Syllabus:

Unit 1

Introduction: Define: Operating System (OS), Computer-system: Organization, Architecture, OS Structure and Operations

System Structure: OS Services, User Operating System Interface, System calls, System services, OS: Design and Implementation

Processes: Concept and Scheduling, Operations on Processes, Interposes communication

Thread and Concurrency: Overview, Multicore programming, Multithreading models

Unit 2

CPU Scheduling: Basic concepts, Scheduling criteria and algorithms, Thread scheduling

Synchronization Tools: Introduction, The criteria-section problem, Peterson's solution, Hardware support for synchronization, Mute locks, Semaphores, Monitors, Synchronization examples

Unit 3

Deadlocks:

System Model, Deadlock in multithreaded applications, Deadlock characterization, Methods for handling deadlocks, Deadlock: Prevention, Avoidance, Detection and Recovery

Unit 4

Memory Management:

Main Memory – Introduction, Contiguous memory allocation, Paging, Structure of Page Table, Swapping

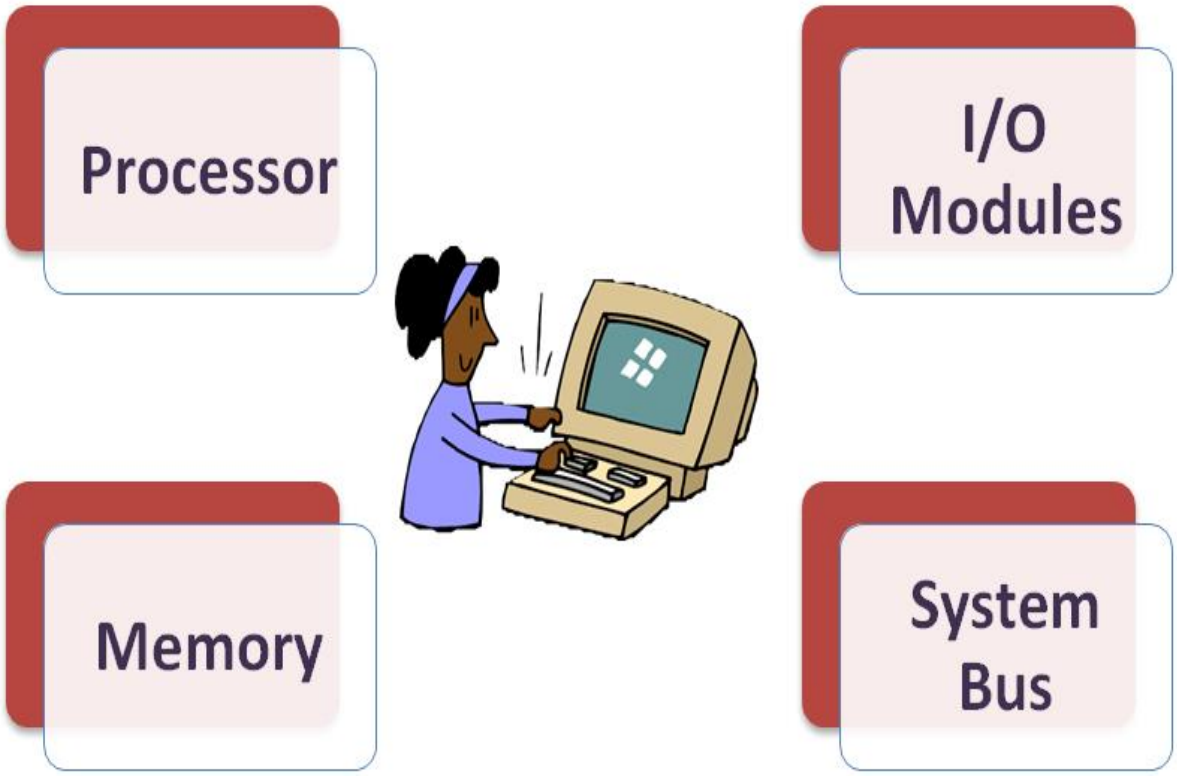
Virtual Memory – Introduction, Demand paging, Copy-on write, Page replacement, Allocation frames, Thrashing and Memory compression

Reference Books:

1. Operating System Principles, ABRAHAM SILBERSCHATZ, PETER BAER GALVIN, GREG GAGNE, WILEY-INDIA EDITION
2. Tanenbaum A.S., "Modern Operating Systems", 4th Edition, PHI, 2001
3. Flynn I.M, "Understanding Operating Systems", Cengage India Publication
4. Bach M J, "The Design of UNIX Operating System", Prentice Hall India, 1993.

Computer System Overview

1. Basic elements of computer



✓Processor

Referred to as the ***Central Processing Unit*** **(CPU)**

Arithmetic & Logic Unit

Performs the **data processing** functions

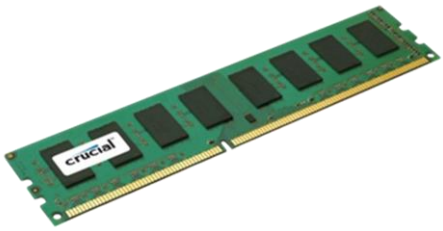
Control Unit

Controls the operation of the computer

Note: Which two companies are famous for manufacturing computer processors?

✓ Memory

Memory is device that is used to store data/information



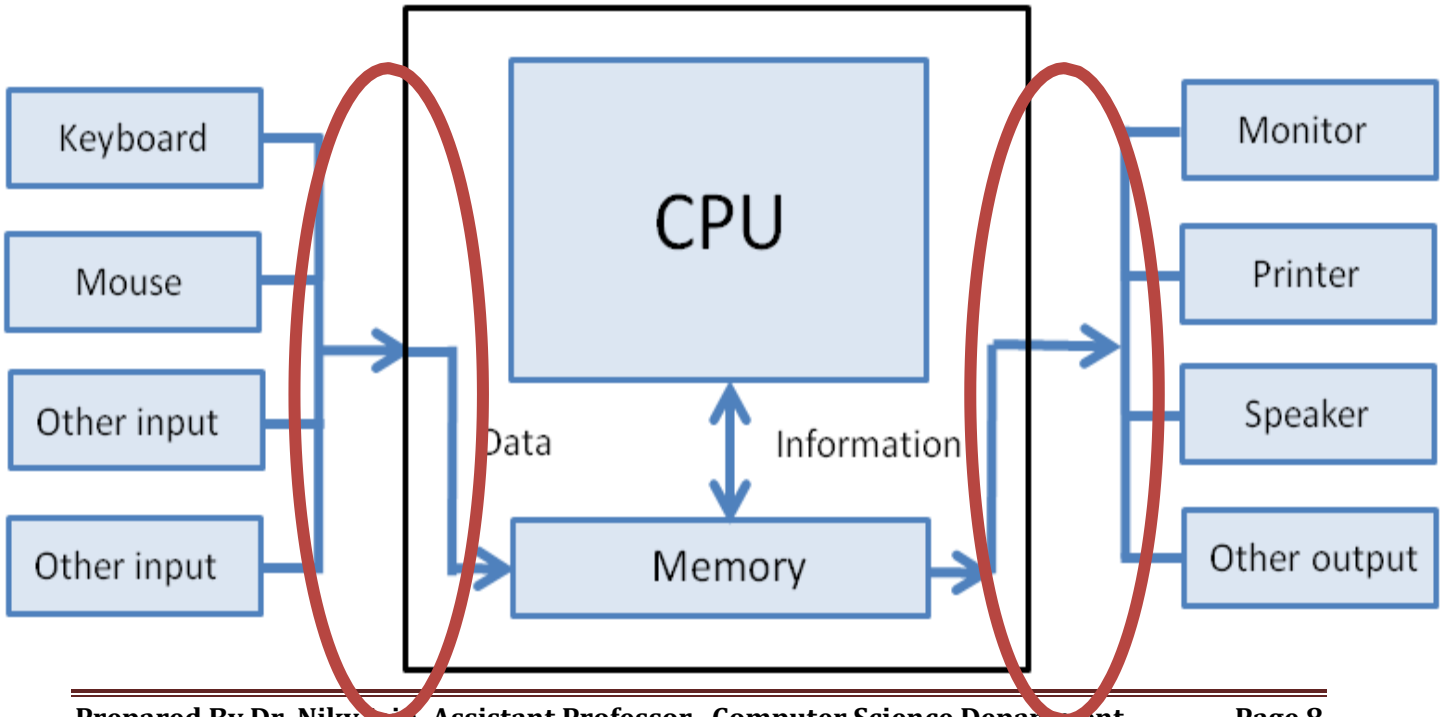
Primary Memory
Volatile
RAM & ROM

Secondary Memory
Non-Volatile
HDD, CD & DVD

Question 1: Give the difference between primary memory and secondary memory

✓ I/O Module (Input/Output Module)

- Input/output module is a device that acts as the connective bridge between a computer system at one end and an I/O or peripheral device at the other, such as a printer, webcam or scanner.
- An I/O module is a mediator between the processor/memory an I/O devices.
- It controls the data exchange between the external devices and main memory or external devices and CPU registers.



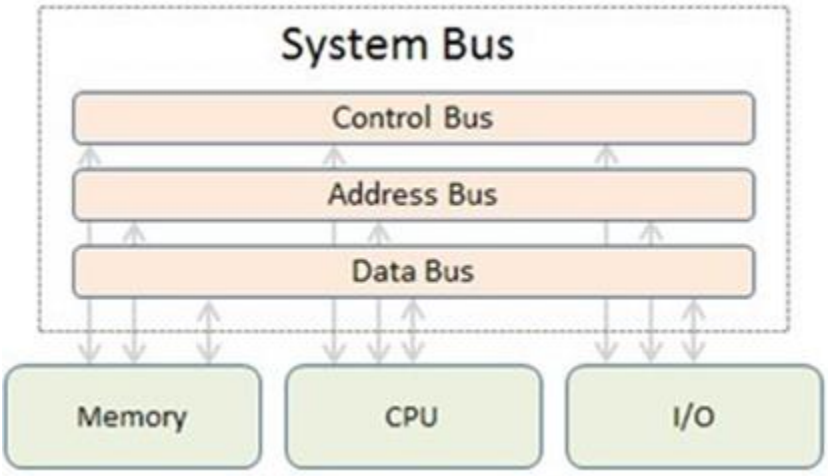
✓ System Bus

- **Definition:** Provides communication among the **processor, main memory, and I/O devices**.
- The **system bus** is a pathway composed of cables and connectors used to carry data between the computer's **microprocessor** and **main memory**.

Types of Buses

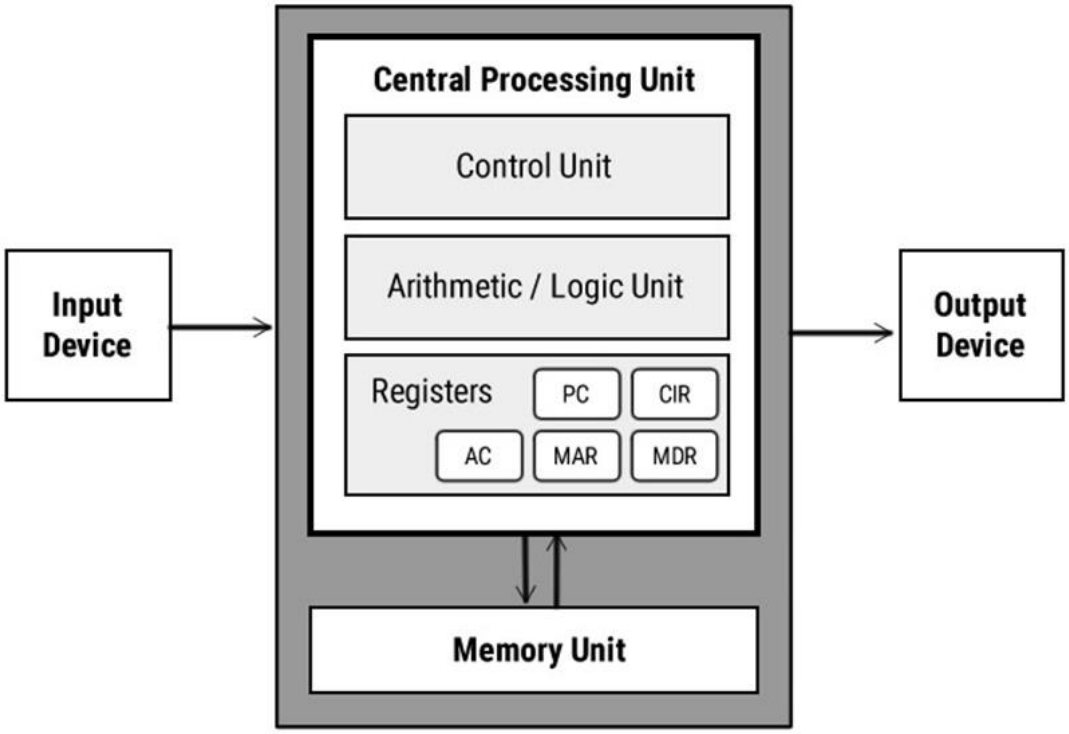
- **Address Bus** – Carries memory addresses from the processor to other components such as primary storage and I/O devices.
- **Data Bus** – Carries the actual data between the processor and other components.
- **Control Bus** –Carries control signals (like read/write, clock, interrupt requests) from the processor to other components.

