Chapter 1. Introduction

Contents

- What Operating Systems Do?
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments
- Open-Source Operating Systems

Objectives

- To provide a grand tour of the major operating systems components
- To describe the basic organization of computer systems.

1.1 What Operating Systems do?

- What is an Operating System ?
  - A program that acts as an intermediary between a user/application of a computer and the computer hardware.
    - manage the computer hardware/resources
    - controls the execution of applications
    - provide a basis for application programs.
- Operating system goals:
  - **Convenience**: OS provide an environment in which make the computer system convenient to use
  - **Efficiency**: Use the computer hardware in an efficient manner.
  - **Ability to evolve**: Permit the effective development, testing, and introduction of new system functions without interfering with service

Convenience	or	**Efficiency**	or	**Both**
PC		Mainframe		

Computer System Structure

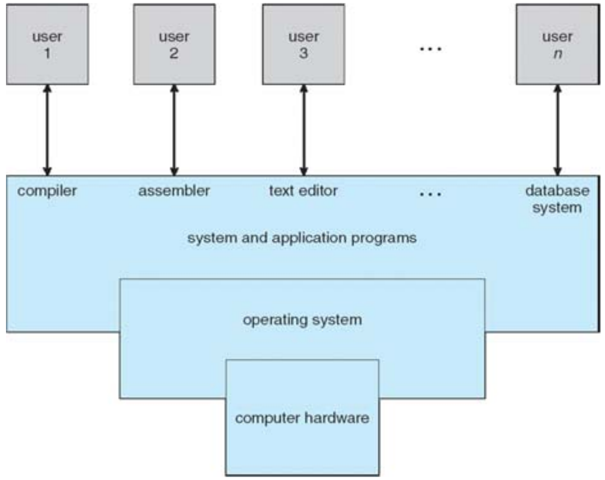
- Computer system can be divided into four components

1. Hardware
   - provides basic computing resources

     (ex) CPU, memory, I/O devices
2. Operating system
   - controls and coordinates the use of the hardware among the various application programs for the various users.
3. Applications programs
   - Define the ways in which the system resources are used to solve the computing problems of the users

     (ex) word processors, compilers, web browsers, database systems, video games, spreadsheet ...
4. Users
   - people, machines, other computers

Four Components of a Computer System



Operating System Definitions

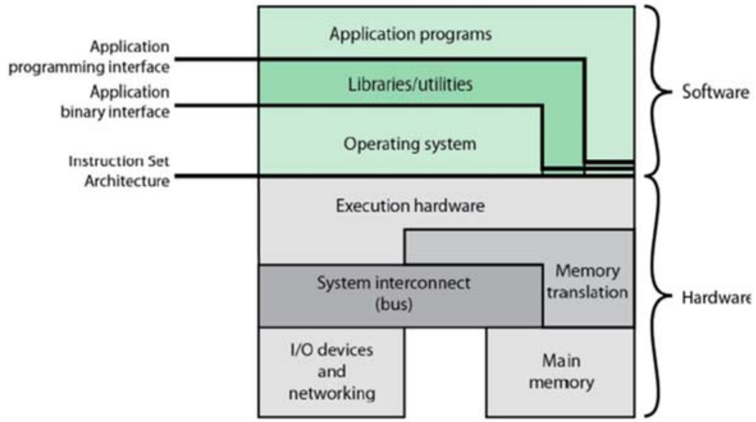
- An operating system provides **an *environment*** within which other programs can do useful work.

  "An operating system is similar to a **government**."
- User View
  - varies according to the ***interface*** being used

  (ex) PC, mainframe/minicomputer, servers, mobile computer, embedded computers
    - PC → ease of use, performance (some) , resource utilization (no)
- Systems View
  - OS is a **resource allocator**
    - manages and allocates resources

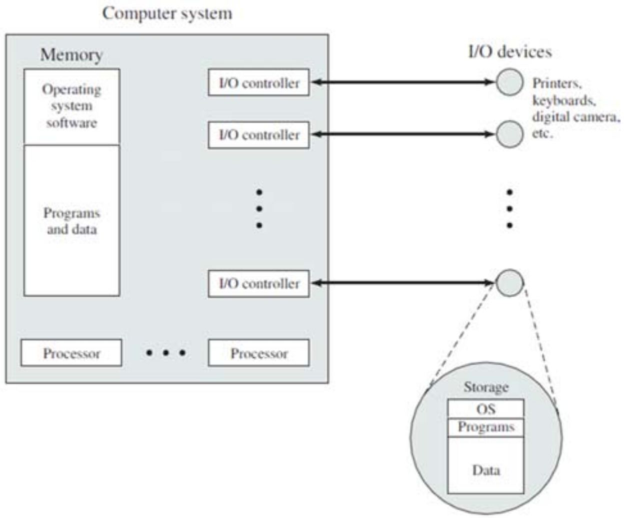
    (resources: CPU time, memory space, file storage space, I/O device)
  - OS is a **control program**
    - controls the execution of user programs and operations of I/O devices .

