Introduction to Operating Systems

Dr. Tarunpreet Bhatia
Assistant Professor
CSED, TIET

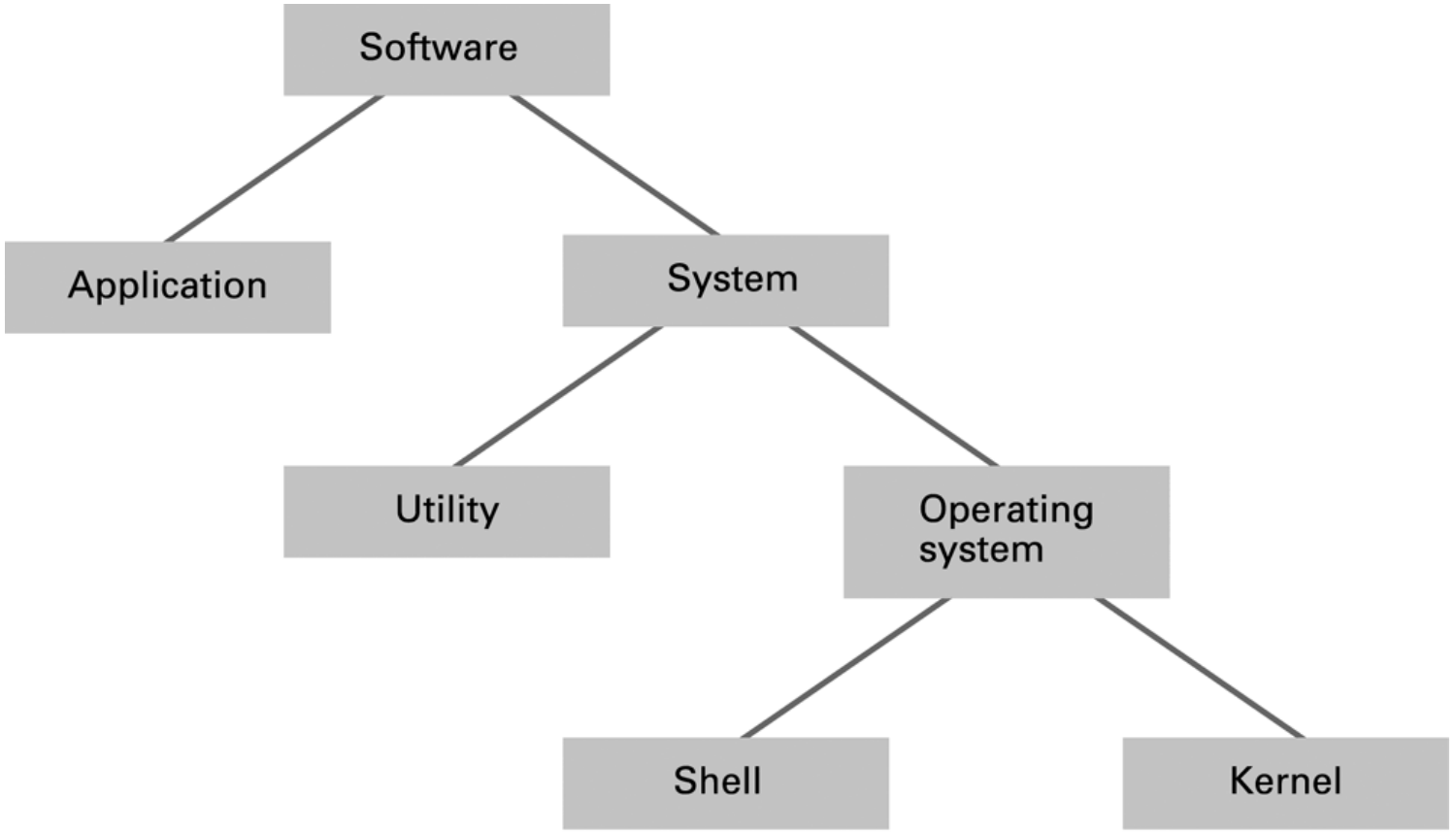
Disclaimer

This is NOT A COPYRIGHT MATERIAL

Content has been taken mainly from the following books:

Operating Systems Concepts By Silberschatz & Galvin
Operating Systems: Internals and Design Principles By William Stallings