Note: What are the 3 types of buses?



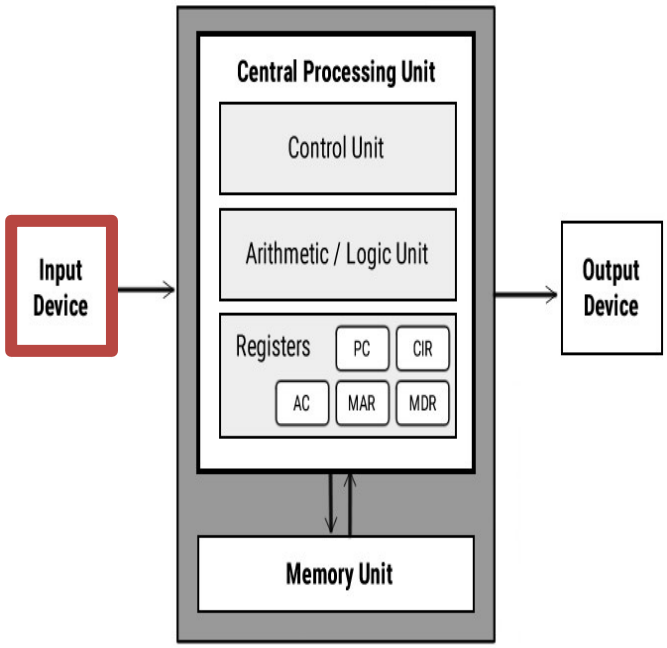
Computer system architecture

✓Computer system architecture



➢Input unit

- **It provides data and instructions to the computer system.**
- Commonly used input devices are **keyboard, mouse, magnetic tape**, etc.

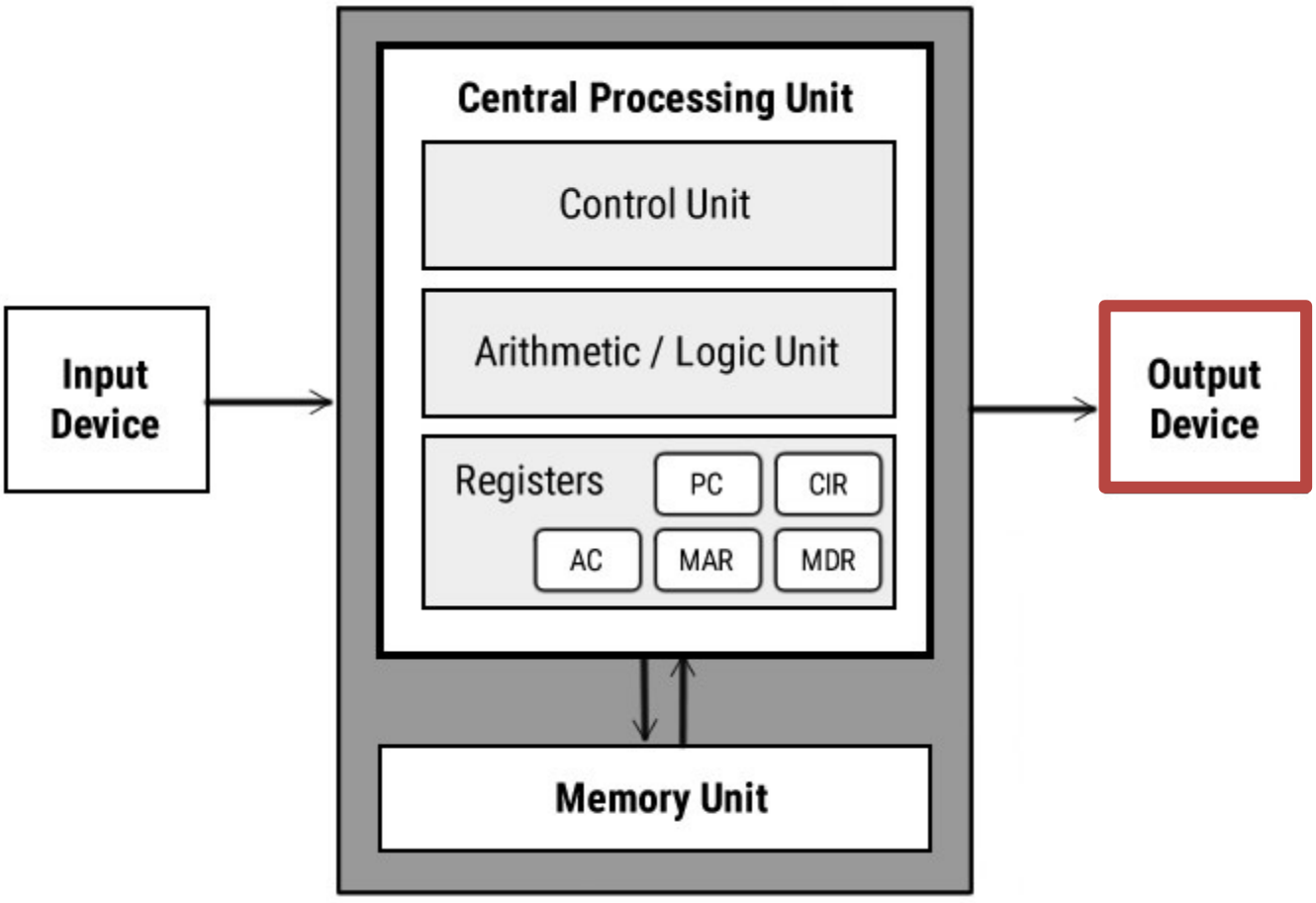


Tasks performed by Input Unit:

- Accepts the data and instructions from the outside environment.
- Converts them into machine language.
- Supplies the converted data to the computer system.

➢Output unit

- **It connects the internal system of a computer to the external environment.**
- It provides the results of any computation or instructions to the outside world.
- Common output devices include **printers, monitors**, etc.

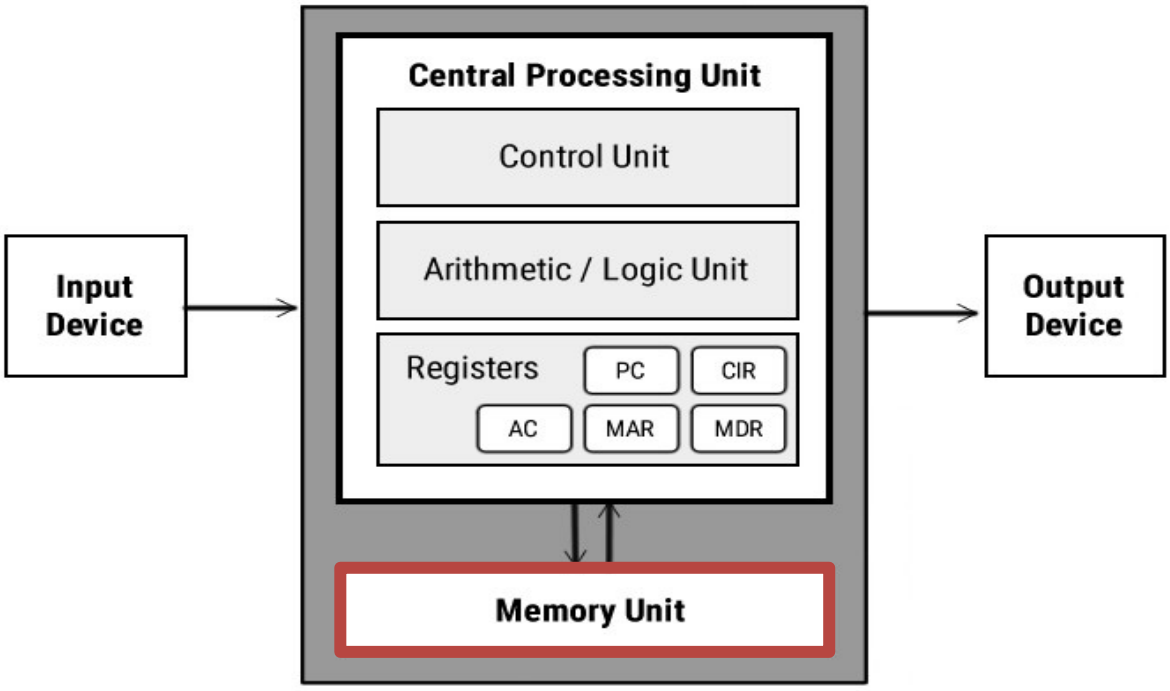


➢Storage unit

- **This unit holds the data and instructions.**
- It also stores the **intermediate results** before they are sent to the output devices.
- Stores data for **later use**.

Categories of Storage Unit:

- **Primary Storage**
- **Secondary Storage**



Primary storage (memory) vs Secondary storage (memory)

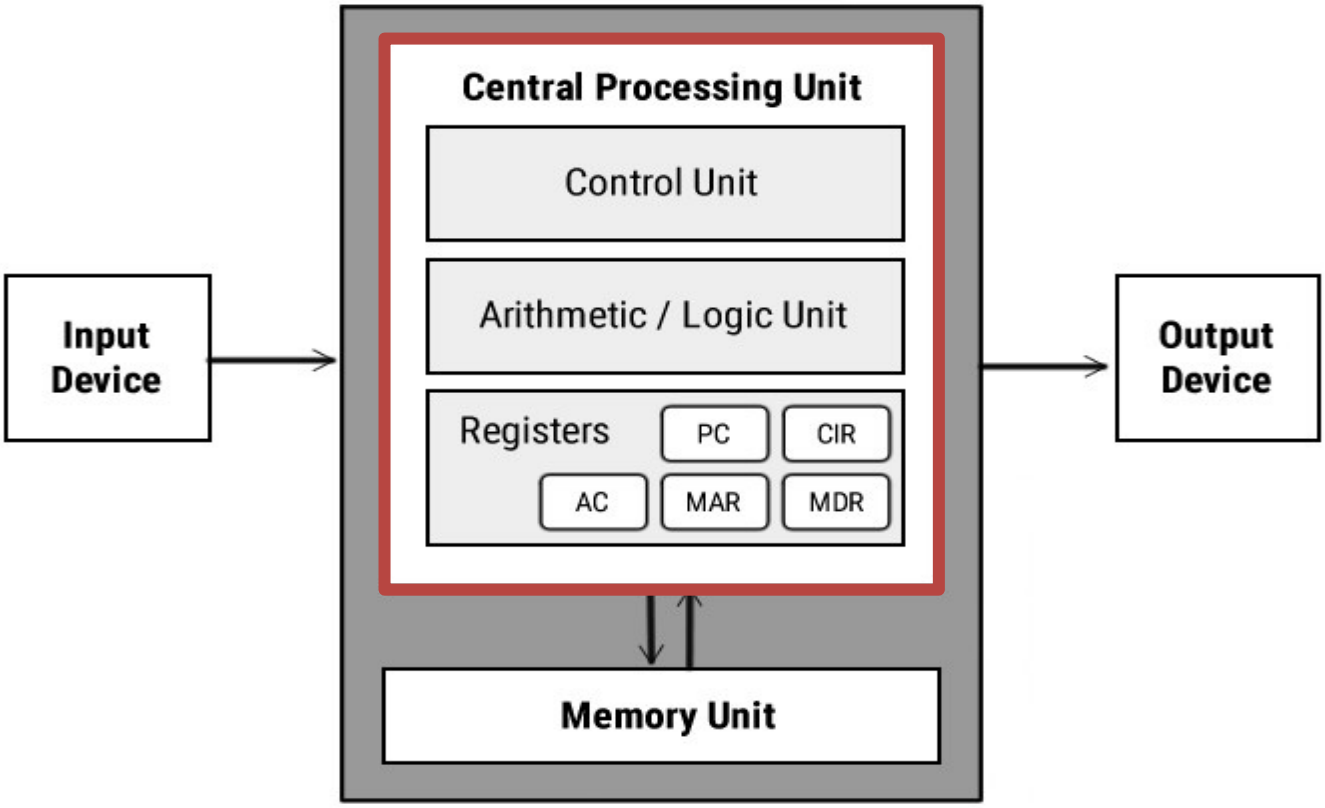
Primary storage	Secondary storage
Examples: RAM, ROM, Cache memory, PROM, EPROM, Registers, etc.	Examples: Hard Disk, Floppy Disk, Magnetic Tapes, etc.
It is temporary and volatile.	It is permanent and Non-volatile.
Primary memory is directly accessible by Processor/CPU.	Secondary memory is not directly accessible by the CPU.
Primary memory devices are expensive.	Secondary memory devices are cheaper.
The memory devices used for primary memory are semiconductor memories.	The secondary memory devices are magnetic and optical memories.
Primary memory is also known as Main memory or Internal memory.	Secondary memory is also known as External memory or Auxiliary memory.