Operating System as a User/Computer Interface



API (Application Programming Interface)
ABI (Application Binary Interface)
ISA (Instruction Set Architecture)

Operating System as Resource Allocator/Controller



Operating System Definitions (cont')

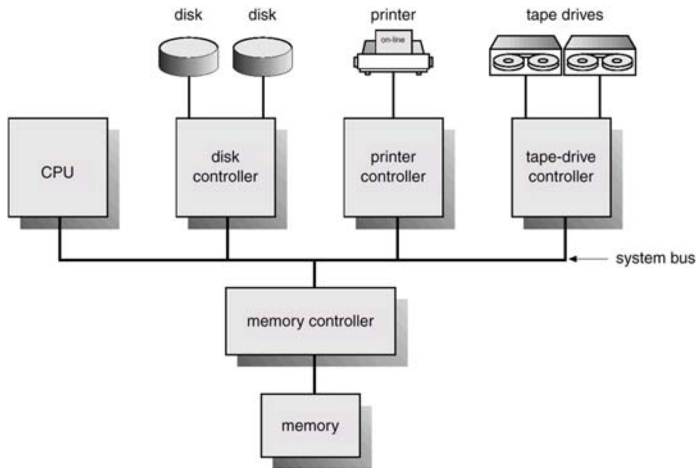
- Defining Operating System
  - **No** universally accepted definition
- "*Everything* a vendor ships when you order an operating system" is good approximation. But varies greatly across systems
  - **Microsoft case** : US Department of Justice filed suit against Microsoft claiming that Microsoft included too much functionality in its OS. (ex: web browser) → guilty of monopoly to limit competition
- More common definition – **kernel**
  - **Kernel**: the one program running <u>at all times</u>"
- Two other types of programs
  - **a system program**: program associated with the OS, but are not part of kernel.
  - **an application program**: program not associated with the operation of the system

Middleware

- Mobile operating systems often include not only <u>core kernel</u> but also <u>middleware</u>
- Middleware
  - a set of software frameworks that provide additional services to application developers
    (ex) Apple iOS, Google Android

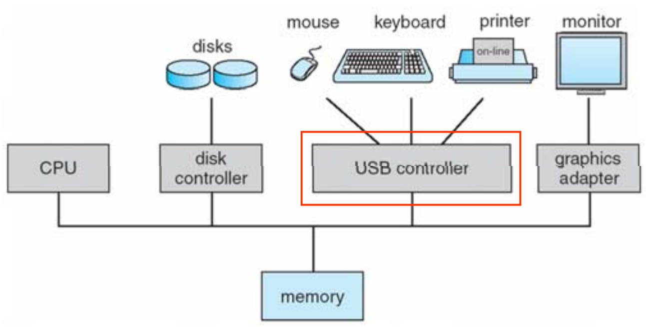
1.2 Computer-System Operation

- Computer-System Architecture (Old-fashion)



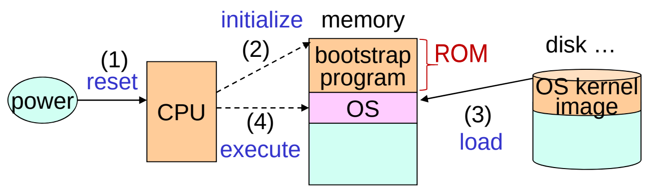
Modern Computer System

- Computer-System Architecture (Modern)
  - **one or more CPU**, **device controllers** connect through **common bus** providing access to **shared memory.**
  - CPUs and devices can execute concurrently, competing for memory cycles



1.2.1 Computer System Operation

- Bootstrap program – computer startup program
  - initial program to run when the computer is powered up or reboot
  - typically stored in ROM or EEPROM, generally known as **firmware**
- Operation of bootstrap program
  - initialize all aspects of system (*ex*: CPU register, device controller, memory contents ...) and
  - locate and load into memory the operating system kernel and start executing the first process (*ex*: init)



Interrupts

- Interrupt
  - a signal to the CPU emitted by hardware or software indicating an event that requires immediate attention.
- Interrupt types

	hardware interrupt (interrupt)	by an external I/O device at any time
by software (exception, or trap)	Internal interrupt (trap, exception)	by an execution error (divide by zero, invalid memory access…)
by software (exception, or trap)	software interrupt (system call)	by a software request for OS service - special instruction

  - 넓은 의미의 interrupt: all kinds of interrupts (이 때에는 trap/exception도 같은 의미로 사용)
  - 좁은 의미의 interrupt: hardware interrupt