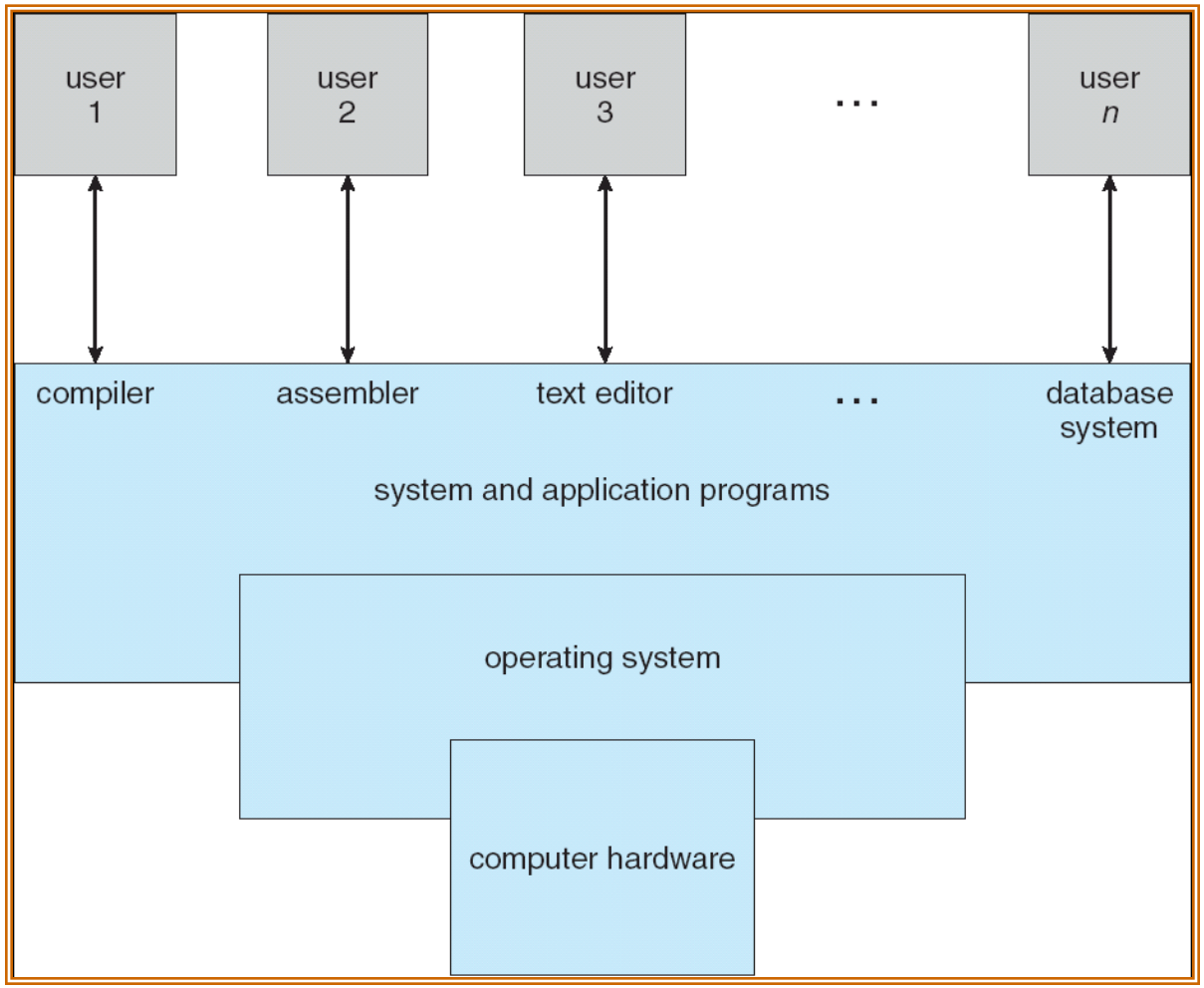
Software classification



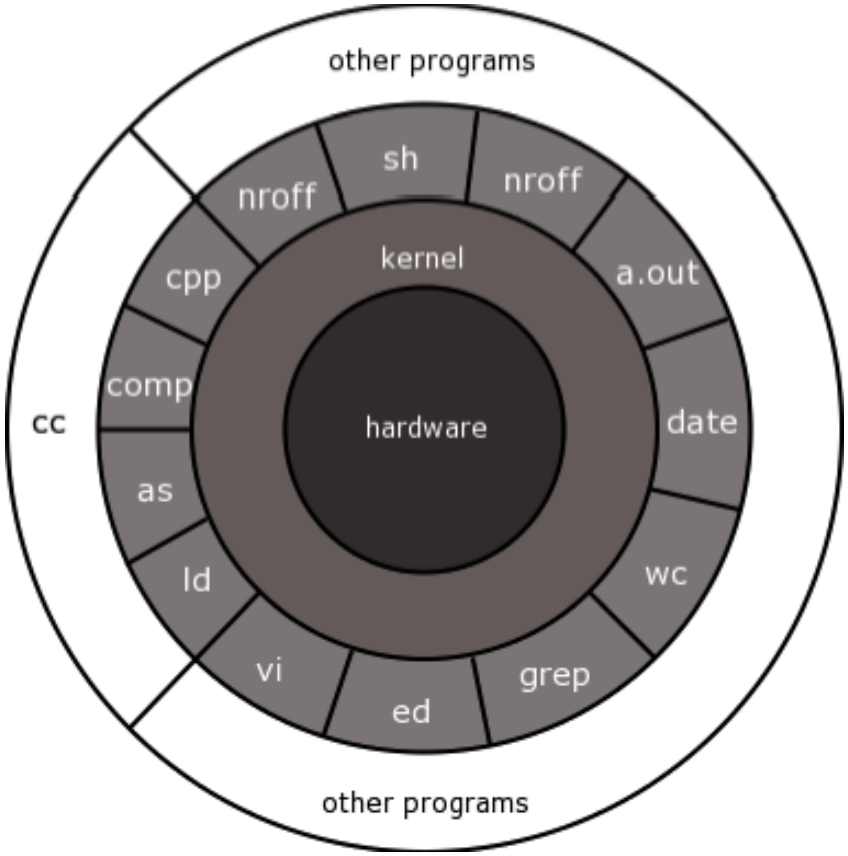
Operating System

- A Program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
  - Use the computer hardware in an efficient manner.

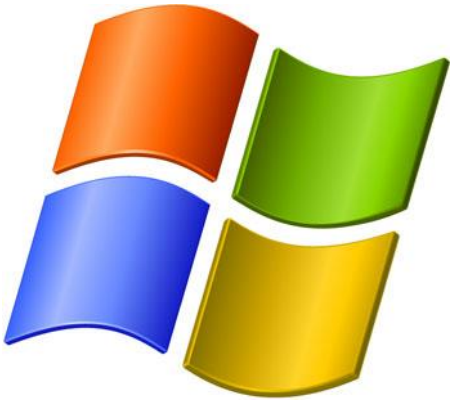
Abstract View



Linux Architecture



Various OS








If operating systems could talk, this is what they would say:

Windows - "Do whatever you want, BUT stay within these boundaries, and use what tools I give you."

Apple/Mac - "Do only what I allow you to do, in the way that I make it possible. Don't try to be smart. You aren't."

Linux - "Do whatever you want with whatever you need to make it work. You will have the minimal tools available, and you have to look for it, otherwise, someone else will have the appropriate tool for you. Use them. If you can make it work (for it is possible), congratulations."

Activity

Compare and contrast various operating systems

- DoS, Unix, Linux, Windows
- Mobile OSs
- Various versions of Windows: Windows NT, XP, Windows Server 2008, 7, 8, 10, 11 etc.
- Different shells available in Linux

What Operating Systems Do?

Depends on the point of view:

- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

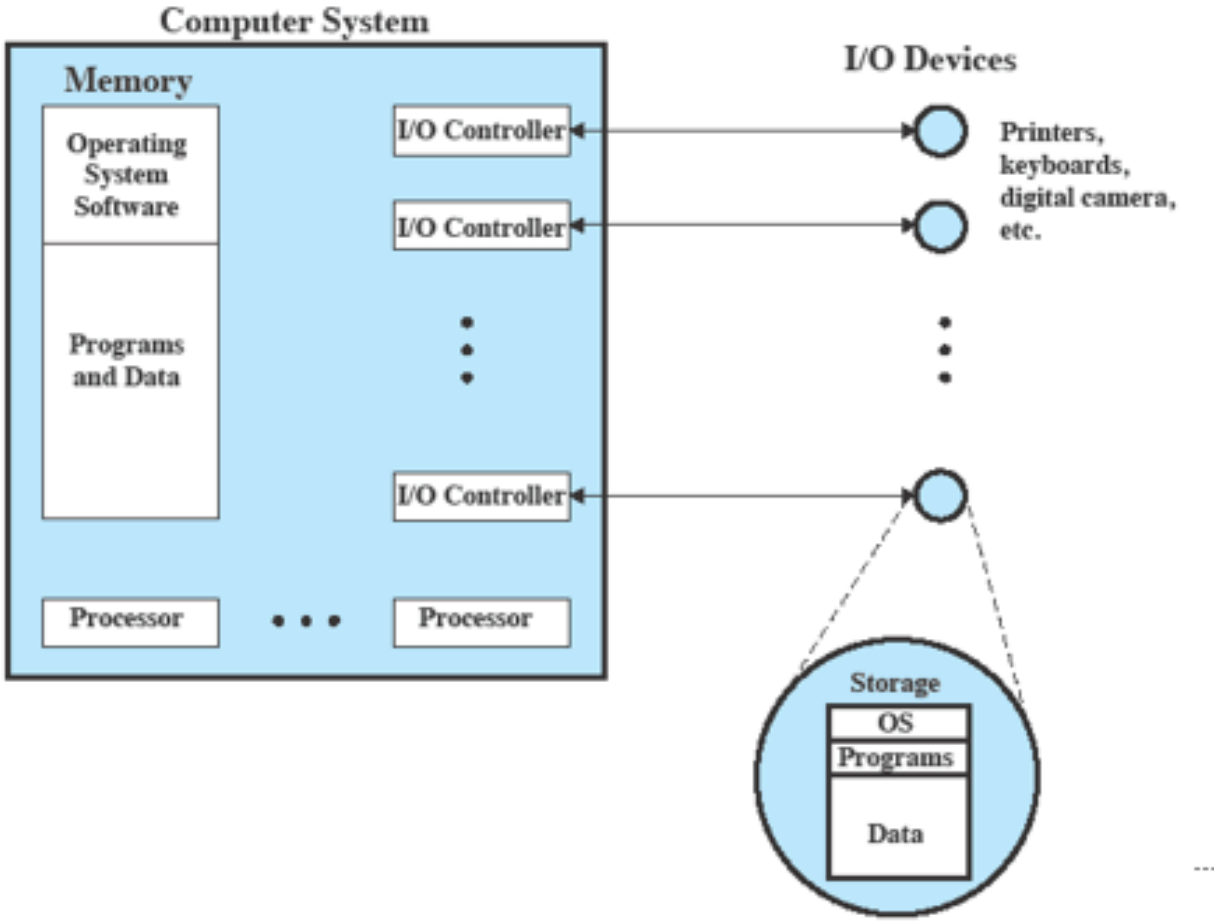
Goals of OS

- Convenience
- Efficiency
- Ability to evolve

OS as a User/Computer Interface

- Program development
- Program execution
- Access to I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

OS as Resource Manager



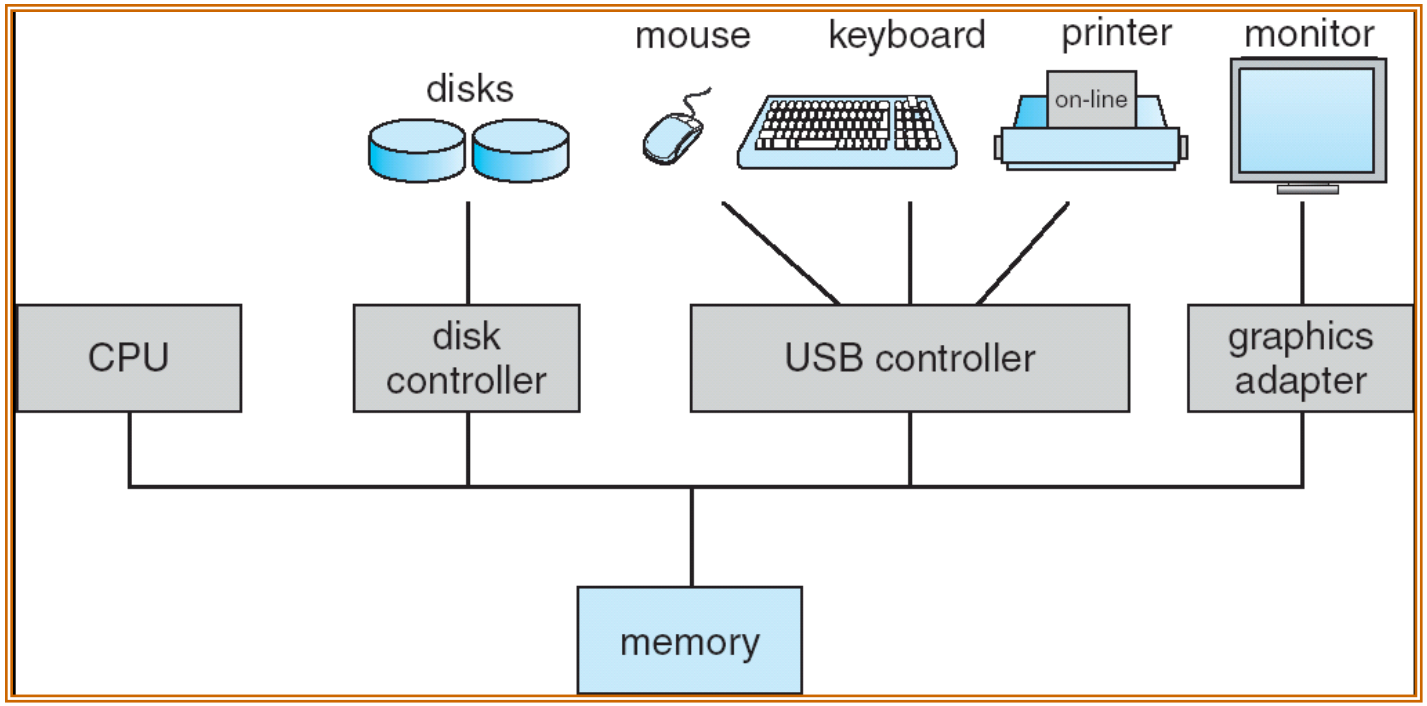
Stallings, William, Operating Systems Internals and Design Principles, Prentice Hall

