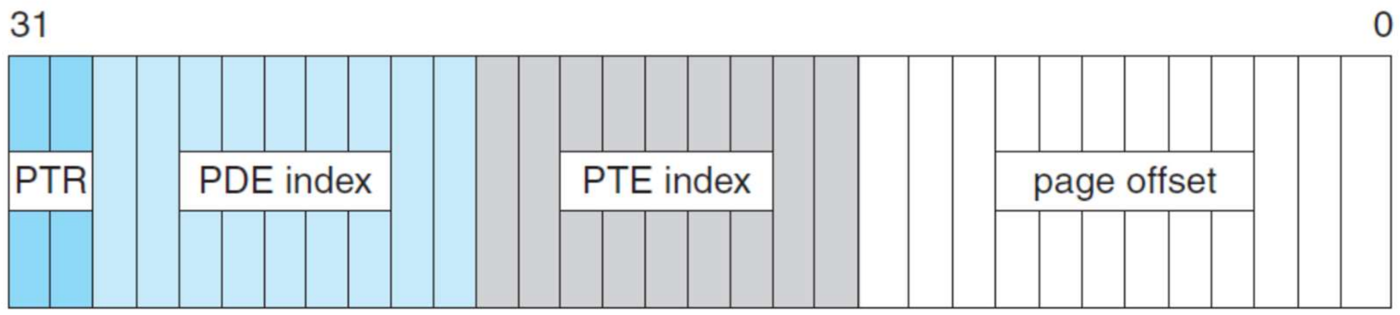


Figure 19.4 Virtual-to-physical address translation on IA-32.

Virtual Memory Manager (Cont.)

- A page can be in one of seven states: valid, zeroed, free standby, modified, transition and bad.
  - A ***free*** page is a page that has no particular content.
  - A ***zeroed*** page is a free page that has been zeroed out and is ready for immediate use to satisfy zero-on-demand faults.
  - A ***modified*** page has been written by a process and must be sent to the disk before it is allocated for another process.
  - A ***standby*** page is a copy of information already stored on disk. Standby pages may be pages that were not modified, modified pages that have already been written to the disk, or pages that were prefetched because they are expected to be used soon.
  - A ***bad*** page is unusable because a hardware error has been detected.
  - A ***transition*** page is on its way in from disk to a page frame allocated in physical memory.
  - A ***valid*** page is part of the working set of one or more processes and is contained within these processes' page tables.

Virtual Memory Manager (Cont.)

- While valid pages are contained in processes' page tables, pages in other states are kept in separate lists according to state type
- The lists are constructed by linking the corresponding entries in the **page frame number (PFN)** database, which includes an entry for each physical memory page.
- Note that the PFN database represents pages of physical memory, whereas the PTEs represent pages of virtual memory.

Virtual Memory Manager (Cont.) – Page replacement

- Windows uses a per-working-set, least-recently-used (LRU) replacement policy to take pages from processes as appropriate.
- When a process is started, it is assigned a default minimum working-set size. The working set of each process is allowed to grow until the amount of remaining physical memory starts to run low, at which point the VM manager starts to track the age of the pages in each working set.
- Eventually, when the available memory runs critically low, the VM manager trims the working set to remove older pages.
- How old a page is depends not on how long it has been in memory but on when it was last referenced. This is determined by periodically making a pass through the working set of each process and incrementing the age for pages that have not been marked in the PTE as referenced since the last pass

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.
- Eg: CreateProcess() API call

Executive — Local Procedure Call Facility

- The LPC passes requests and results between client and server processes within a single machine.
- In particular, it is used to request services from the various Windows 7 subsystems.
- When a LPC channel is created, one of three types of IPC 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 type uses APIs that read and write directly into a process's address space. ALPC provides functions and synchronization so that a server can access the data in a client. This technique is normally used by RPC to achieve higher performance for specific scenarios.

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
- Controls the Windows 7 cache manager, which handles caching for the entire I/O system
- Supports both synchronous and asynchronous operations, provides time outs for drivers, and has mechanisms for one driver to call another

Executive — Security Reference Monitor

- The object-oriented nature of Windows 7 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 and takes care of loading the appropriate drivers.