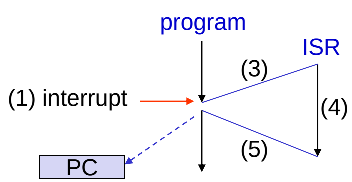
Relationship between OS and Interrupt

- An operating system is interrupt driven.
  - An Operating System performs appropriate interrupt handling action.
  - “no interrupt, no work”
- Example

interrupts	**OS's works**
hardware interrupt	→ I/O handling, timer handling
internal interrupt (exception)	→ error handling
software interrupt (system call)	→ provide OS services to applications

Interrupt Sequence

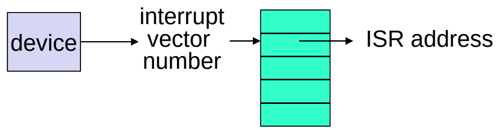
- Interrupt Sequence
  - When the CPU is <u>interrupt</u>, it *stops* what it is doing and <u>save the address</u> of the interrupted instruction.
  - Interrupts transfers control to the <u>interrupt service routine (ISR)</u>
  - After the interrupt is serviced, the control is transferred <u>to the saved return address</u> → *resume* the interrupted computation.



(2) save return address
(where ? → <u>fixed location</u> or <u>system stack</u>)

Determining the address of ISR

- Polled interrupt
  - invoke a <u>general polling routine</u> (at a fixed location) to examine the interrupt information
  - then, the routine calls the <u>interrupt-specific</u> handler by polling devices
- Vectored interrupt
  - a unique number is given with the interrupt request
  - The address of the interrupt service routines is provided through the <u>interrupt vector</u> (array of addresses), indexed by a given unique number .
  - quick interrupt handling

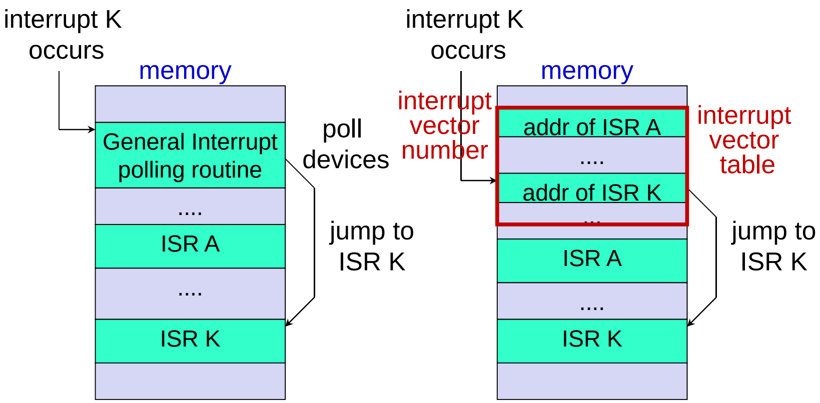


interrupt vector table

Polled Interrupt vs. Vectored Interrupt

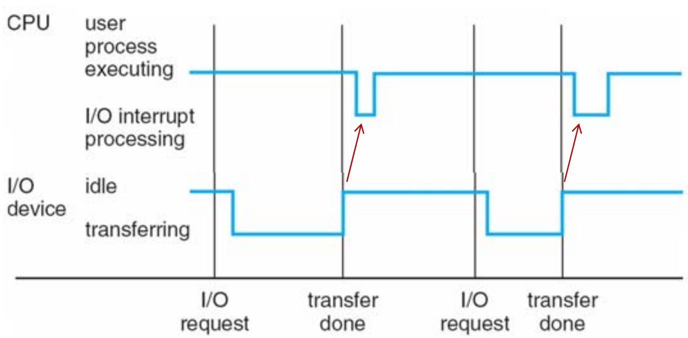
Polled Interrupt

Vectored Interrupt

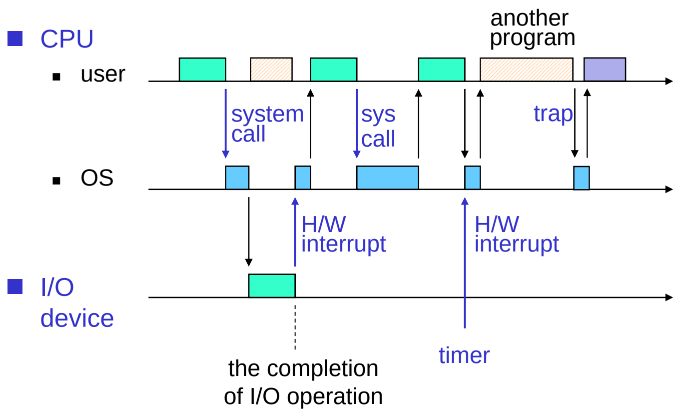


Interrupt Time Line

- I/O devices and the CPU can execute <u>concurrently</u>
- I/O device usually interrupts CPU when I/O transfer is done.