➢CPU (Central Processing Unit)

- ❖ The **Arithmetic Logical Unit (ALU)** and **Control Unit (CU)** together form the **CPU**.
- ❖ CPU is considered the **brain of the computer system**.

Tasks performed by CPU:

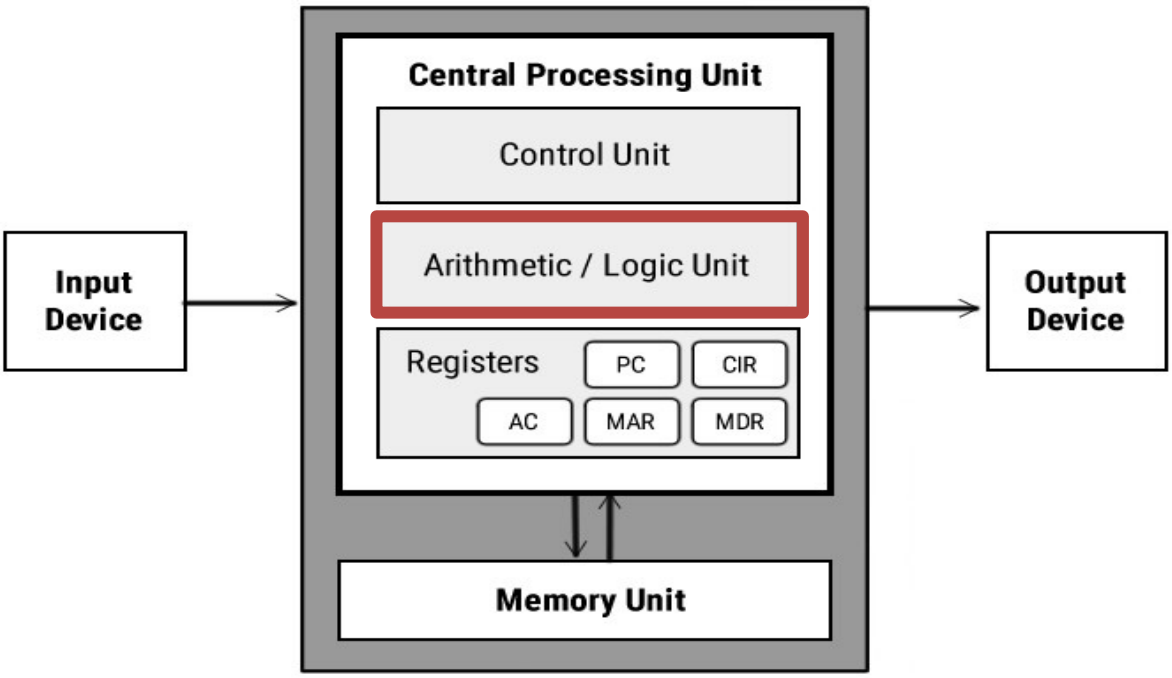
- ❖ Performs all operations.
- ❖ Takes all decisions.
- ❖ Controls all the units of the computer.



✓ALU (Arithmetic Logical Unit)

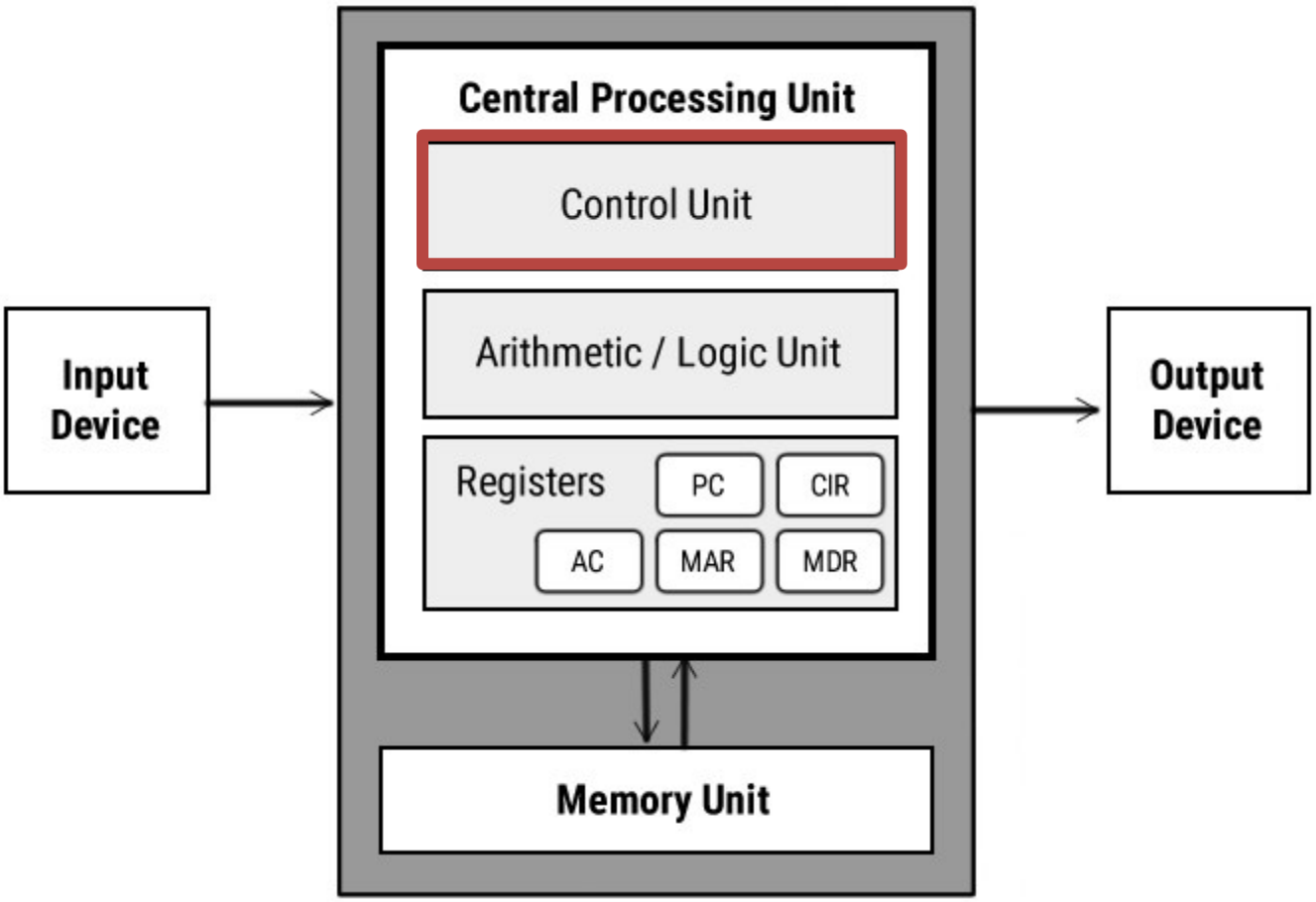
- ❖ All the **calculations** are performed in the **ALU** of the computer system.
- ❖ The ALU can perform **basic operations** such as:
  - Addition
  - Subtraction
  - Division
  - Multiplication

- ❖ Whenever calculations are required, the **Control Unit (CU)** transfers the data from the **Storage Unit** to the ALU.
- ❖ After the operations are done, the **result is transferred back** to the storage unit.



✓CU (Control Unit)

- The **Control Unit** controls all other units of the computer.
- It manages the **flow of data and instructions** between the **Storage Unit** and the **Arithmetic Logic Unit (ALU)**.
- Hence, it is also known as the **Central Nervous System** of the computer.



What is Operating System (OS)

✓What is Operating System (OS)?

- A Computer System consists of various hardware's such as

Processor

RAM

Keyboard & Mouse

Hard Disk

Monitor

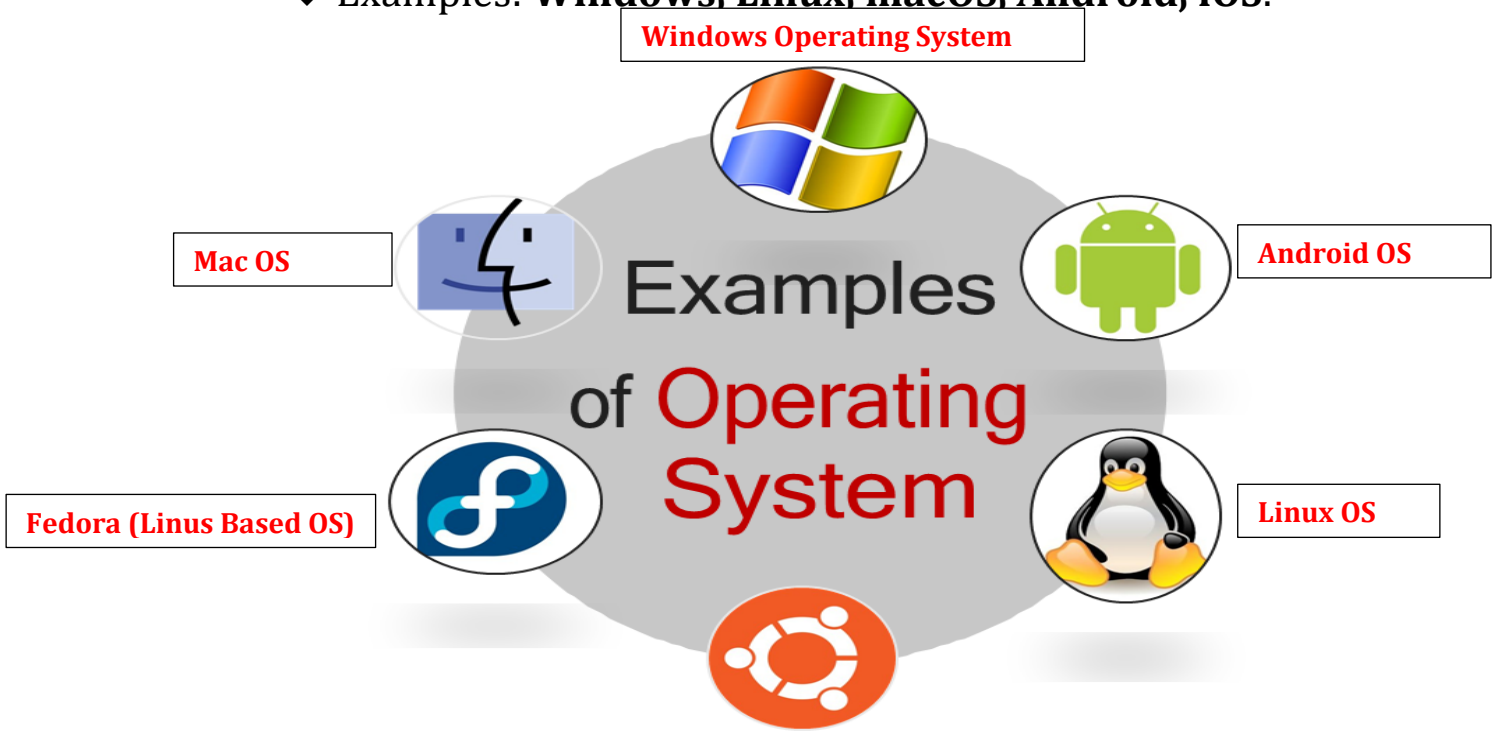
Printer

Who manages (controls) these hardwares??? → **Operating System**

✓Definition of Operating System (OS)

An **Operating System (OS)** is a **system software** that acts as an **interface between the user and the computer hardware**.