- Queries used by PnP manager:
  - add-device – request sent to appropriate driver
  - start-device – request to driver after resource assignments
  - query-stop – asks the driver if device can be temporarily disabled (eg: when device need to be reconfigured)
  - stop – stops the device from usage (temporarily)
  - query-remove – similar to query stop (eg: user gets ready to eject device)
  - surprise-remove – when device fails or user unplugs without notification
  - Remove – remove device permanently

Executive – Power Manager

- Concentrates on energy efficiency
- Sleep mode
  - For quick stop and resume (within seconds)
  - All open documents and applications are kept in RAM
  - The power is turned down low on the CPUs and I/O devices, but the memory continues to be powered enough that its contents are not lost
- Hibernate mode
  - can even write all the contents of memory to disk and turn off the power to allow the system to go into ***hibernation***
  - Zero power consumption
  - Once the computer is powered back on, it will resume everything where you left off but takes time

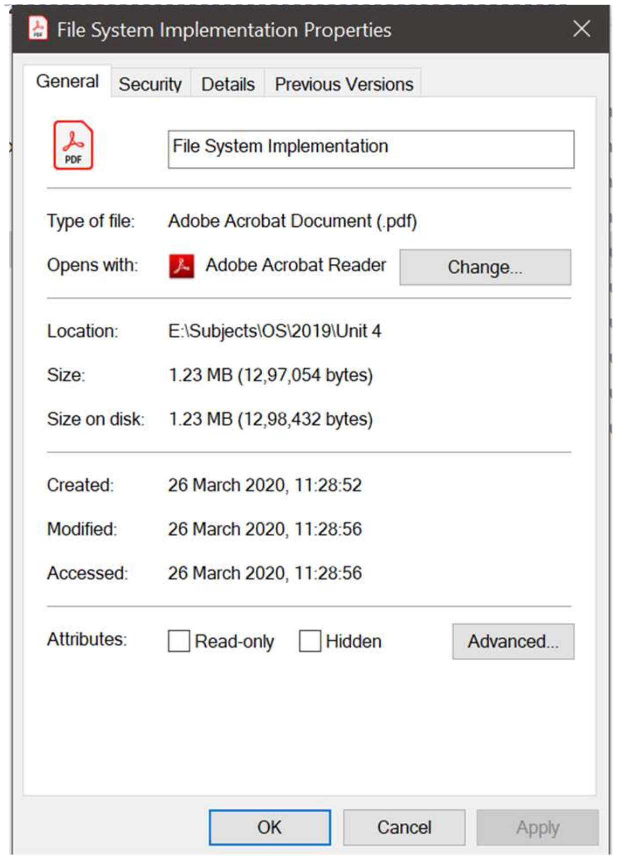
File System

File System

- The fundamental structure of the Windows 7 file system (NTFS) is a volume
  - Created by the Windows 7 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
  - The cluster size is configured when an NTFS file system is formatted.
  - The default cluster size is based on the volume size—4 KB for volumes larger than 2 GB.
  - Larger cluster size means more internal fragmentation

File System — Internal Layout

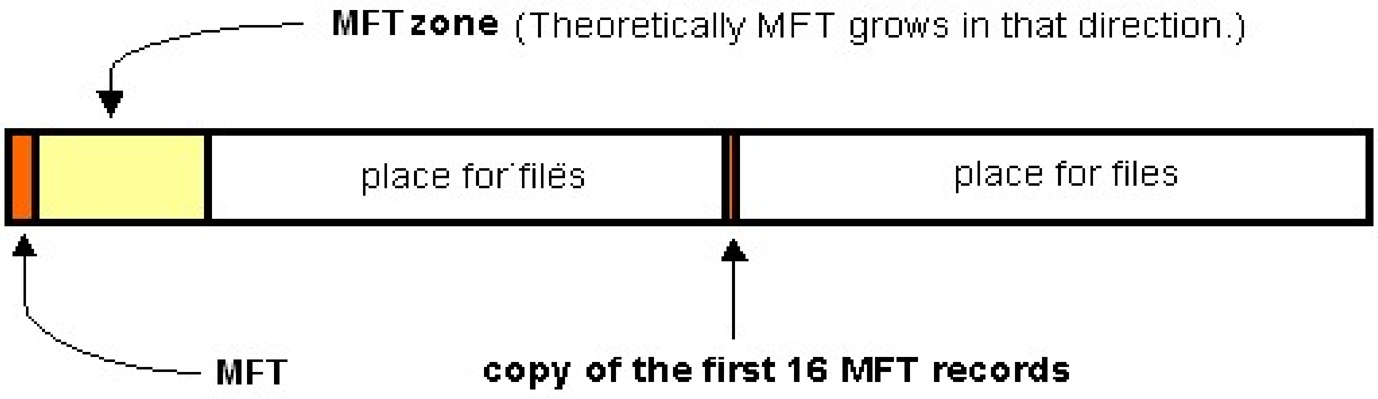
- NTFS uses logical cluster numbers (LCNs) as disk addresses
- A file in NTFS is not a simple byte stream, as in MS-DOS or UNIX, rather, it is a structured object consisting of attributes
- Each attribute of a file is an independent byte stream that can be created, deleted, read, and written.
- Some attribute types are standard for all files, including the file name, the creation time, and the security descriptor that specifies the access control list.
- User data are stored in ***data attributes***.