Ease of Evolution of an OS

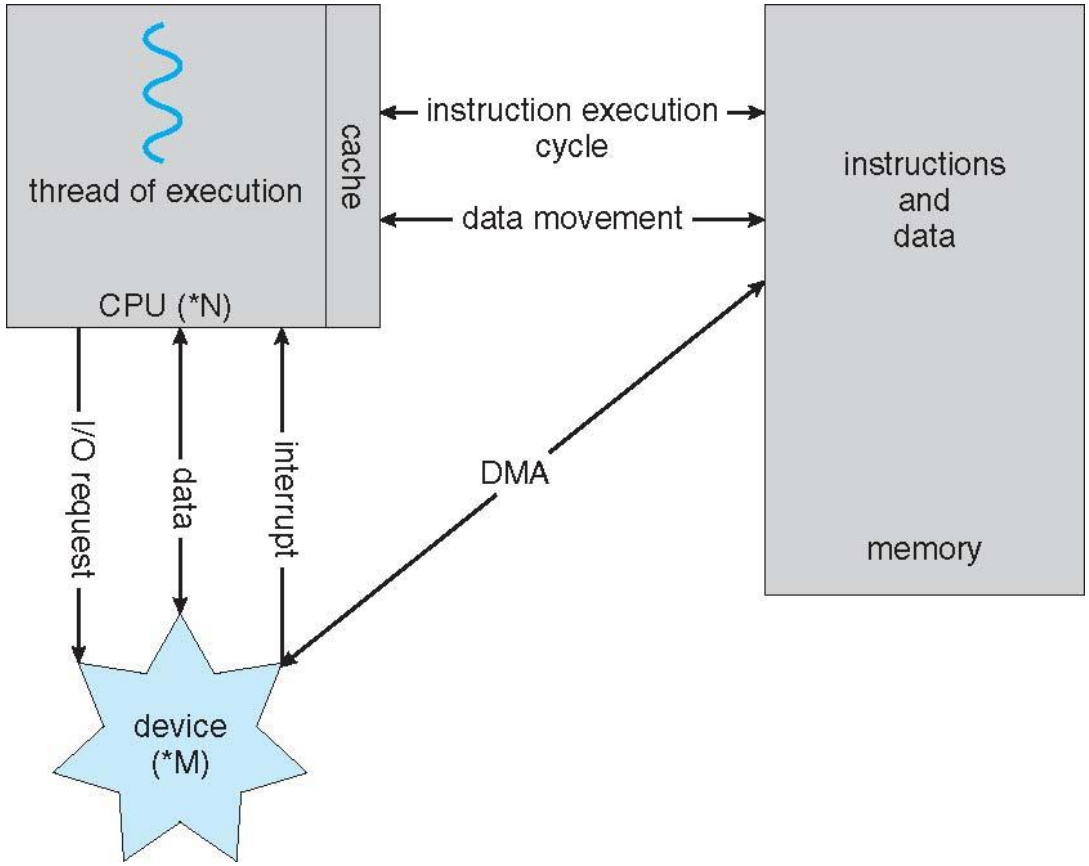
A major OS will evolve over time for a number of reasons:

- Hardware upgrades plus new types of hardware
- New services
- Fixes

Computer System Organization



How a Modern Computer Works



A von Neumann architecture

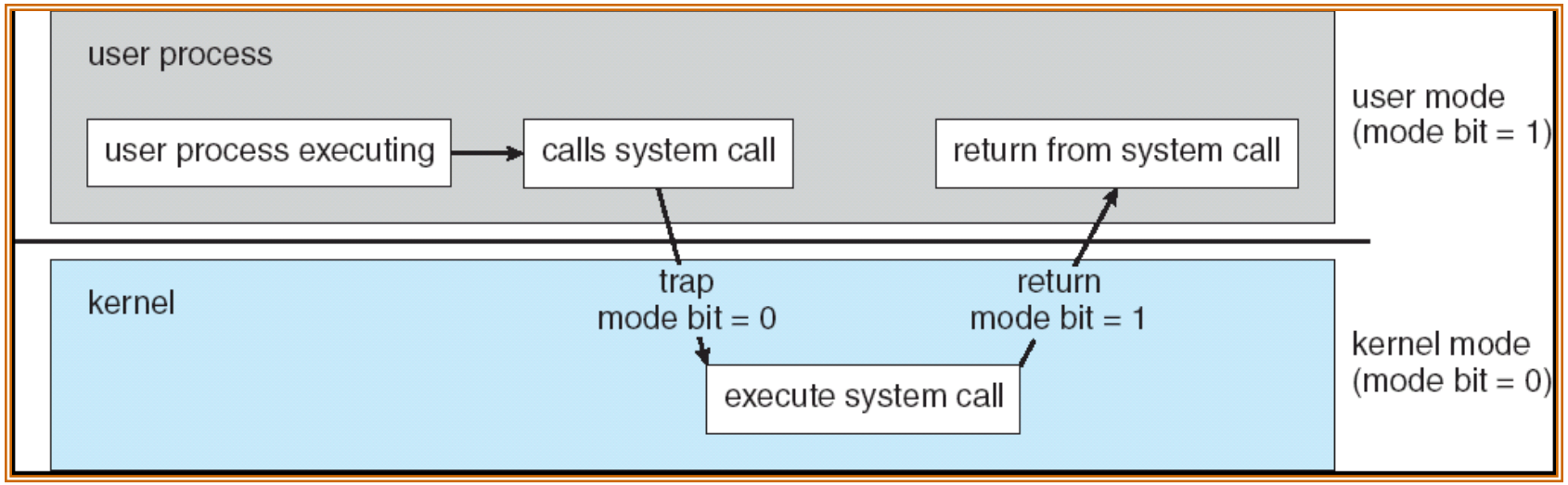
Computer-System Operation

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt.

OS Operations- Dual mode and Multimode

- Dual-mode operation allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode



Timer

- Timer to Prevent Infinite Loop / Process hogging resources

  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

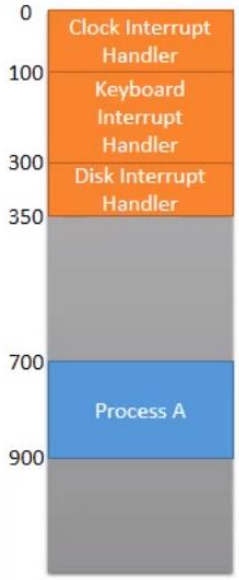
Which of the following instructions should be privileged?

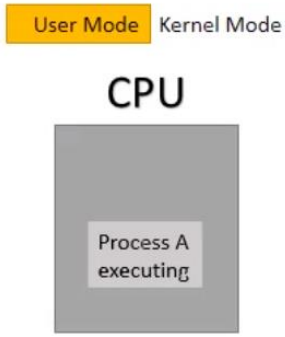
a. Set value of timer
b. Read the clock
c. Clear memory
d. Issue a trap instruction
e. Turn off interrupts
f. Modify entries in device-status table
g. Access I/O device

Interrupt Handling

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- An operating system is **interrupt driven**
- The operating system preserves the state of the CPU by storing registers and the program counter
- Separate segments of code determine what action should be taken for each type of interrupt