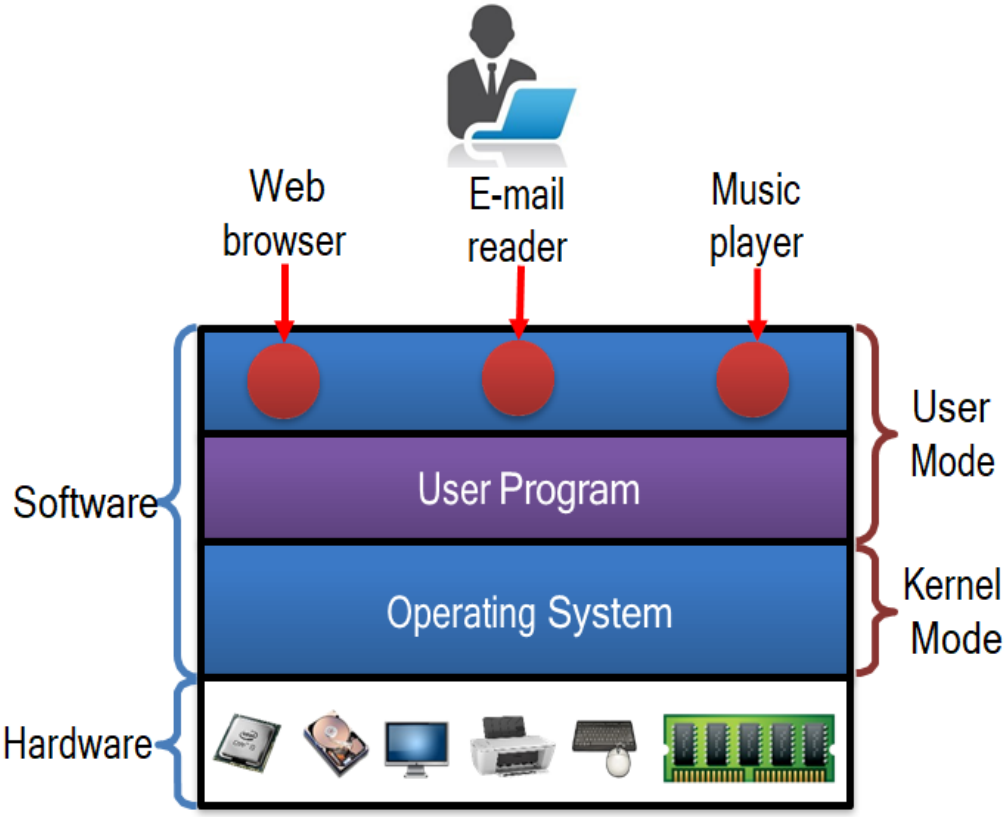
- ❖ It manages the computer's hardware, software resources, and provides common services for programs.
- ❖ The OS makes the computer system **convenient, efficient, and easy to use**.
- ❖ OS is the **manager of all resources** (CPU, memory, I/O devices, files).
- ❖ It acts as a **bridge between user programs and hardware**.
- ❖ Examples: **Windows, Linux, macOS, Android, iOS**.



✓Where OS lies? (Interaction of OS & Hardware)

Operating System as an Intermediary

- The **Operating System (OS)** lies **between hardware and user programs**.
- It acts as an **intermediary** between the **user** and the **computer hardware**, ensuring smooth interaction.



Modes of Operation of a Computer

1. Kernel Mode

- Has **complete access** to all hardware resources.
- Can execute **any machine instruction**.
- Possesses **high privileges (rights)**.
- Example: CPU executes OS kernel instructions like process scheduling, memory management, device control, etc.

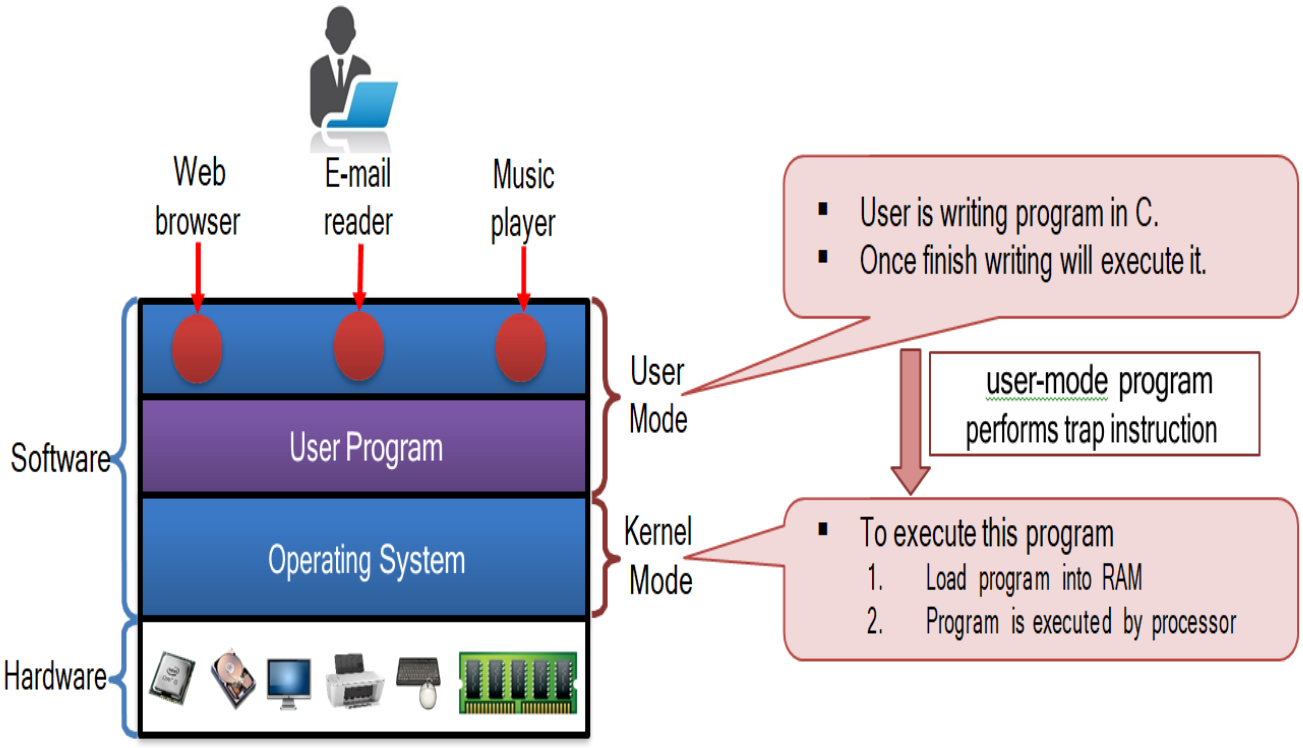
2. User Mode

- Has **limited access** to hardware resources.
- Can execute only a **subset of instructions**.
- Possesses **less privileges (rights)**.
- Example: Applications like MS Word or a browser run in **user mode** and request OS services when needed.

Note:

- **Kernel Mode = Full authority (OS level work)**
- **User Mode = Limited authority (Application level work)**

Why and How switch occur?



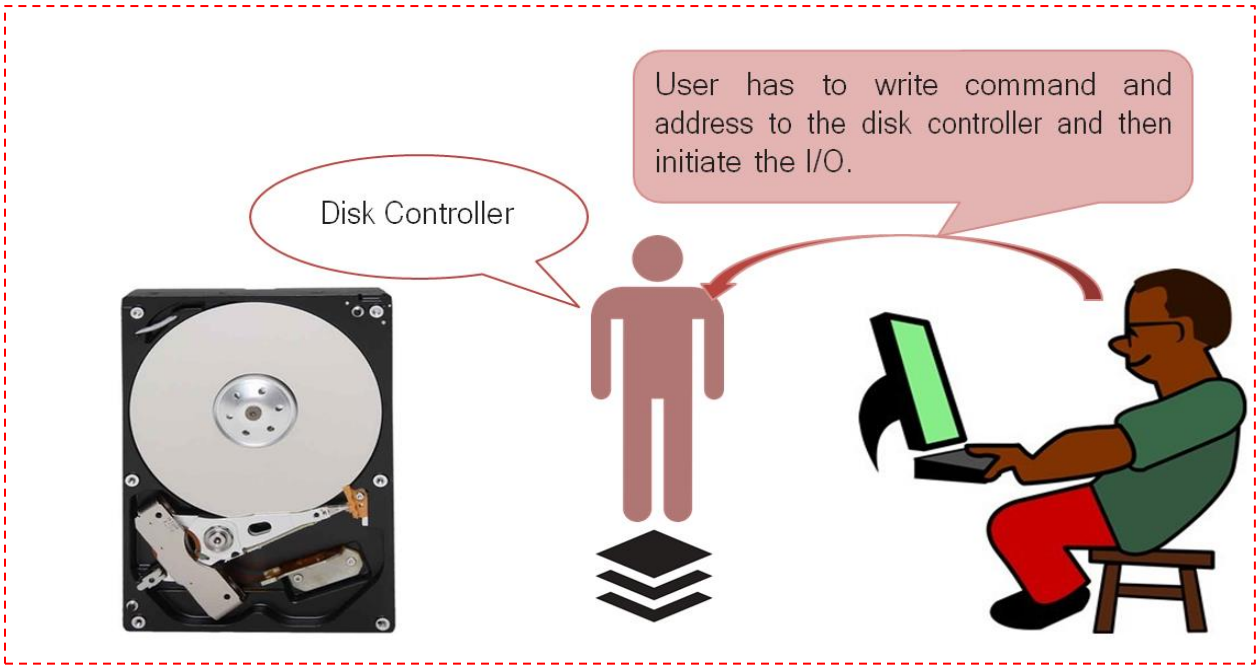
Roles of Operating System (OS)