File System Implementation Properties

File System Implementation

Type of file:	Adobe Acrobat Document (.pdf)
Opens with:	Adobe Acrobat Reader
Location:	E:\Subjects\OS\2019\Unit 4
Size:	1.23 MB (12,97,054 bytes)
Size on disk:	1.23 MB (12,98,432 bytes)
Created:	26 March 2020, 11:28:52
Modified:	26 March 2020, 11:28:56
Accessed:	26 March 2020, 11:28:56
Attributes:	☐ Read-only ☐ Hidden

NTFS Volume structure


MFT zone (Theoretically MFT grows in that direction.)
place for files
place for files
MFT
copy of the first 16 MFT records

Because the MFT stores information about itself, NTFS reserves the first 16 records of the MFT for metadata files (approximately 16 KB), which are used to describe the MFT.

To prevent the MFT from becoming fragmented, NTFS reserves 12.5 percent of volume by default for exclusive use of the MFT.
This space, known as the MFT zone, is not used to store data unless the remainder of the volume becomes full.

Master File Table

- MFT is a database in which information about every file and directory on an NTFS volume is stored.
- There is at least one record for every file and directory on the NTFS logical volume.
- Each record contains attributes that tell the OS how to deal with the file or directory associated with the record.
- The size of a record is determined when the file system is created;
  - it ranges from 1 to 4 KB.
- Typically, each file uses one file record.
- If a file has a large number of attributes or becomes highly fragmented, it might need more than one file record. In this case, the first record for the file, the base file record, stores the location of the other file records required by the file.

Master File Table (Contd.,)

- Small attributes are stored in the MFT record itself and are called ***resident attributes***.
  - File name and date of creation are by default resident attributes
- Large attributes, such as the unnamed bulk data, are called ***nonresident attributes*** and are stored in one or more contiguous **extents** on the disk.
- A pointer to each extent is stored in the MFT record.

Master File Table (Contd.,)

- Each file on an NTFS volume has a unique ID called a file reference
- File reference is a 64-bit quantity that consists of
  - a 48-bit file number – denotes the index of the record array in the MFT
  - and a 16-bit sequence number – incremented every time the MFT entry is reused - Can be used to perform internal consistency checks such as catching a stale reference to a deleted file after the MFT entry has been reused for a new file.

NTFS B+ Tree

- The NTFS name space is organized by a hierarchy of directories
- Each directory uses a data structure called a **B+ tree** to store an index of the file names in that directory.
  - Small folder records reside entirely within the MFT structure,
  - Large folders are organized B-tree structures and have records with pointers to external clusters that contain folder entries that cannot be contained within the MFT structure
- Advantages of B+ Trees:
  - The B-tree structure allows NTFS to group, or index, similar file names and then search only the group that contains the file, minimizing the number of disk accesses needed to find a particular file, especially for large folders.
  - Because of the B-tree structure, NTFS outperforms FAT for large folders because FAT must scan all file names in a large folder before listing all of the files.