Main Memory

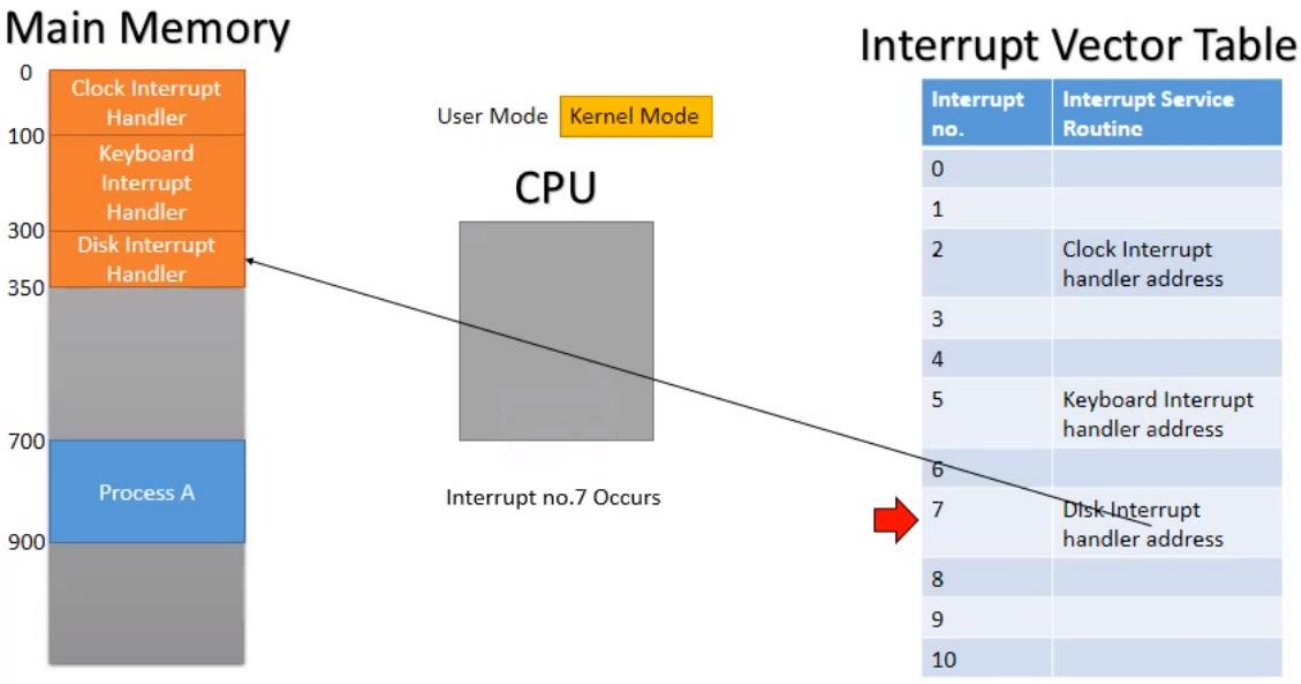


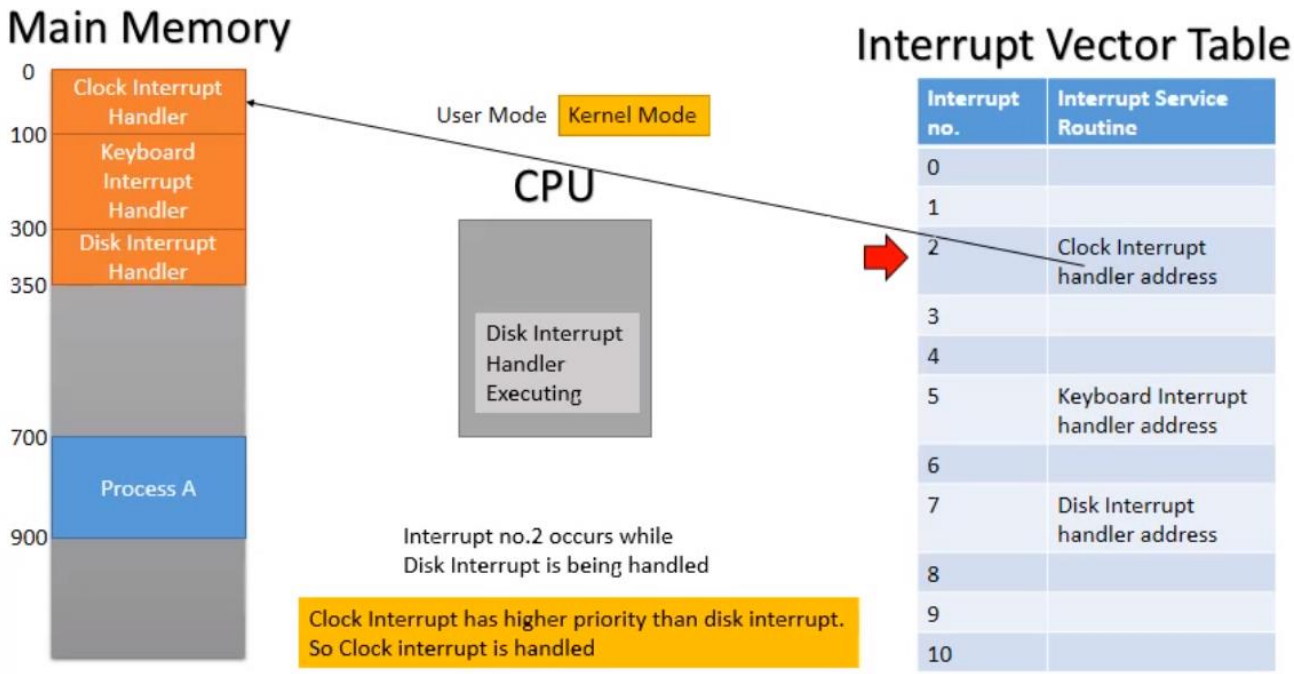


Interrupt no.7 Occurs

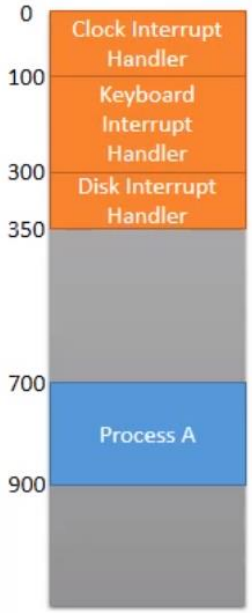
Interrupt Vector Table

Interrupt no.	Interrupt Service Routine
0	
1	
2	Clock Interrupt handler address
3	
4	
5	Keyboard Interrupt handler address
6	
7	Disk Interrupt handler address
8	
9	
10	


Main Memory
0
Clock Interrupt Handler
100
Keyboard Interrupt Handler
300
Disk Interrupt Handler
350
700
Process A
900
User Mode
Kernel Mode
CPU
Interrupt no.7 Occurs
Interrupt Vector Table
Interrupt no.
Interrupt Service Routine
0
1
2
Clock Interrupt handler address
3
4
5
Keyboard Interrupt handler address
6
7
Disk Interrupt handler address
8
9
10


Main Memory
0
Clock Interrupt Handler
100
Keyboard Interrupt Handler
300
Disk Interrupt Handler
350
700
Process A
900
User Mode
Kernel Mode
CPU
Disk Interrupt Handler Executing
Interrupt no.2 occurs while Disk Interrupt is being handled
Clock Interrupt has higher priority than disk interrupt. So Clock interrupt is handled
Interrupt Vector Table
Interrupt no.
Interrupt Service Routine
0
1
2
Clock Interrupt handler address
3
4
5
Keyboard Interrupt handler address
6
7
Disk Interrupt handler address
8
9
10

Main Memory



User Mode | Kernel Mode

CPU

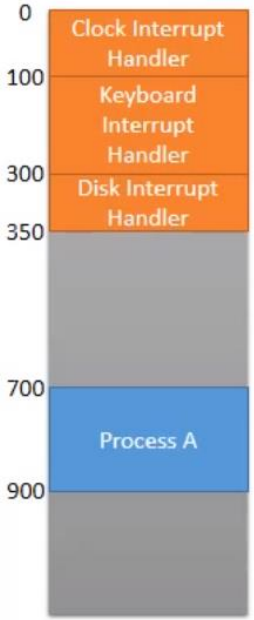
Disk Interrupt Handler Executing

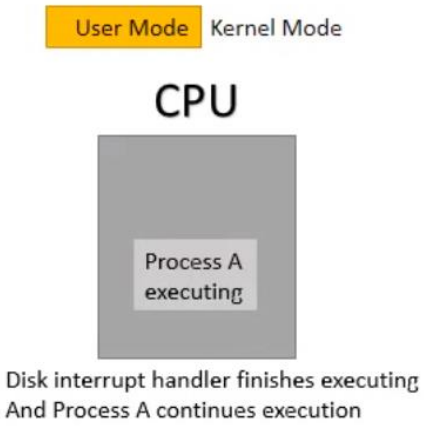
Clock Interrupt handler finishes executing and disk interrupt handler continues

Interrupt Vector Table

Interrupt no.	Interrupt Service Routine
0	
1	
2	Clock Interrupt handler address
3	
4	
5	Keyboard Interrupt handler address
6	
7	Disk Interrupt handler address
8	
9	
10	

Main Memory





Disk interrupt handler finishes executing
And Process A continues execution

Interrupt Vector Table

Interrupt no.	Interrupt Service Routine
0	
1	
2	Clock Interrupt handler address
3	
4	
5	Keyboard Interrupt handler address
6	
7	Disk Interrupt handler address
8	
9	
10	

MCQ

A computer handles several interrupt sources of which the following are relevant for this question.