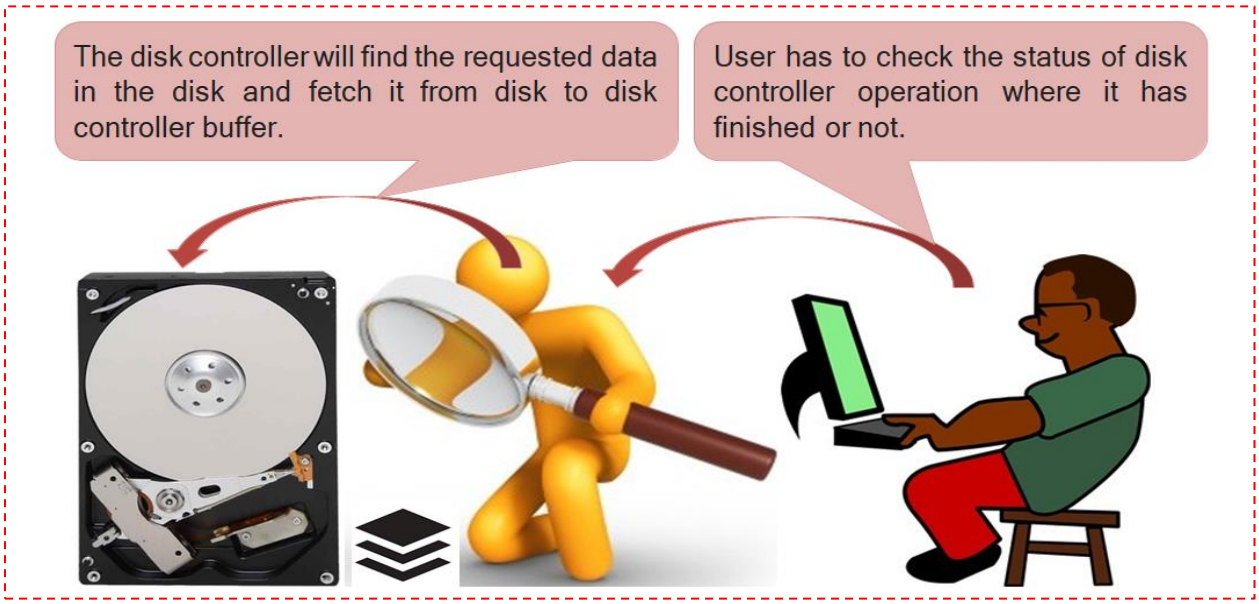
OS as Extended Machine

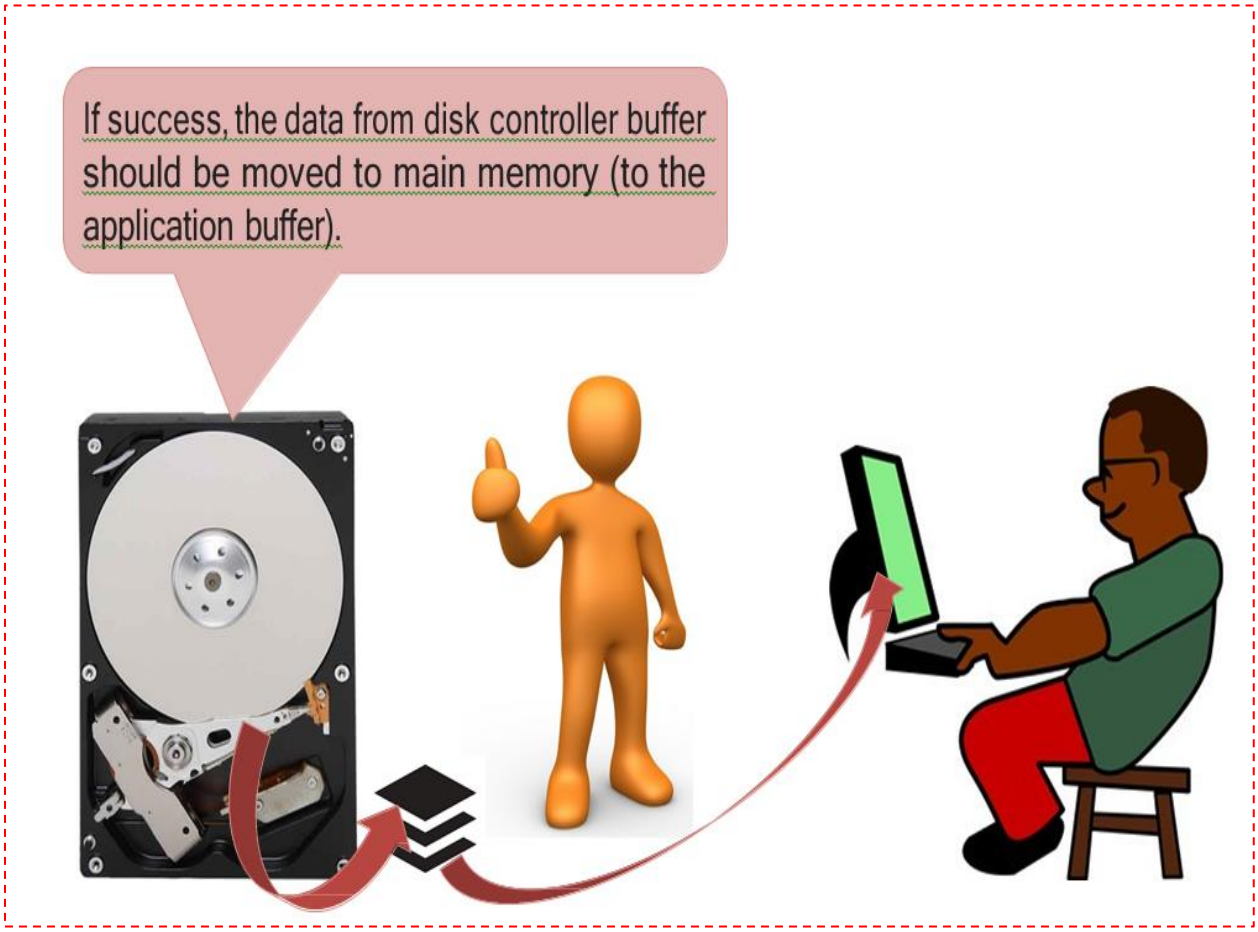
- **The Architecture of a Computer is Difficult to Program**
  - ✓ The **architecture** of a computer — including **instruction set, memory organization, I/O devices, and bus structure** — is **primitive and awkward** to handle directly at the **machine language level**.
  - ✓ Programming at this level is **complex, time-consuming, and error-prone**.

Example: If **user want to read from floppy or hard disk:**

Example: If user want to read from floppy or hard disk:






If success, the data from disk controller buffer should be moved to main memory (to the application buffer).

Why Operating System is Needed?

- ✓ If all the users had to manage these **messy hardware details**:
  - Programs would be very difficult to write and unnecessarily long.
  - Programs would be **hardware dependent**.

User Perspective

- ✓ Users do **not want** to be involved in programming storage devices.
- ✓ They need **simple and easy commands** to perform tasks.

Role of OS

- ✓ The **Operating System** provides a set of **basic commands** or instructions to perform operations such as:
  - **Read**
  - **Write**
  - **Modify**
  - **Save**
  - **Close**
- ✓ Dealing with these commands is **much easier** than directly handling hardware.
- ✓ The OS **hides the complexity** of hardware and presents a **user-friendly interface** to the users.

OS as Resource Manager

A computer system consists of several important resources, such as:

- ✓ **CPU (Processor)**
- ✓ **Memory (Primary & Secondary Storage)**
- ✓ **Input/Output Devices** – e.g., Hard Disk, Mouse, Keyboard, Printer, Scanner, etc.

Resource Sharing Challenge

- If a computer system is used by **multiple applications** (or **multiple users**) at the same time,
- They will **compete for these resources**.

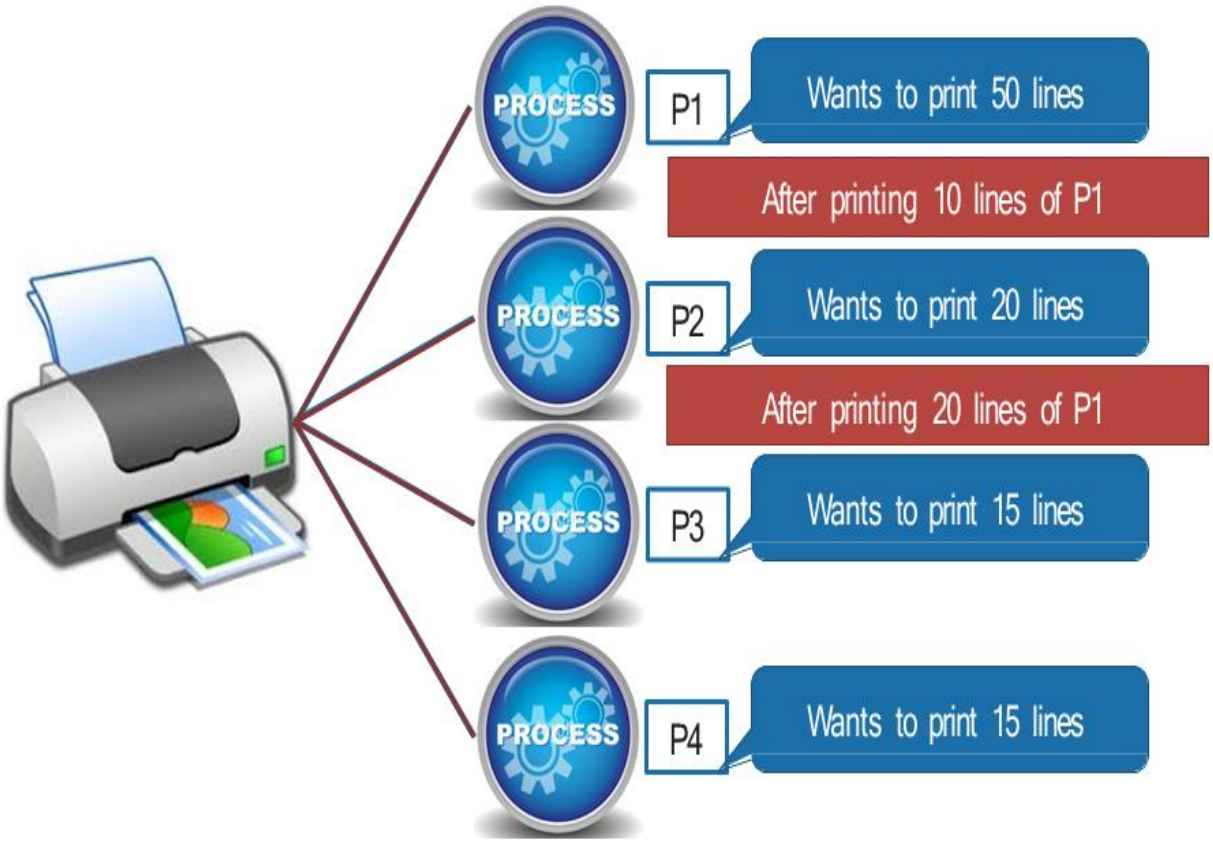
Example:

- ✓ Two programs may try to use the **CPU** simultaneously.
- ✓ Multiple users may try to **print** at the same time.
- ✓ Several applications may request **memory space** together.

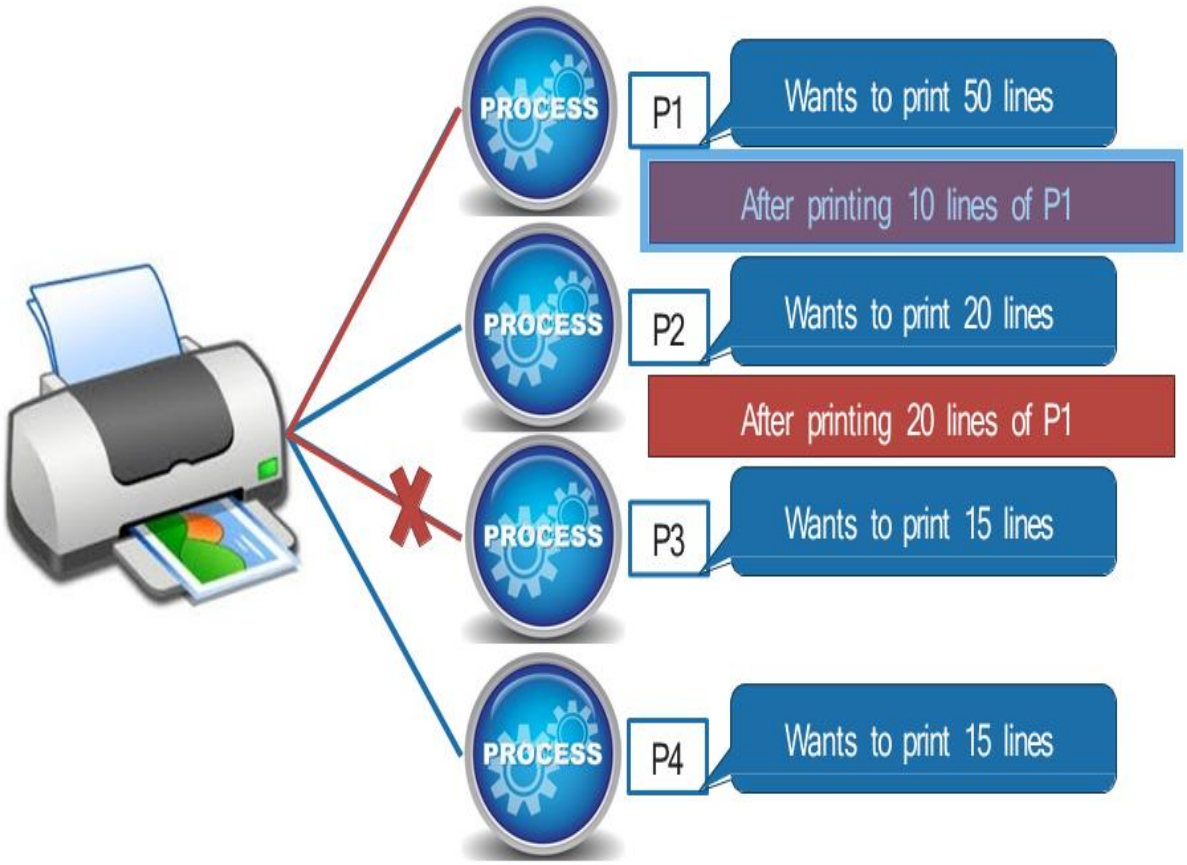
Role of OS

- ✓ The **Operating System** acts as a **resource manager**.
- ✓ It **allocates resources fairly and efficiently** among users/programs.
- ✓ Ensures that one program/user does not **block or misuse** resources needed by others.