NTFS Metadata

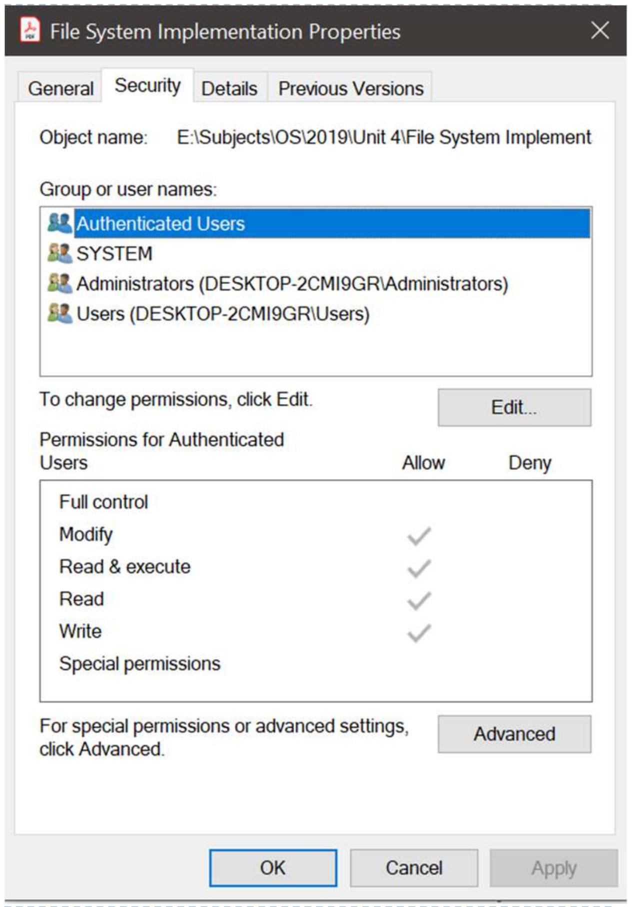
- The NTFS volume's metadata are all stored in files
- **Log file** records all metadata updates to the file system.
- **Volume file** contains the name of the volume, the version of NTFS that formatted the volume, and a bit that tells whether the volume may have been corrupted and needs to be checked for consistency using the chkdsk program.
- **Attribute-definition table** indicates which attribute types are used in the volume and what operations can be performed on each of them.
- **Root directory** is the top-level directory in the file-system hierarchy.
- **Bitmap file** indicates which clusters on a volume are allocated to files and which are free.
- **Boot file** contains the startup code for Windows and must be located at a particular disk address so that it can be found easily by a simple ROM bootstrap loader. The boot file also contains the physical address of the MFT.
- **Bad-cluster file** keeps track of any bad areas on the volume; NTFS uses this record for error recovery.

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.
  - Periodically (usually every 5 seconds), a checkpoint record is written to the log. The system does not need log records prior to the checkpoint to recover from a crash. They can be discarded, so the log file does not grow without bounds.
  - This scheme does not guarantee that all the user-file contents are correct after a crash

File System — Security

- 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.


File System Implementation Properties

Object name: E:\Subjects\OS\2019\Unit 4\File System Implement

Group or user names:

- Authenticated Users
- SYSTEM
- Administrators (DESKTOP-2CMI9GR\Administrators)
- Users (DESKTOP-2CMI9GR\Users)

To change permissions, click Edit.