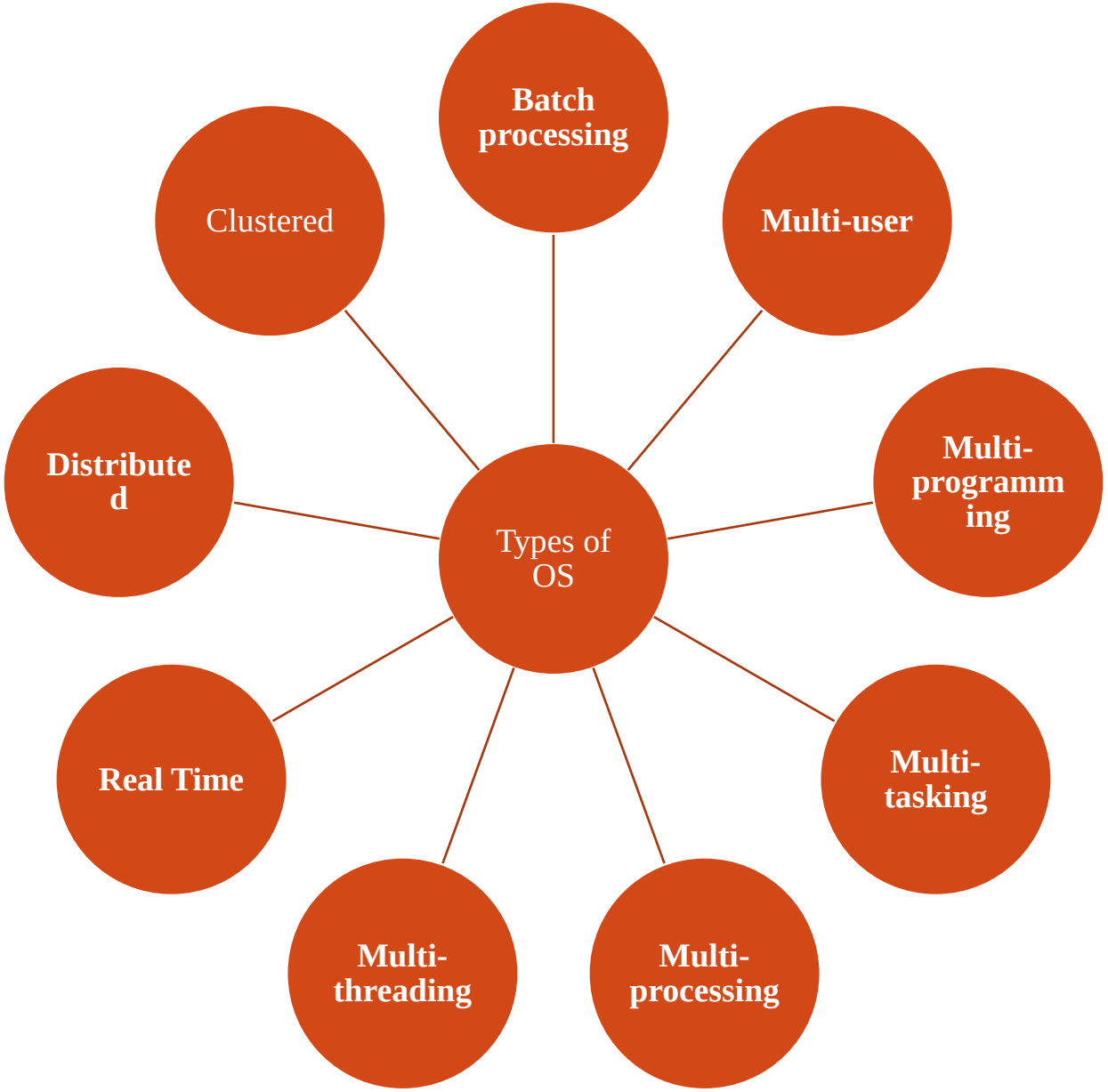
1. Interrupt from CPU temperature sensor (raises interrupt if CPU temperature is too high)

2. Interrupt from Mouse(raises interrupt if the mouse is moved or a button is pressed)

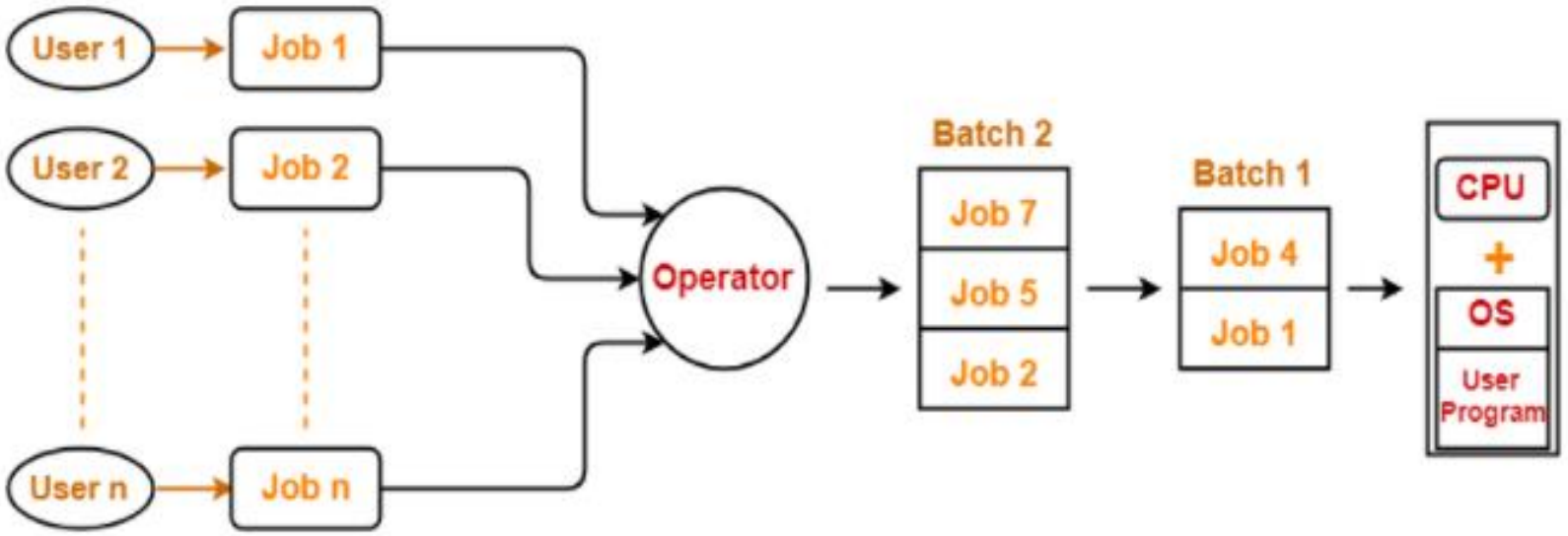
3. Interrupt from Keyboard(raises interrupt when a key is pressed or released)

4. Interrupt from Hard Disk(raises interrupt when a disk read is completed)

Which one of these will be handled at the HIGHEST priority?


Types of OS
Batch processing
Multi-user
Multi-programming
Multi-tasking
Multi-processing
Multi-threading
Real Time
Distributed
Clustered

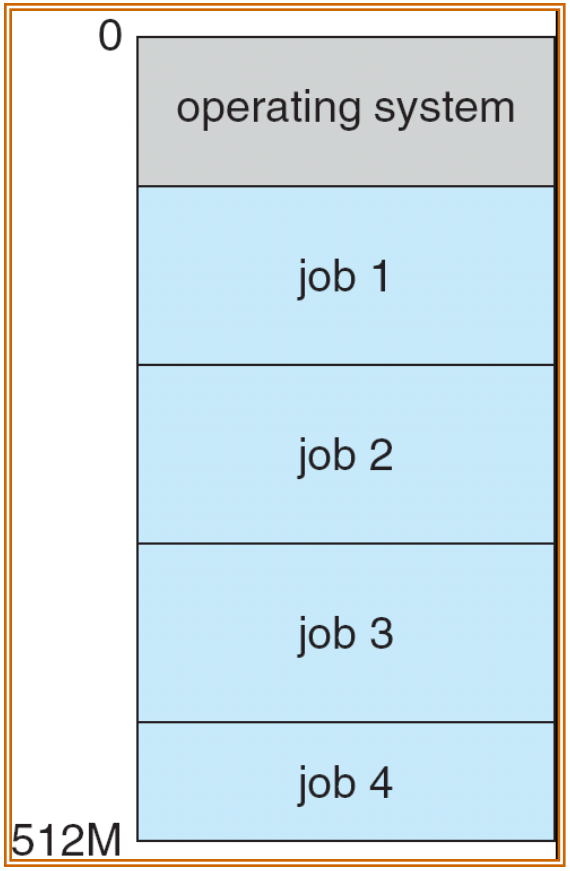
Batch OS


User 1
Job 1
User 2
Job 2
User n
Job n
Operator
Batch 2
Job 7
Job 5
Job 2
Batch 1
Job 4
Job 1
CPU
+
OS
User Program