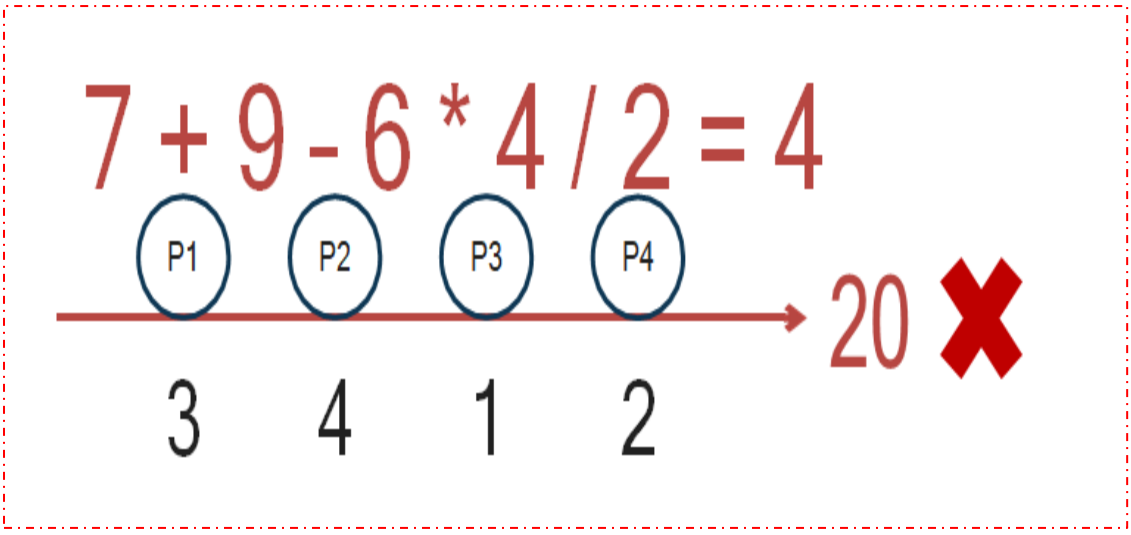
- It is the job of OS to allocate these resources to the various applications so that:
  - The resources are allocated fairly (equally)



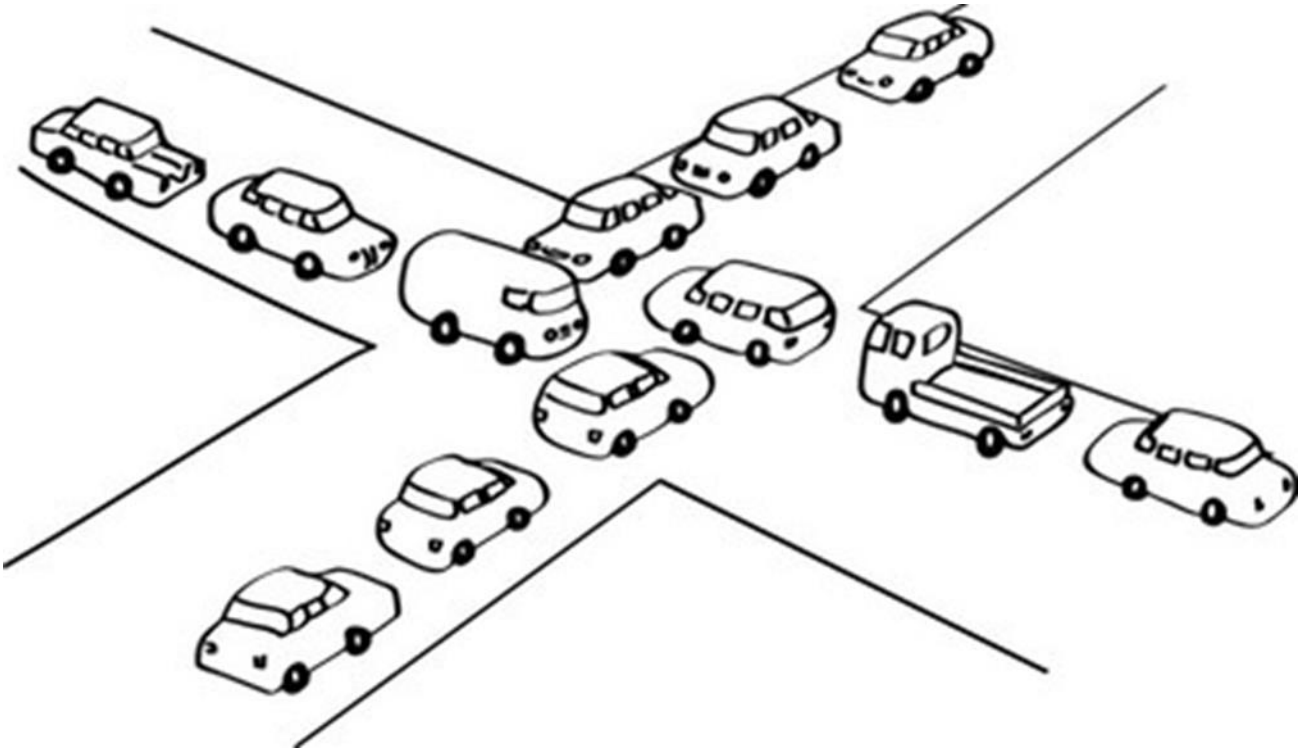
- It is the job of OS to allocate these resources to the various applications so that:
  - The resources are protected from cross-access.



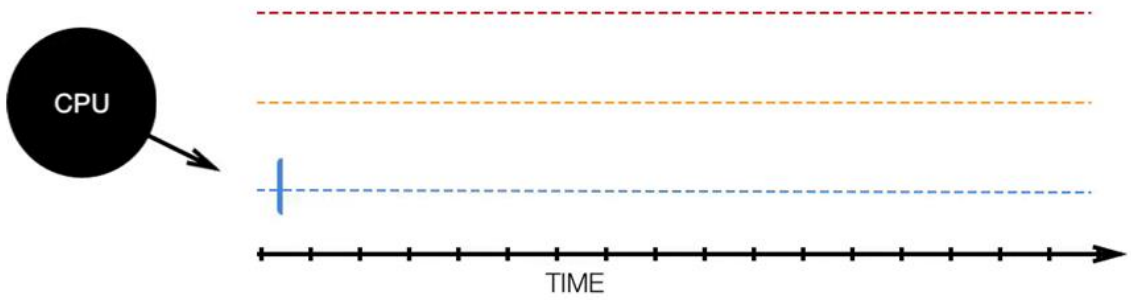
- It is the job of OS to allocate these resources to the various applications so that:
  - ✓ Access to the resources is synchronized so that operations are correct and consistent
  - ✓ Example: If we write a program to calculate below in C language



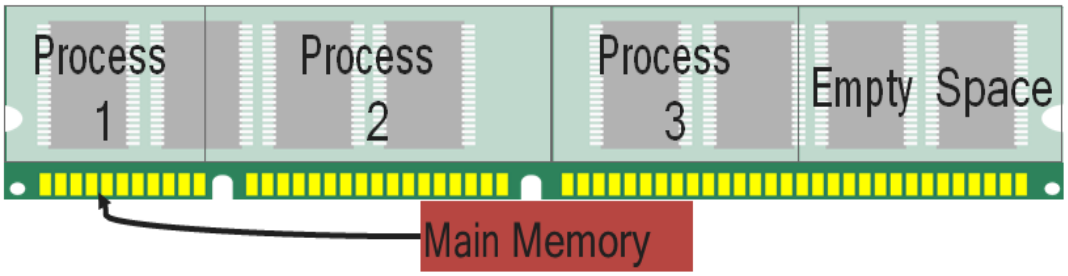
- It is the job of OS to proper allocate these resources to the various applications so that:
  - ✓ Deadlock are detected, resolved and avoided.



- Resource manager – sharing resources in two different ways:
  - ✓ In time sharing/multiplexing (i.e CPU)



  - ✓ In space sharing/multiplexing. (i.e Memory)



Objectives / Goals of Operating System (OS)

- Make the computer system convenient to use in an efficient manner.
- Hide the details of the hardware resources from the users.
- Provide users a convenient interface to use the computer system.
- Act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- Manage the resources of a computer system.
- Keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- Provide efficient and fair sharing of resources among users and programs.

Generations of Operating Systems (OS)

➢ First generation (1945-1955)

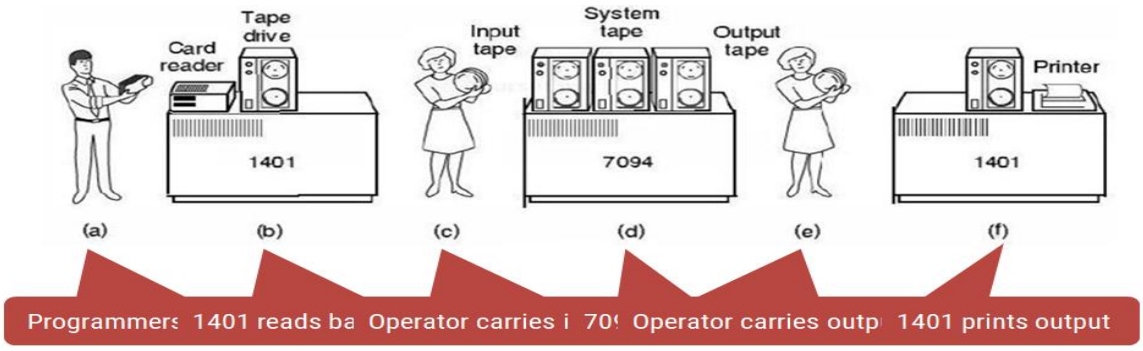
- Vacuum tubes and plug-boards are used in these systems.

Vacuum tubes

Plug board

➢ Second generation (1955-1965)

- Transistors are used in these systems
- The machine that are produced are called mainframes.
- Batch systems was used for processing.



➢ Third generation (1965-1980)

- Integrated circuits (IC's) are used in place of transistors in these computers.
- It provides multiprogramming (the ability to have several programs in memory at once, each in its own memory partition).

➢ Fourth generation (1980-present)

- Personal Computers (PC)
- LSI (Large Scale Integration) circuits, chips containing thousands of transistors are used in these systems.