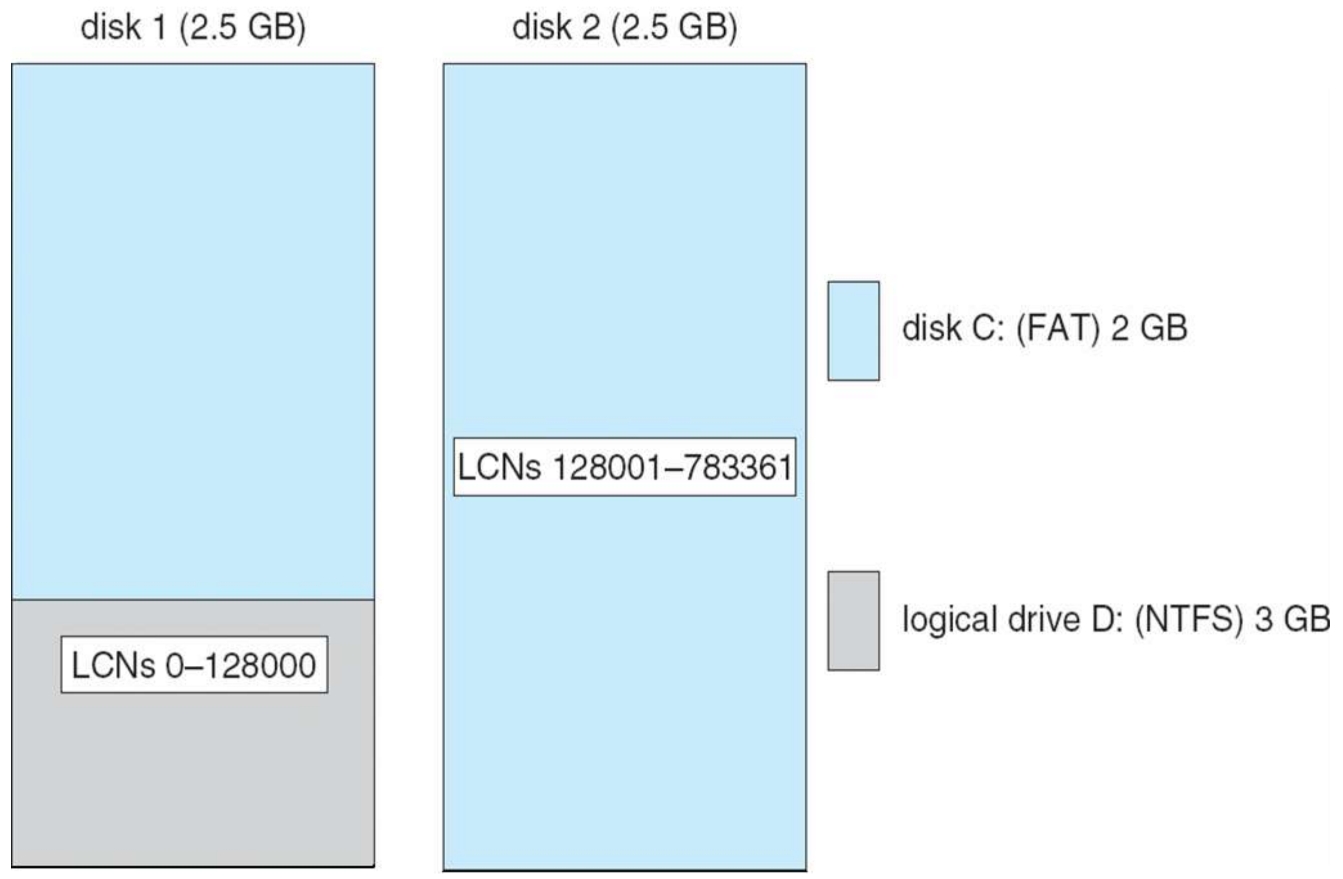
Permissions for Authenticated Users	Allow	Deny
Full control		
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.

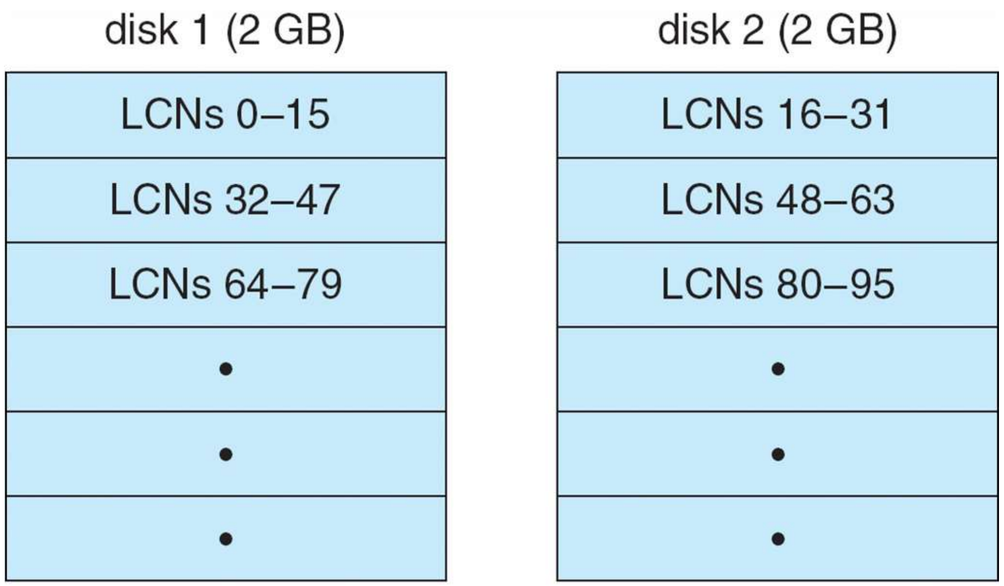
Volume Management and Fault Tolerance

- When installed, 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 two disks with identical data contents