Interrupt Time Line (detail)



1.2.2 Storage Structure

- Main memory
  - the CPU can access main memory directly ⇒ memory address
  - the CPU can load instruction only from memory
    → programs must be in main memory to be executed.
    (**von Neumann** architecture - both programs and data are stored in main memory)
  - DRAM : the most common main memory
  - cannot reside programs/data in main memory permanently for two reasons : (1) too small (2) volatile storage
- Secondary storage
  - extension of main memory that provides large nonvolatile storage capacity.
  - Magnetic disks: the most common secondary storage
  - The disk controller determines the logical interaction between the device and the computer.

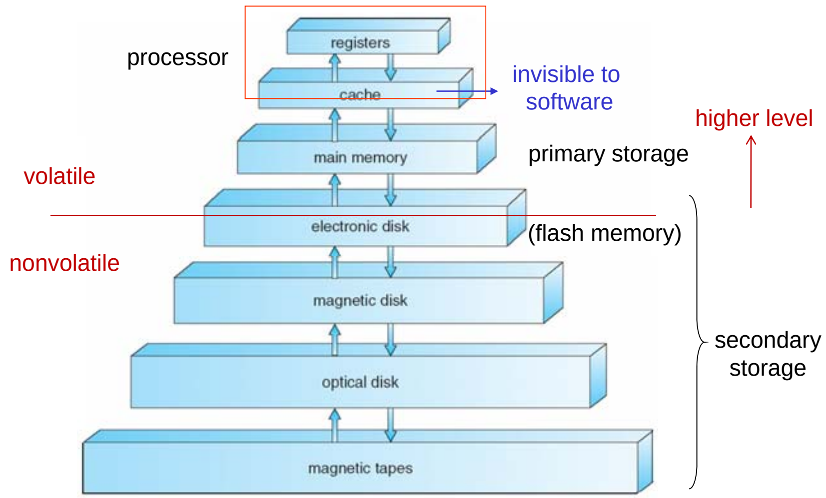
Storage Hierarchy

- Storage systems organized in hierarchy.

	Higher level	Lower level
Speed:	fast	slow
Cost:	expensive	inexpensive
Volatility:	volatile	nonvolatile

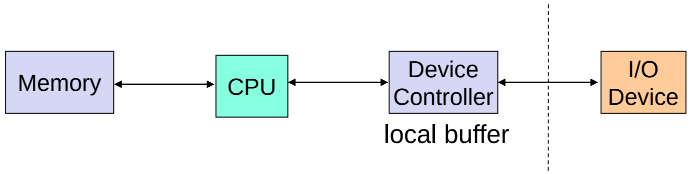
- Caching
  - copying information into faster storage system on a temporary basis
  - Examples
    - cache memory : invisible to software
    - main memory can be viewed as a cache for secondary storage

Storage-Device Hierarchy



1.2.3 I/O Structure

- Data transfer between I/O device and the CPU is done through a device controller.
- Device controller
  - Each **device controller** is in charge of a specific type of device
  - Depending on the controller, more than one device may be attached (ex) SCSI controller
  - A device controller has a local buffer and a set of special-purpose registers.
  - The device controller is responsible for moving the data between the peripheral devices and its local buffer storage



Device Driver

- I/O data transfer
  - Main memory ⇔ Local buffer (by CPU)
  - Local buffer ⇔ the device (by device controller)
- A large portion of OS code is dedicated to managing I/O
  - because of its importance to the reliability and performance
  - because of varying nature the devices
- Device driver
  - Typically, operating systems have a **device drivers** for each device controller.
  - understands the device controller and presents a uniform interface to the device to the rest of the OS

I/O operations

- To start an I/O operation,
  - the device driver loads the appropriate registers within the device controller.
  - the device controller examines the registers to determine what action to take.
  - the controller start the transfer
- Three modes of I/O operations
  - Programmed I/O (Polling)
  - Interrupt-driven I/O
  - Direct memory access(DMA) based I/O

Programmed I/O (Polling)

- CPU waits I/O transfer completion.
  - CPU tests I/O transfer completion by repeatedly reading status information.
- Once the transfer is complete, CPU performs transfer of data between device controller and memory
- CPU cannot execute any other jobs while an I/O operation is in progress

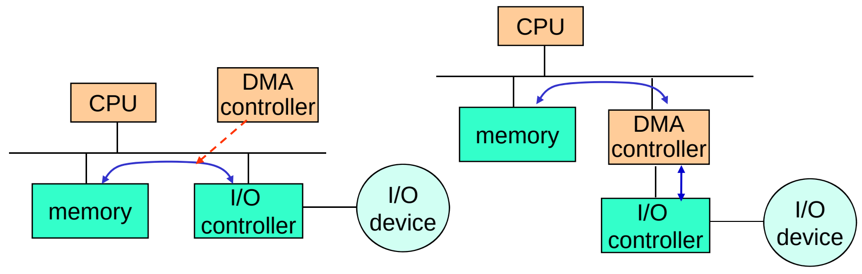
  ➔ solution : **Interrupt-driven I/O**

Interrupt-driven I/O operation

- CPU can execute any other jobs after starting I/O operation.
- Once the transfer is complete,
  - the device controller informs the device driver <u>via an **interrupt**</u>
  - then, CPU performs transfer of data between device controller and memory.
- Interrupt-driven I/O is fine for moving <u>small</u> amounts of data, but can produce high overhead for <u>bulk data movement</u> such as disk I/O
  - ➔ solution: **DMA (direct memory access)**