Operating Systems (OS) services

Definition of Operating System

An Operating System (OS) is a collection of software that

- ✓ **manages hardware resources**
- ✓ **provides various service to the user**

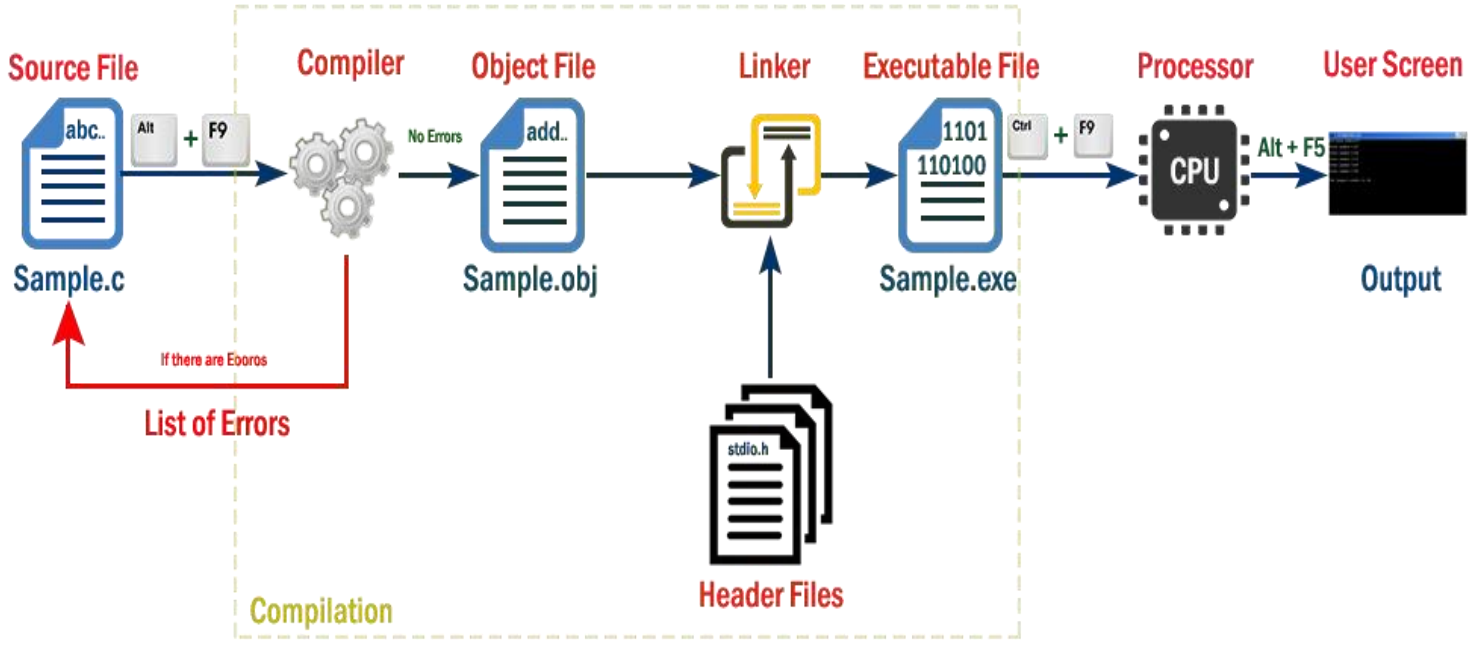
1. Program development
2. Program execution
3. Access to I/O devices (Resource allocation)
4. Memory management
5. Controlled access to file
6. Communication
7. Error detection and response
8. Accounting
9. Protection & Security

1. Program development

- It provides editors and debuggers to assist (help) the programmer in creating programs.

2. Program execution

- Following tasks need to be perform to execute a program:
  - Instructions and data must be loaded into main memory.
  - I/O devices and files must be initialized.
- The OS handles all these duties for the user.

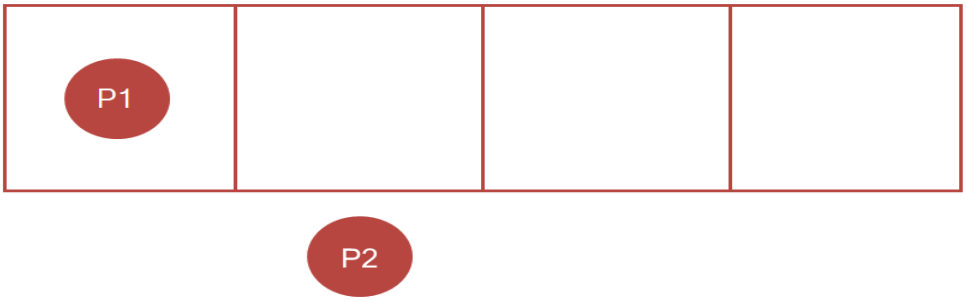


3. Access to I/O devices (Resource allocation)

- A running program may require I/O, which may involve file or an I/O device.
- For efficiency and protection, users cannot control I/O devices directly.
- Therefore, the OS controls these I/O devices and provides to program as per requirement.

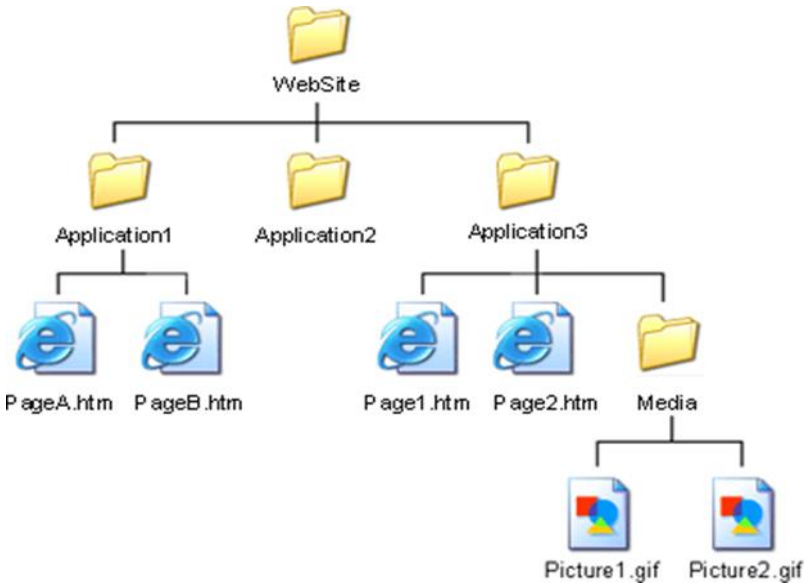
4. Memory management

- OS manages memory hierarchy.
- OS keeps the track of which part of memory area in use and free memory.
- It allocates memory to program when they need it.
- It de-allocate the memory when the program finish execution.



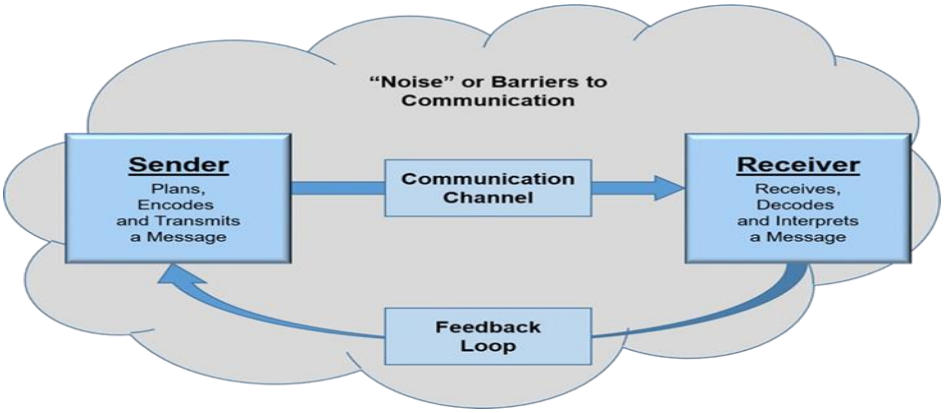
5. Controlled access to file

- In case of file access, OS provides a directory hierarchy for easy access and management of file.
- OS provides various file handling commands using which user can easily read, write and modify file.



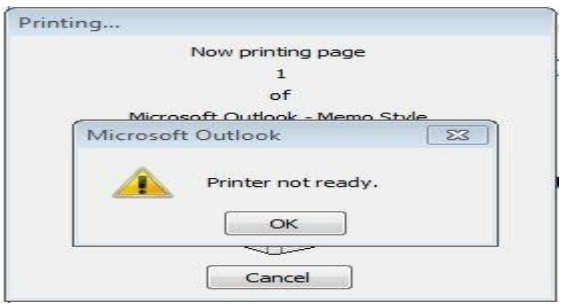
6. Communication

- In multitasking environment, the processes need to communicate with each other and to exchange their information.
- Operating system performs the communication among various types of processes in the form of shared memory.



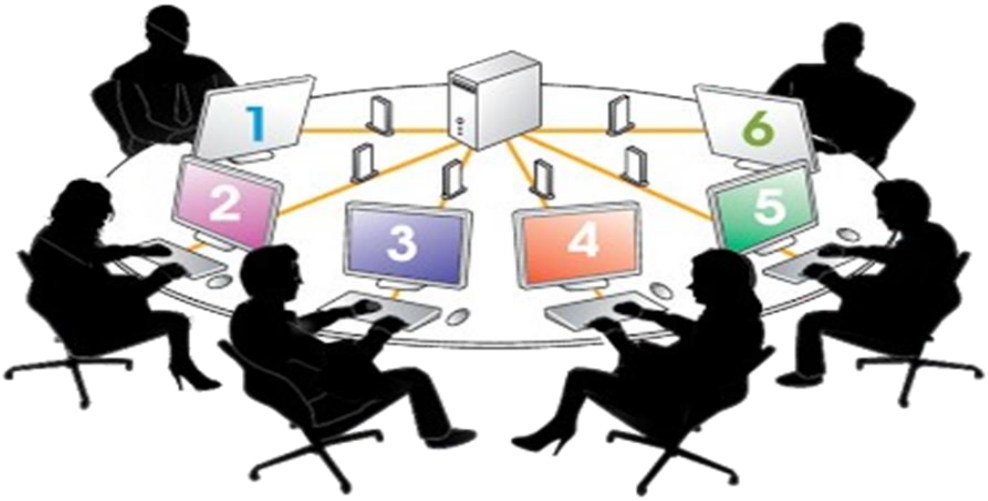
7. Error detection and response

- An error may occur in CPU, in I/O devices or in the memory hardware.
- Following are the major activities of an operating system with respect to error handling –
  - ✓ The OS constantly checks for possible errors.
  - ✓ The OS takes an appropriate action to ensure correct and consistent computing.



8. Accounting

- Keeping a track of which users are using how much and what kinds of computer resources can be used for accounting or simply for accumulating usage statistics.
- Usage statistics is used to reconfigure the system to improve computing services.



9. Protection & Security

- Protection involves ensuring that all accesses to system resources is controlled.
- To make a system secure, the user needs to authenticate himself or herself to the system.