Batch OS

- Advantages
  - Multiples users can share
- Disadvantages
  - Priority can be implemented
  - May lead to starvation
  - CPU may remain idle for a long time
  - Lack of interaction

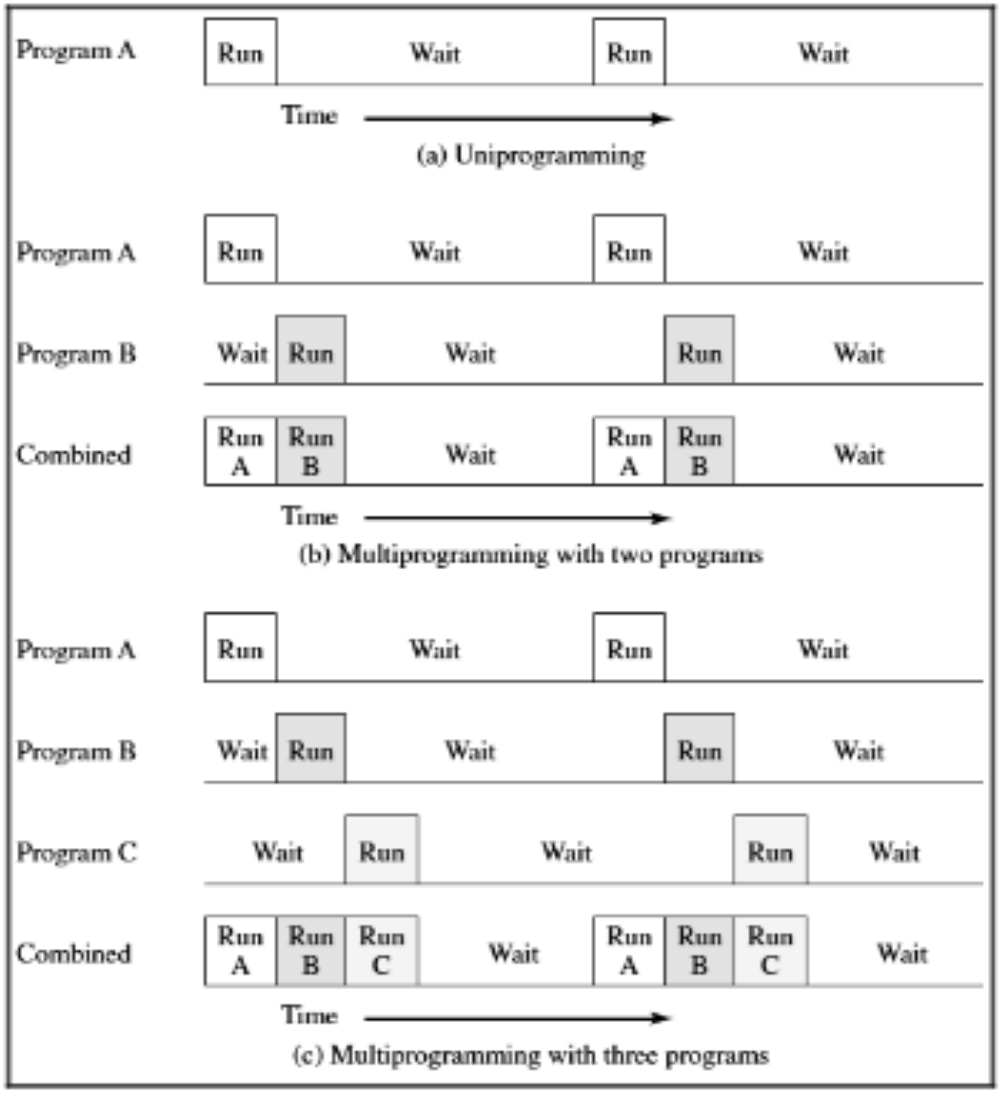
Memory Layout for Multiprogramming System



Multiprogramming needed for efficiency

- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- When it has to wait (for I/O for example), OS switches to another job

Multiprogramming Example



Question

There are three jobs running in a system with the following requirements:

Job1: Requires disk after every 2 min (I/O device service time = 2 min). Total processing time = 6 min.

Job2: Requires printer after every 5 min (I/O device service time = 2 min). Total processing time = 7 min.

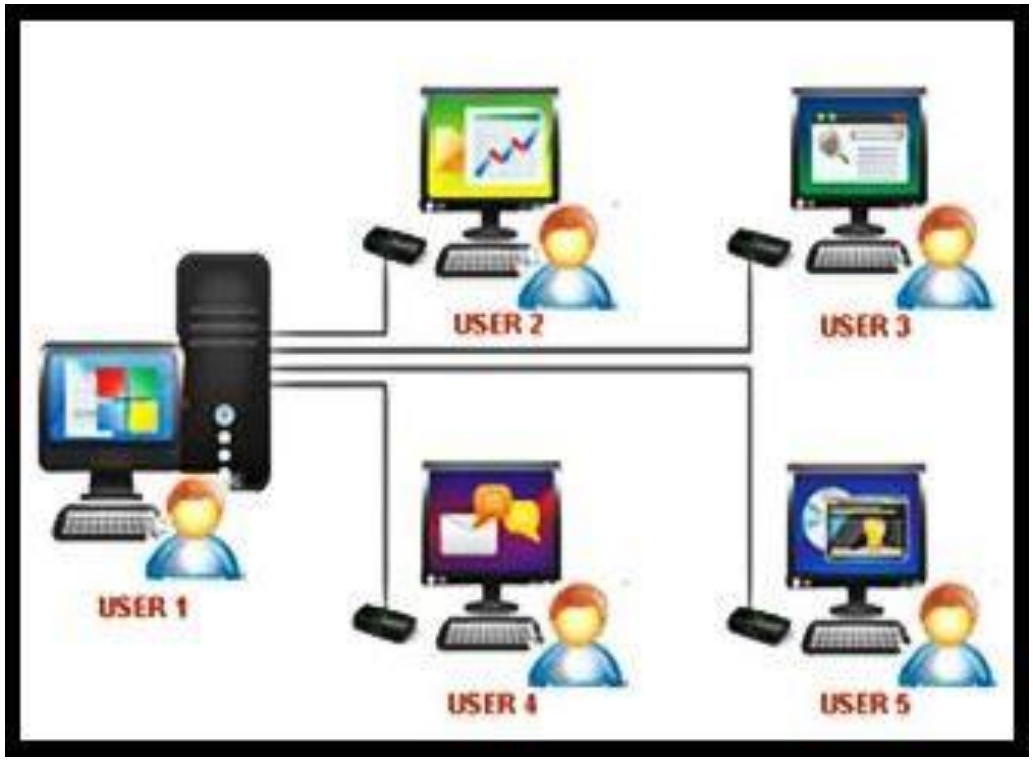
Job3: Requires disk after every 3 min (I/O device service time = 2 min). Total processing time = 5 min.

Compute the total time for execution using mono-programming and multi-programming. Assume the system with single CPU and single I/O processor.

Question

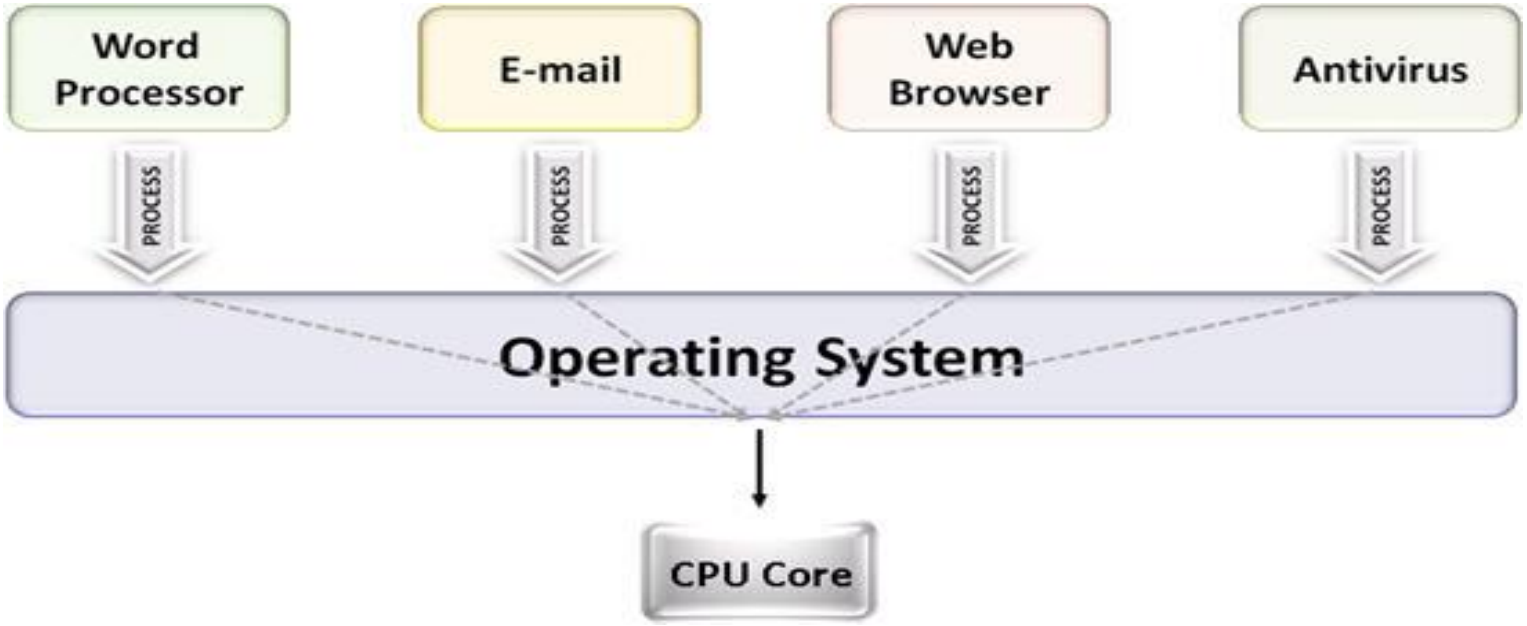
Suppose a new process in a system arrives at an average of six processes per minute and each such process requires an average of 8 seconds of service time. Estimate the fraction of time the CPU is busy in a system with a single processor.