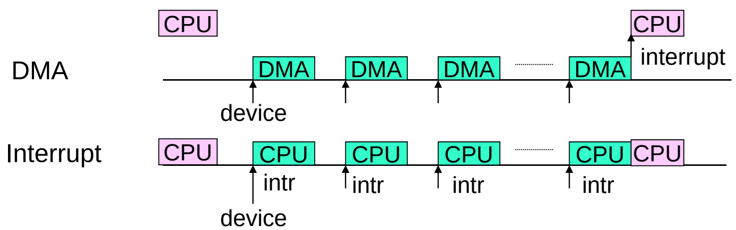
Direct Memory Access (DMA)

- Direct Memory Access(DMA)
  - DMA controller transfers <u>directly</u> a block of data <u>between memory and the buffer in the device controller</u> without CPU intervention
  - able to transmit information at close to memory speeds.
    ➔ used for high-speed I/O devices
  - CPU initiates I/O operation, but is not involved in data transfer.



DMA operation

- Operation
  - <u>CPU</u> sets up buffers, <u>pointers</u>, and <u>counters</u> for I/O device
  - DMA controller performs the <u>data transfer by DMA</u>
  - DMA controller <u>interrupts</u> the CPU when the block transfer has been completed.
- Comparison between DMA and Interrupt I/O
  - DMA: one interrupt per block
  - Interrupt-driven I/O: one interrupt per byte.

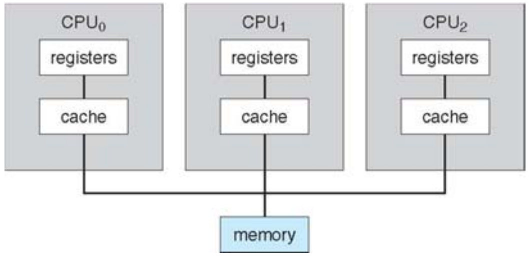


1.3 Computer-System Architecture

- Single-processor systems
- Multiprocessor systems
  - also known as <u>parallel system</u>
  - have more than one CPU in close communication
    → called "<u>tightly coupled system</u>"
- Advantages of multiprocessor systems
  - Increased throughput
  - Economy of scale
    - because of sharing peripherals, mass storage
  - Increased reliability
    - graceful degradation
    - fault tolerant systems

Symmetric vs. Asymmetric Multiprocessing

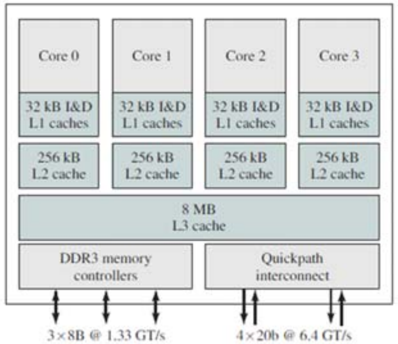
- Symmetric multiprocessing (SMP) vs. Asymmetric multiprocessing
  - **SMP** : each processor performs all tasks within the OS



  - **asymmetric multiprocessing**: each processor is assigned a specific task. A master processor controls the system

Multi-core Processor and Blade server

- Multi-core processor on a single chip
  - multi-core CPUs appear to the OS as N standard processors
- Blade servers
  - multiple processor boards, I/O boards, and networking boards are placed in the same chassis.
  - each blade processor board boots independently and runs its own OS