Volume Set On Two Drives



Stripe Set on Two Drives



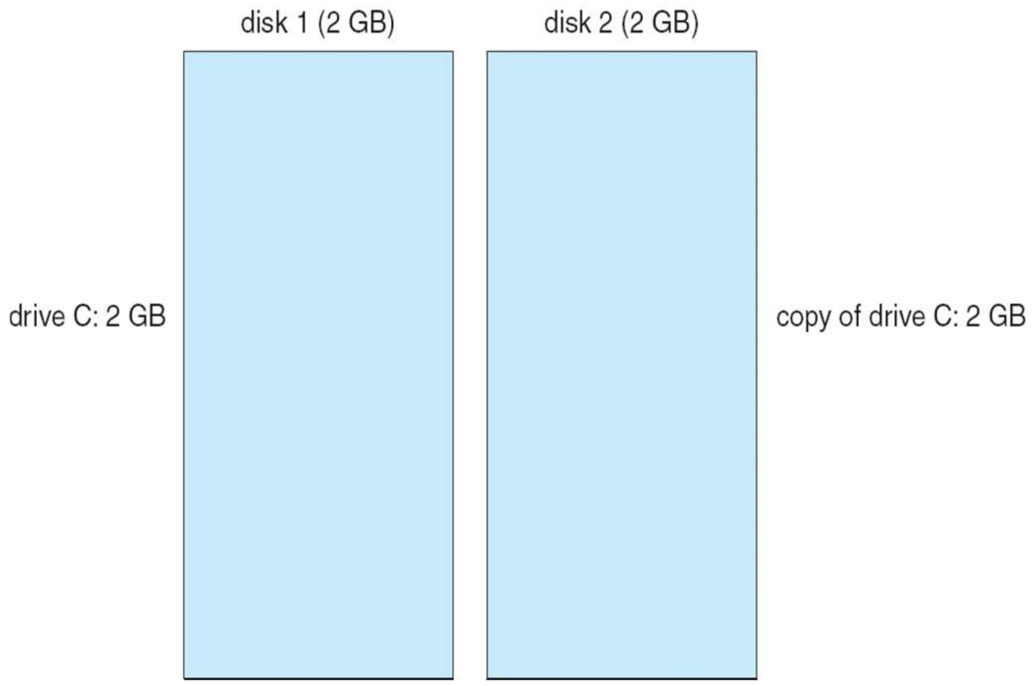
FtDisk uses a stripe size of 64 KB.

First 64 KB of the logical volume are stored in the first physical partition, the second 64 KB in the second physical partition, and so on, until each partition has contributed 64 KB of space.

Then, the allocation wraps around to the first disk, allocating the second 64-KB block.

A stripe set forms one large logical volume, but the physical layout can improve the I/O bandwidth, because for a large I/O, all the disks can transfer data in parallel.

Mirror Set on Two Drives



Bad blocks

- 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
- **Sector Sparing**
  - When a disk drive is formatted, it creates a map from logical block numbers to good sectors on the disk.
  - It also leaves extra sectors unmapped, as spares. If a sector fails, FtDisk instructs the disk drive to substitute a spare.
- **Cluster remapping**
  - is a software technique performed by the file system. If a disk block goes bad, NTFS substitutes a different, unallocated block by changing any affected pointers in the MFT.
  - NTFS also makes a note that the bad block should never be allocated to any file

File System — Compression

- NTFS can perform data compression on individual files or on all data files in a directory.
  - To compress a file, NTFS divides the file's data into compression units, which are blocks of 16 contiguous clusters.
  - When a compression unit is written, a data-compression algorithm is applied.
  - If the result fits into fewer than 16 clusters, the compressed version is stored.
- For sparse files or files that contain mostly zeros, 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.

Other services

- Symbolic links and Hard links
- Change Journal
  - NTFS keeps a journal describing all changes that have been made to the file system.
  - User-mode services can receive notifications of changes to the journal and then identify what files have changed by reading from the journal
  - Eg: The file-replication service uses it to identify files that need to be replicated across the network
- Volume Shadow Copies
  - Snapshot of a volume in its consistent state
  - Eg: The server version of Windows uses shadow copies to efficiently maintain old versions of files stored on file servers. The user can use this feature to recover files that were accidentally deleted or simply to look at a previous version of the file, all without pulling out backup media.

Summary

- Windows – Design principles
- Architecture – Layered with subsystems
- System components – Kernel, I/O Manager, PnP manager, Object Manager, VM Manager etc.,
- File System
  - NTFS Structure
  - Fault Tolerance and reliability
  - Compression

End of Chapter 19