Multi-user



Multitasking (or Time Sharing)

Multitasking



Time Sharing

- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Each user has at least one program executing in memory ⇨process
  - If several jobs ready to run at the same time ⇨ CPU scheduling
  - If processes don’t fit in memory, swapping moves them in and out to run
  - Virtual memory allows execution of processes not completely in memory

Job/CPU Scheduling

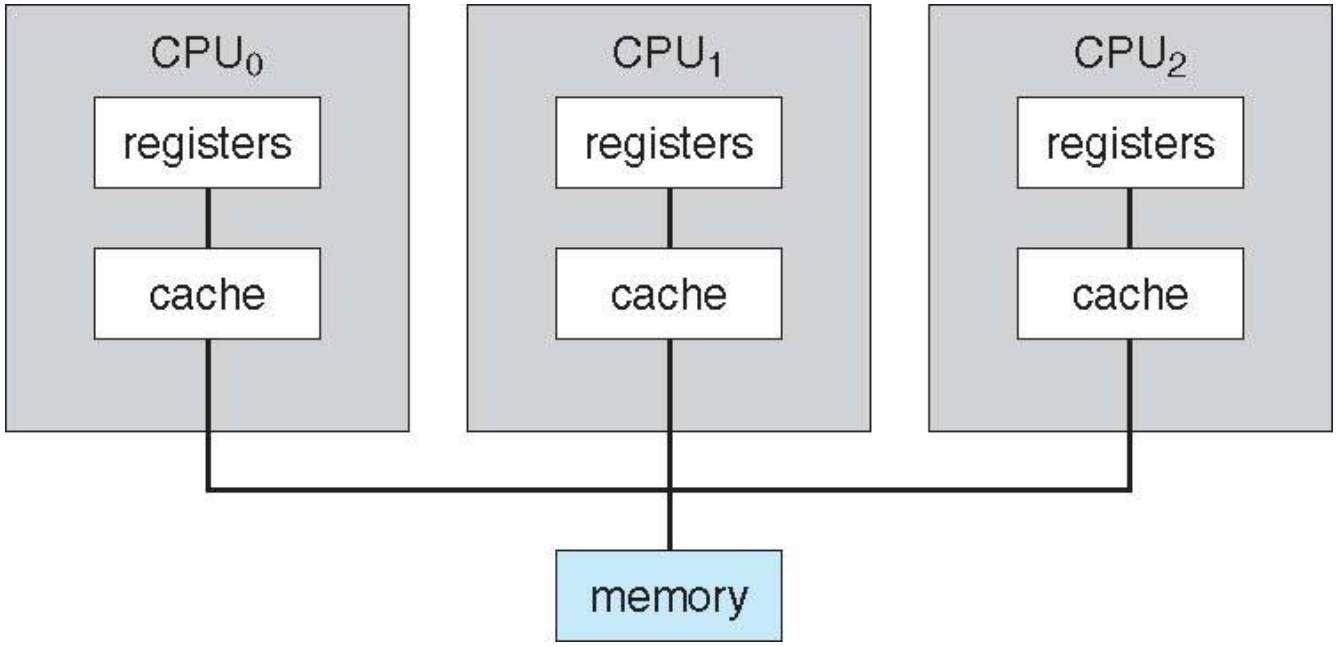
- *Job Scheduling* – Choosing the job among several jobs are ready to be brought into Memory when there is not enough room for all of them. (Bringing the particular job into ready queue)

- *CPU Scheduling* – Choosing the job among the several when jobs are ready to run at the same time. (Allocation of CPU to particular job)

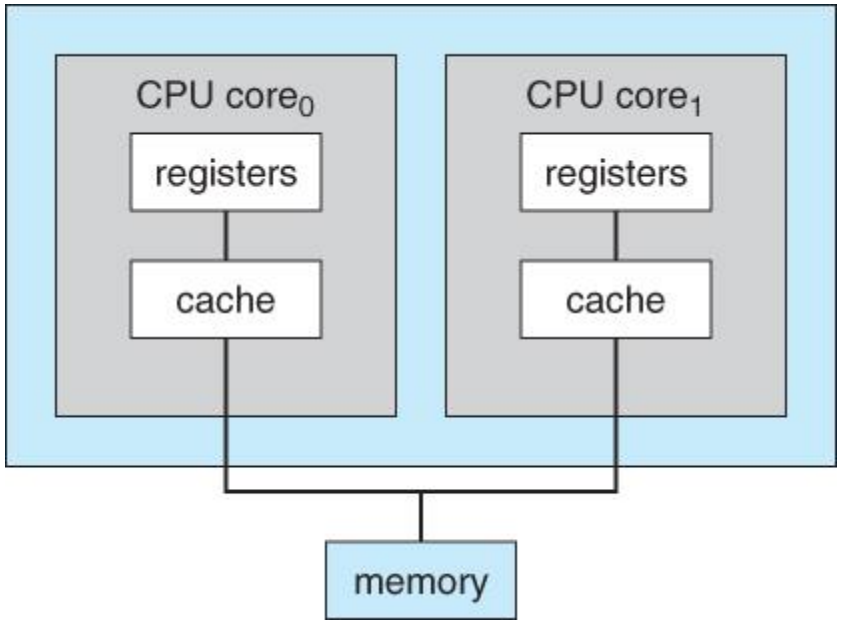
Multiprocessing

- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a special task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture

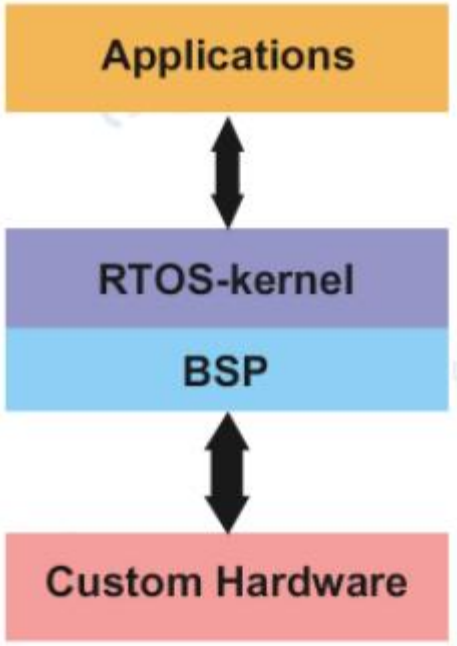


A dual-core design with two cores placed on the same chip



Real time OS (RTOS)