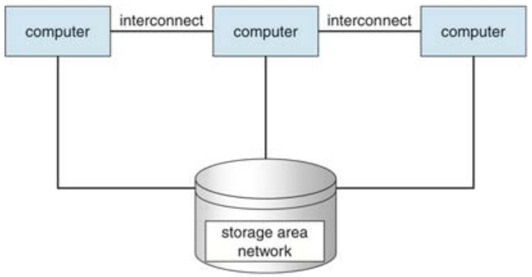


Intel Core i7 quad-core

blade server

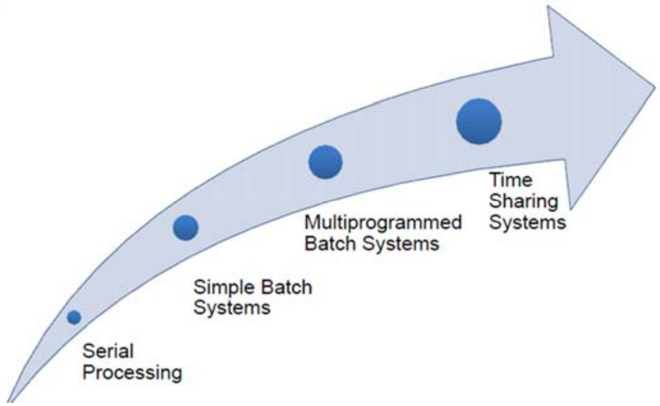
Clustered Systems

- Like multiprocessor systems, **clustered systems** gather together multiple systems to accomplish computational work.
  - linked via a LAN or a faster interconnect (ex. InfiniBand).
  - Provides a high-availability service which survives failures
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization
  - Usually sharing storage via a **storage-area network (SAN)**



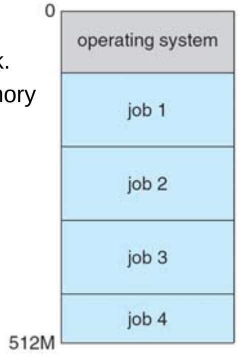
1.4 Operating-System Structure

- Evolution of Operating Systems

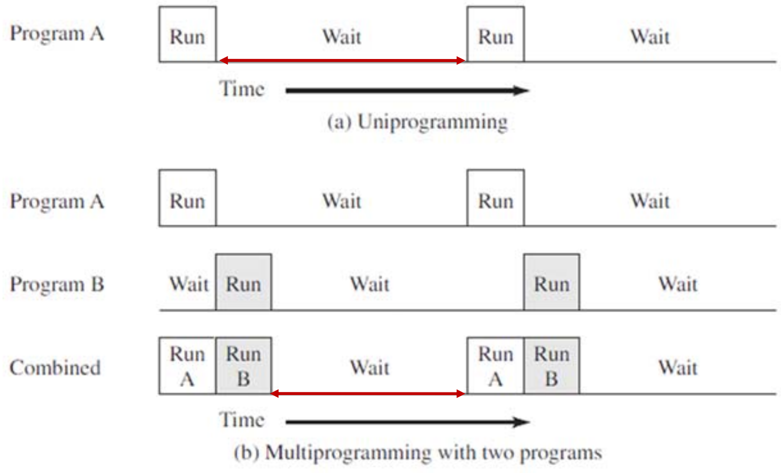


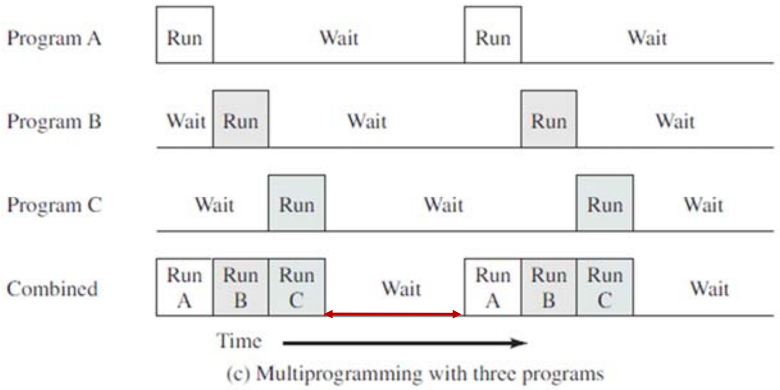
1.4 Operating-System Structure

- Multiprogramming – needed for efficiency
  - **Multiprogramming** : Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.
  - Single user cannot keep CPU and I/O devices busy at all times
    → Multiprogramming increase CPU utilization by keeping CPU and I/O busy at all times
  - **job pool** : the jobs are kept initially on the disk.
  - A subset of total jobs in system is kept in memory
  - one job of them is selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job



Uniprogramming and Multiprogramming





Operating-System Structure (cont')

- Time-Sharing (or Multitasking)
  - a logical extension of multiprogramming
  - The CPU is multiplexed among multiple jobs so frequently that users can interact with each job while it is running.
    → **interactive computer system**
    - The **response time** should be short (< 1sec)
- In time shared operating system,
  - each user has at least one separate program in memory ➔ process
  - if several jobs are ready to run at the same time ➔ CPU scheduling
  - If processes do not fit in the memory, they are swapped in and out of main memory to the disk ➔ swapping
  - virtual memory allows execution of processes not completely in memory

Batch Multiprogramming vs. Time Shaing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

1.5 Operating-System Operations

- Operating systems are <u>interrupt driven</u>
  - I/O requests → hardware interrupt
  - software error → internal interrupt (exception, trap)
    - division by zero or invalid memory access ...
  - OS service request → software interrupt (system call)
    - request from a user program for operating system service