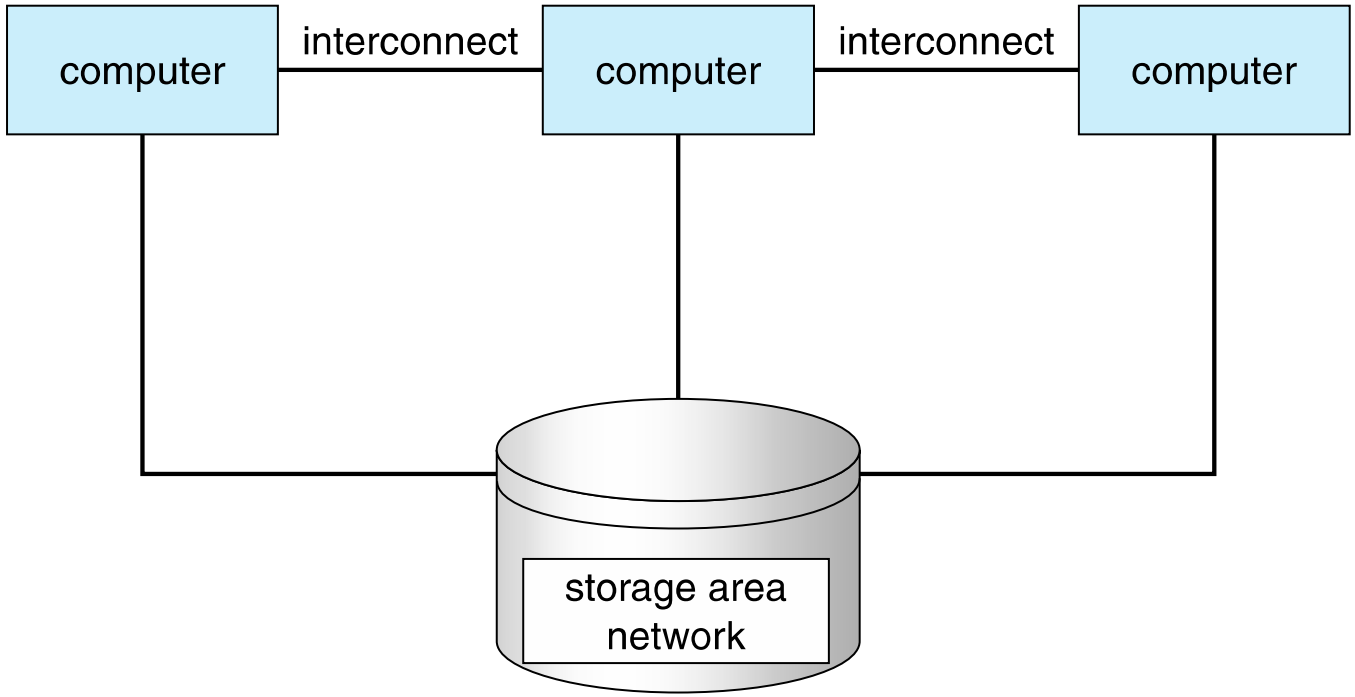
- An operating system which guarantees output or response within a specified time constraint.
- The real-time operating system can be classified in three categories:
  1. Hard RTOS
  2. Soft RTOS
  3. Firm



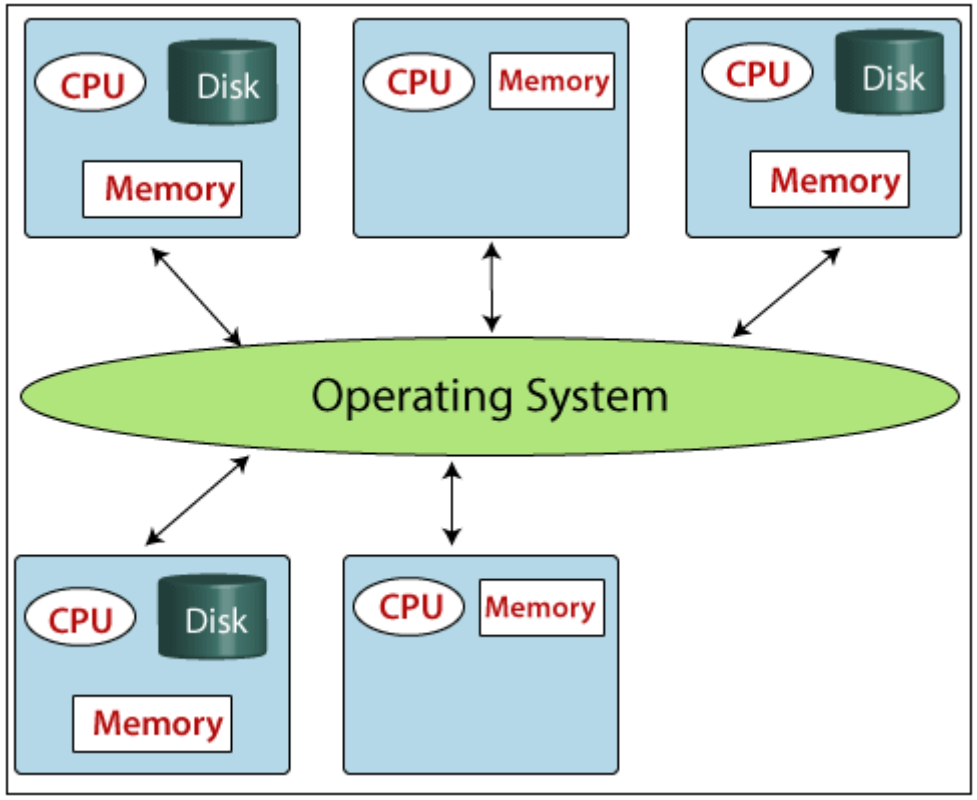
Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization**

Clustered Systems



Distributed OS



OS functions

- Process Management
- File Management
- Device Management
- Security Management
- Communications
- Command Interpreter

Process Management Activities

- 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

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

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

File System management

- Files usually organized into directories
- Access control on most systems to determine who can access what
- OS activities include
  - 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.
- Proper Management is of central importance
- Entire Speed of computer operation hinges on disk subsystem and its algorithms
- OS Activities
  - Free-Space Management
  - Storage Allocation
  - Disk Scheduling

I/O System

- 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, caching, spooling
  - General device-driver interface
  - Drivers for specific hardware devices

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 identities (user IDs, security IDs) Group ID Privilege escalation allows user to change to effective ID with more rights

Computing Environments - Traditional

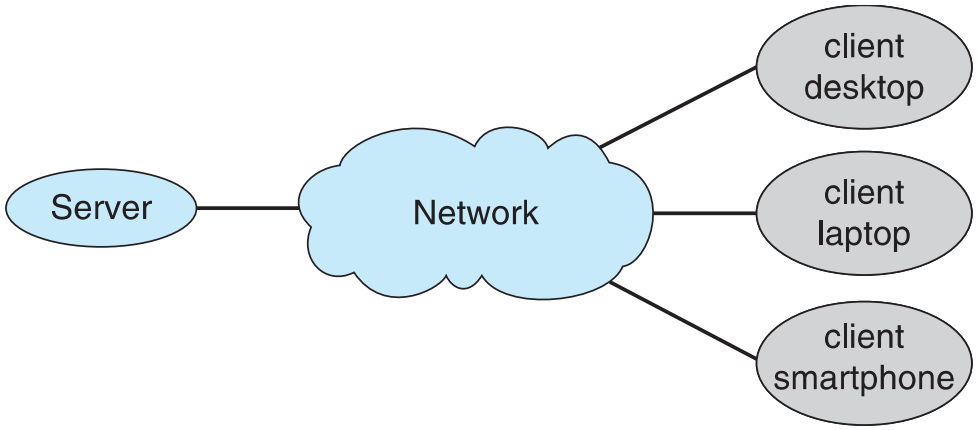
- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Computing Environments - Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a "traditional" laptop?
- Extra feature – more OS features (GPS)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

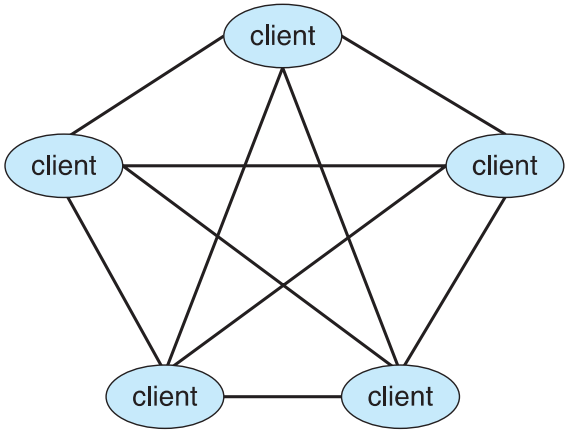
Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server system** provides an interface to client to request services (i.e., database)
    - **File-server system** provides interface for clients to store and retrieve files



Computing Environments - Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype



Classes of Operating Systems

OS Class	Period	Prime Concern	Key Concepts
Batch Processing Systems	1960s	CPU idle time	Automate transition between jobs
Time Sharing Systems	1970s	Good response time	Time-slice, round-robin scheduling
Multiprocessing Systems	1980s	Master/Slave processor priority	Symmetric/Asymmetric multiprocessing
Real Time Systems	1980s	Meeting time constraints	Real-time scheduling
Distributed Systems	1990s	Resource sharing	Distributed control, transparency
Desktop Systems	1970s	Good support to a single user	Word processing, Internet access
Handheld Systems	Late 1980s	32-bit CPUs with protected mode	Handle telephony, digital photography, and third party applications
Clustered Systems	Early 1980s	Low cost µps, high speed networks	Task scheduling, node failure management

Storage Definitions and Notation Review

The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
A **kilobyte**, or **KB**, is 1,024 bytes
a **megabyte**, or **MB**, is $1{,}024^2$ bytes
a **gigabyte**, or **GB**, is $1{,}024^3$ bytes
a **terabyte**, or **TB**, is $1{,}024^4$ bytes
a **petabyte**, or **PB**, is $1{,}024^5$ bytes

Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).