  $\}$ execute OS program

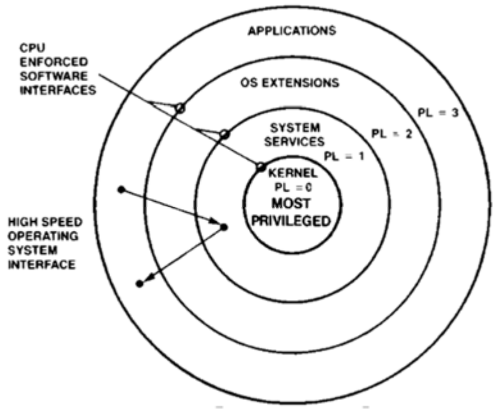
- With sharing, many processes could be affected by a bug in a program.
  - infinite loop
  - modify another process or the operating system

- <u>Sharing</u> system resources requires <u>**protection**</u>
  - protect operating system and all other programs from any <u>malfunctioning</u> program to ensure proper operation

Dual-Mode Operation

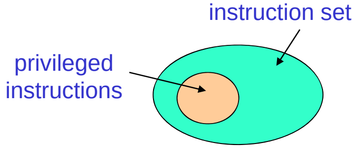
- Dual mode operation
  - most CPU provide hardware support to differentiate between <u>at least two modes of operations</u>.
    1. **User mode**
    2. **Kernel mode**
       (also called supervisor, monitor, system, privileged mode)
  - **CPU *mode* bit** indicates the current mode.
    - for example, kernel (0) or user (1)
- Dual-mode operation allows OS to protect itself and other system components
  - distinguish between the execution of OS code and user-defined code
- Multimode operation
  - the concepts of modes can be extended beyond two modes
  - CPUs that support virtualization frequently have a separate mode for virtualization

(ex) 4-level Protection of Intel 386



Privileged Instruction

- Privileged Instruction
  - the hardware allows privileged instructions to be executed in only kernel mode.
  - An attempt to execute a privileged instruction in user mode causes an exception.
    → **privilege violation**



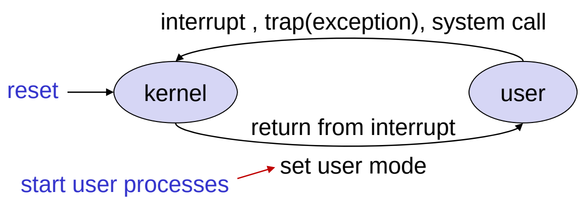
- Examples of privileged instructions
  - I/O instructions
  - timer management, interrupt management, MMU register management
  - system control instructions: HALT, Enable/Disable interrupt,
    Switch to kernel mode …

User Mode and Kernel Mode

- User Mode
  - user program execute in user mode.
  - privileged instructions cannot be executed.
  - certain areas of memory are protected from user access
- Kernel Mode
  - OS (monitor) executes in kernel mode
  - privileged instructions may be executed
  - protected areas of memory may be accessed
  - switch into kernel mode when an interrupt occurs

Transition from User to Kernel mode

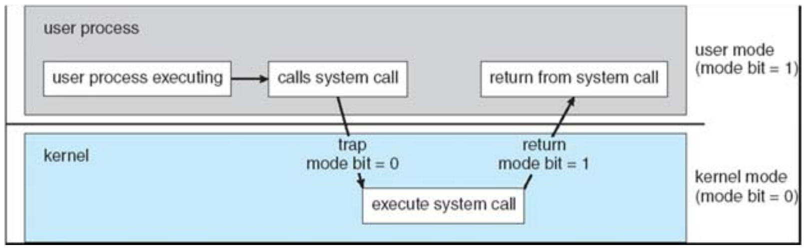
- Transition of CPU operation mode
  - At power on, start in kernel mode
  - The OS is loaded and start user processes in user mode
  - **When an interrupt occurs, switches to kernel mode.**
  - when return from interrupt, switches to user mode (original mode).



- The lack of a hardware-supported dual mode
  - can cause serious shortcomings in an operating system.

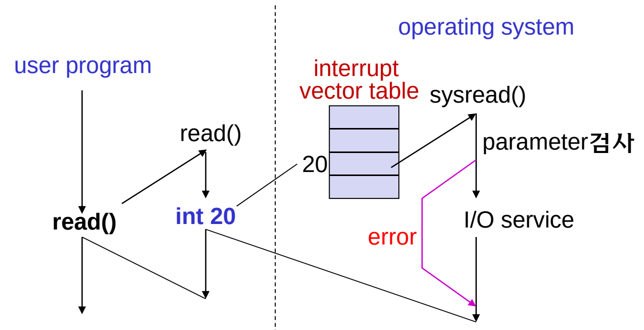
  (ex) 8088 architecture and MS-DOS : no dual mode

Transition from User to Kernel mode (cont')



- System call sequence
  - Invoke system call (by INT, trap, or syscall instruction)
  - Control passes to a service routine in the OS, and the mode bit is set to kernel mode.
  - The service routine verifies that the parameters are correct and legal, and executes the request
  - returns control to the instruction following the system call.

System Call Sequence



(**int**는 IA32의 software interrupt 명령어)

user mode **kernel mode**

Timer and CPU Protection

- CPU protection
  - use timer to prevent infinite loop / process hogging resources
- Timer Interrupt
  - A timer interrupts the CPU after specified period (ex: 1/60 sec)
  - OS sets the timer (counter).
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- The use of timer
  - to implement time sharing.
  - to compute the current time.

1.6 Process Management

- A **process** is a program in execution.
  - It is a unit of work within the system.
  - **Program** is a passive entity, **process** is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Typically system has many processes running concurrently on one or more CPUs – some user processes, some OS
  - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

1.7 Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines *what* is in memory *when*
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

1.8 Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit → **file**
  - Each medium is controlled by storage device(ex: disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management - one of most visible components of OS
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - File-system management activities :
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store
  - data that does not fit in main memory, or
  - data that must be kept for a “long” period of time.
- Entire speed of computer operation hinges on disk subsystem and its algorithms because disk is used frequently.
- Proper management is of central importance
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling

Caching

- Caching - copying information into faster storage system on a temporary basis
- Important principle, performed at many levels in a computer
  - cache memory
  - main memory can be viewed as a cache for secondary storage
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache is smaller than storage being cached
  - cache management is important
  - selection of the cache size and a replacement policy

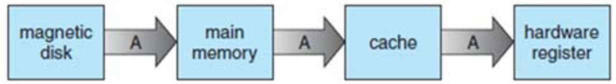
Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Coherency and Consistency
  - the same data may appear in different level of the storage hierarchy ➔ data in different level must be consistent.
    (ex) cache coherency
  - In multitasking environment, multiprocessor environment, distributed environment, the situation becomes more complex

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including
    - **buffering** – storing data temporarily while it is being transferred
    - **caching** – storing parts of data in faster storage for performance
    - **spooling** – storing output for a device that cannot accept interleaved data stream
  - General device-driver interface
  - Drivers for specific hardware devices

1.9 Protection and Security

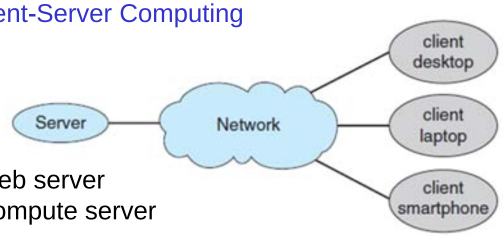
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - user ID, group ID ...

1.11 Computing Environments

- Traditional Computing
  - batch
    → interactive, time-sharing by multi-users
    → interactive, time-sharing by single user
  - Blurring over time
  - Office environment
    - PCs connected to a network,
      terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

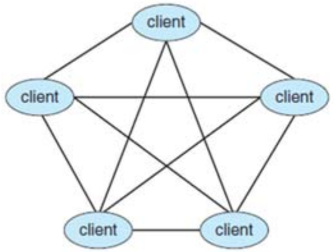
- Mobile Computing
  - computing on mobile devices such as smartphones and tablet computers
  - mobile devices use wireless or cellular data network
  - applications that take advantage of the unique features of mobile devices, such as GPS and accelerometers ..
    → augmented-reality application
- Distributed Systems
  - Distribute the computation among several physical processors.
  - Each processor has its own local memory; processors communicate with one another through communications lines
  - network : LAN, WAN, MAN, PAN ...
  - network operating system vs. distributed operating systems

- Client-Server Computing

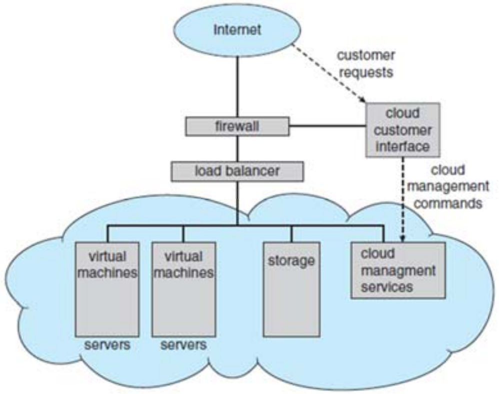


- web server
- compute server
- file server

- Peer-to-Peer Computing
  - clients and servers are not distinguished from one another



- Cloud Computing
  - a type of computing that delivers computing, storage, and even applications as a service across a network.



Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to the copy protection and Digital Rights Management (DRM) movement
- Started by Free Software Foundation (FSF), which has "copyleft" GNU Public License (GPL)
- Examples include
  - GNU/Linux - hundreds of distributions
  - BSD UNIX (including core of Mac OS X) - FreeBSD, OpenBSD,Darwin
  - Sun (now Oracle) Solaris - OpenSolaris