Types of Operating Systems (OS)

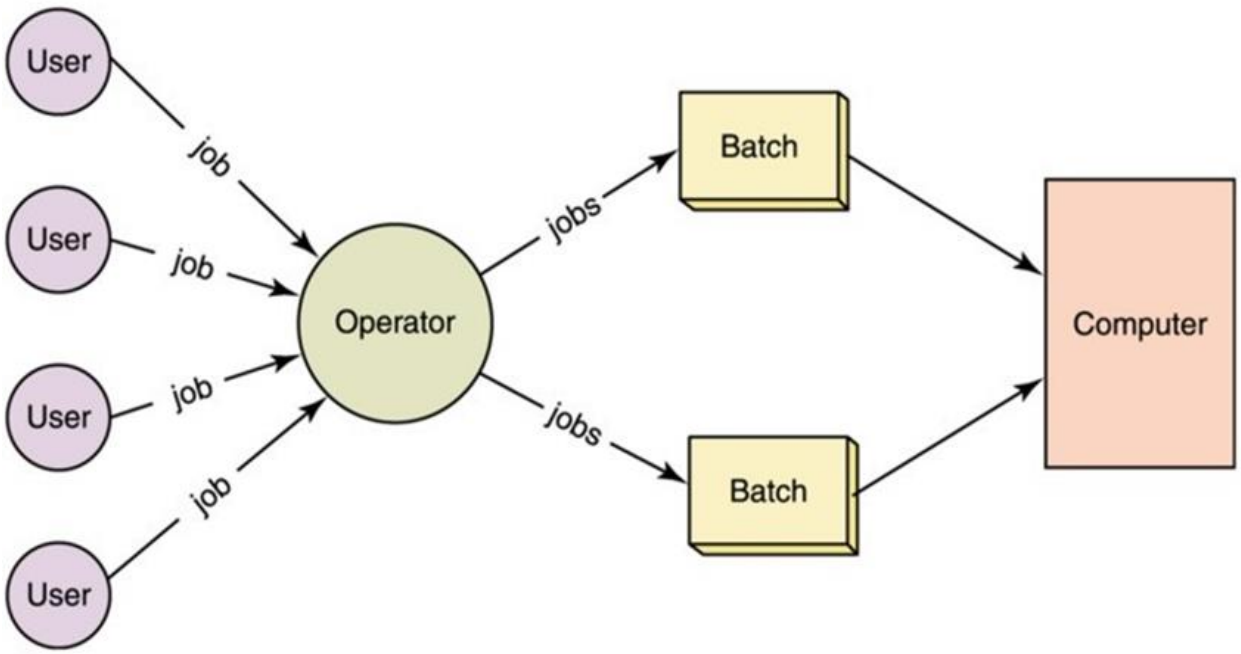
1. Mainframe operating systems

- ❖ OS found in room sized computers which are still found in major corporate data centers.
- ❖ They offer three kinds of services:

1. Batch OS
2. Transaction processing
3. Timesharing

- ❖ Examples: OS/390, OS/360.

Batch OS – processes routine jobs without any interactive user presents i.e. claim processing in insurance

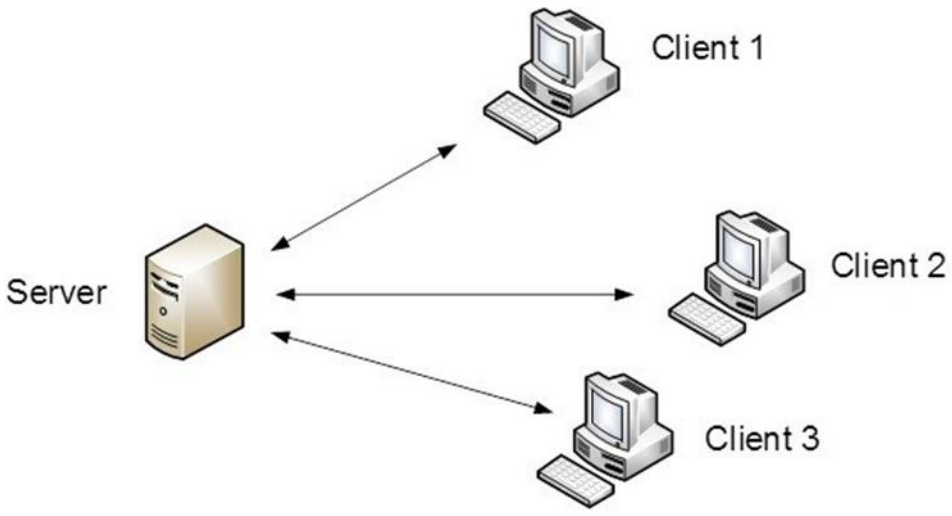


Transaction processing – handles large numbers of small processes i.e. cheque processing at banks

Timesharing – allows multiple remote users to run their jobs at once i.e. querying a database, airline booking system

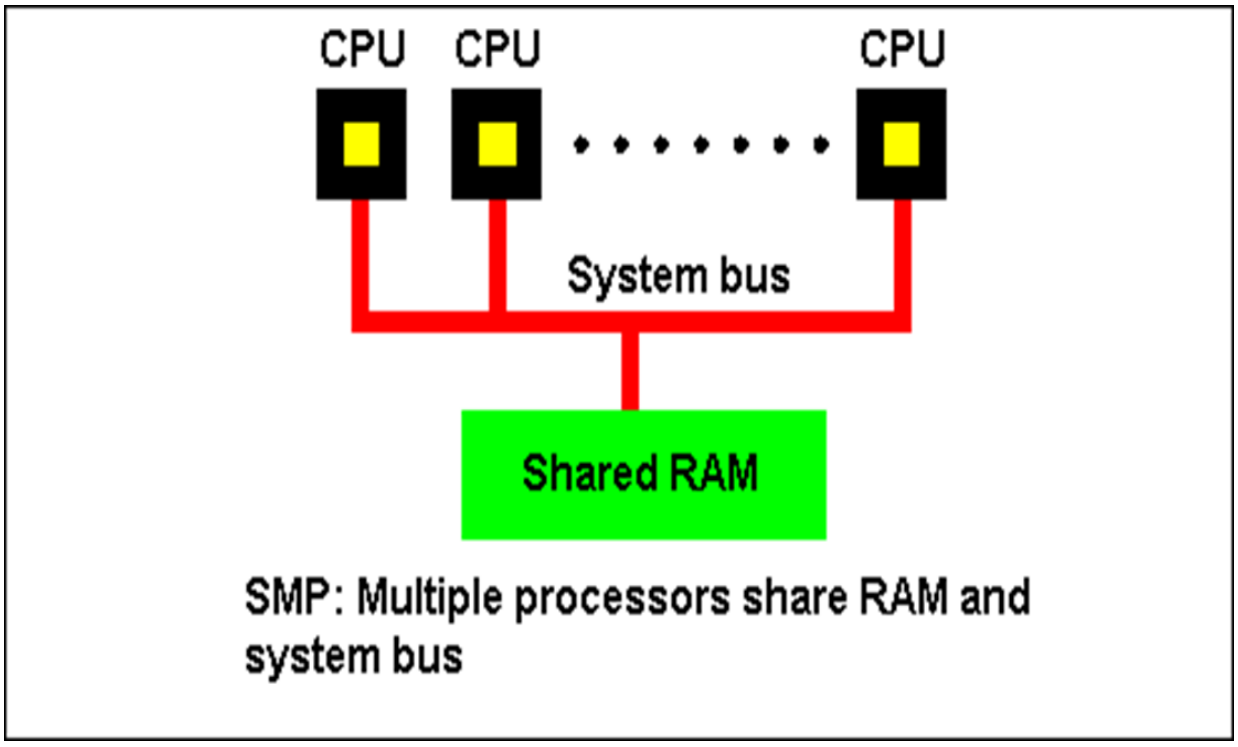
2. Server operating systems

- This OS runs on servers which are very large PC, workstations or even mainframes.
- They serve multiple users at once over a network and allow the users to share hardware & software resources.
- It provides print services, file service or web service.
- It handles the incoming requests from clients.
- Examples: Solaris, FreeBSD, and Linux and Windows Server 200x



3. Multiprocessor operating systems

- A computer system consist two or more CPUs is called multiprocessor.
- It is also called parallel computers, multicomputer or multiprocessor.
- They need special OS or some variations on server OS with special features for communication, connectivity and consistency.
- Examples: Windows and Linux.



SMP: Multiple processors share RAM and system bus

4. Personal computer operating systems

- The operating systems installed on our personal computer and laptops are personal OS.
- Job of this OS is to provide good support to single user.
- This OS is widely used for word processing, spreadsheet and internet access.
- Examples: Linux, Windows vista and Macintosh.

5. Handhelds computer operating systems

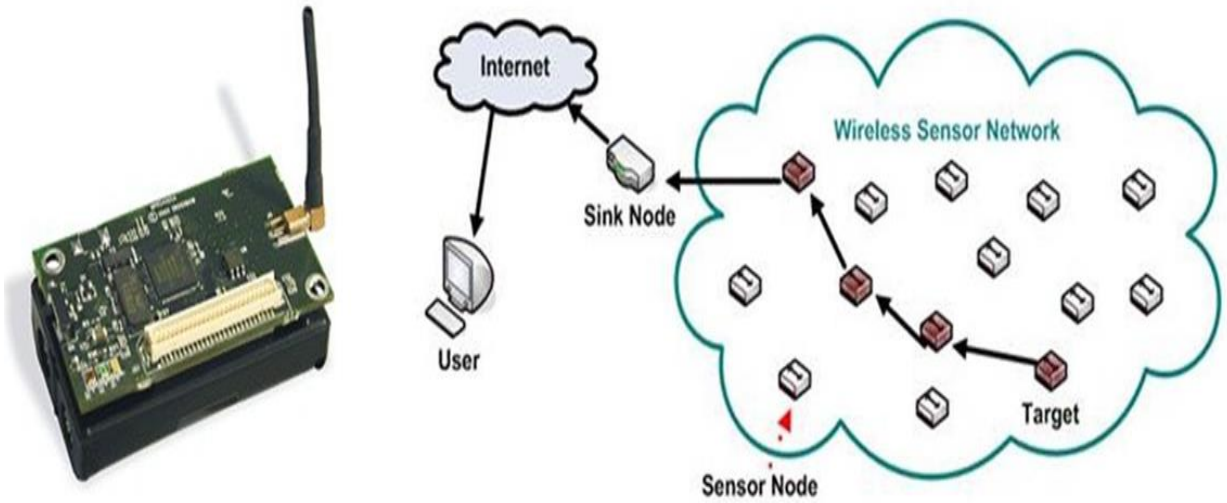
- A handheld computer or PDA (Personal Digital Assistant) is small computer that fit in a Pocket and perform small number of functions such as electronic address book, memo pad.
- The OS runs on these devices are handheld OS.
- These OS also provides ability to handle telephony, digital photography and other functions.
- Examples: Symbian OS, Palm OS.

6. Embedded operating systems

- This OS is installed in ATMs, printers, calculators and washing machine.
- It runs on the computer that control devices.
- It neither allow to download new software nor accept user installed software. So there is no need for protection.
- Examples: QNX, VxWorks.


7. Sensor node operating systems

- Network of tiny sensor nodes are being developed for numerous purpose.
- Each nodes are tiny computers with a CPU, RAM, ROM and one or more environmental sensors.
- The OS installed in these nodes are sensor node OS.
- They communicate with each other and with base station using wireless communication.
- These sensor network are used to protect area of building, detect fires in forest, measure temperature.
- Examples: TinyOS



8. Real time operating systems

- These systems having time as a key parameter.
- Real time OS has well defined fixed time constraints.
- Processing must be done within defined time constraints otherwise system fails.
- Two types of real time OS:
  - ✓ Hard real time – missing an occasional deadline can cause any permanent damage. Many of these are found in industrial process control, car engine control system.
  - ✓ Soft real time – missing an occasional deadline does not cause any permanent damage. Used in digital audio, multimedia system.
- Examples: e-Cos

9. Smart card operating systems

- Smallest OS run on smart cards which are credit card sized devices containing CPU chip.
- These OS are installed on electronic payments cards such as debit card, credit card etc.
- They have limited processing power.
- Some smart cards are Java oriented. ROM on smart card holds an interpreter for the JVM – small program.

System calls

What is System calls?

- A system call is a way for programs to interact with the operating system.
- A system call is a mechanism that provides the interface between a process and the operating system.
- A computer program makes a system call when it makes a request to the operating system's kernel.
- It is a programmatic method in which a computer program requests a service from the kernel of the OS.
- System call provides the services of the operating system to the user programs via Application Program Interface(API).
- System calls are the only entry points for the kernel system.

Types of system calls

1. Process Control: This system calls perform the task of process creation, process termination, etc.

- Functions:
  - ✓ End and abort
  - ✓ Load and execute
  - ✓ Create process and terminate process
  - ✓ Wait and signed event
  - ✓ Allocate and free memory

2. File Management: File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc.

- Functions:
  - ✓ Create a file
  - ✓ Delete file
  - ✓ Open and close file
  - ✓ Read, write and reposition
  - ✓ Get and set file attributes

3. Device Management: Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.

- Functions
  - ✓ Request and release device
  - ✓ Logically attach/ detach devices
  - ✓ Get and Set device attributes

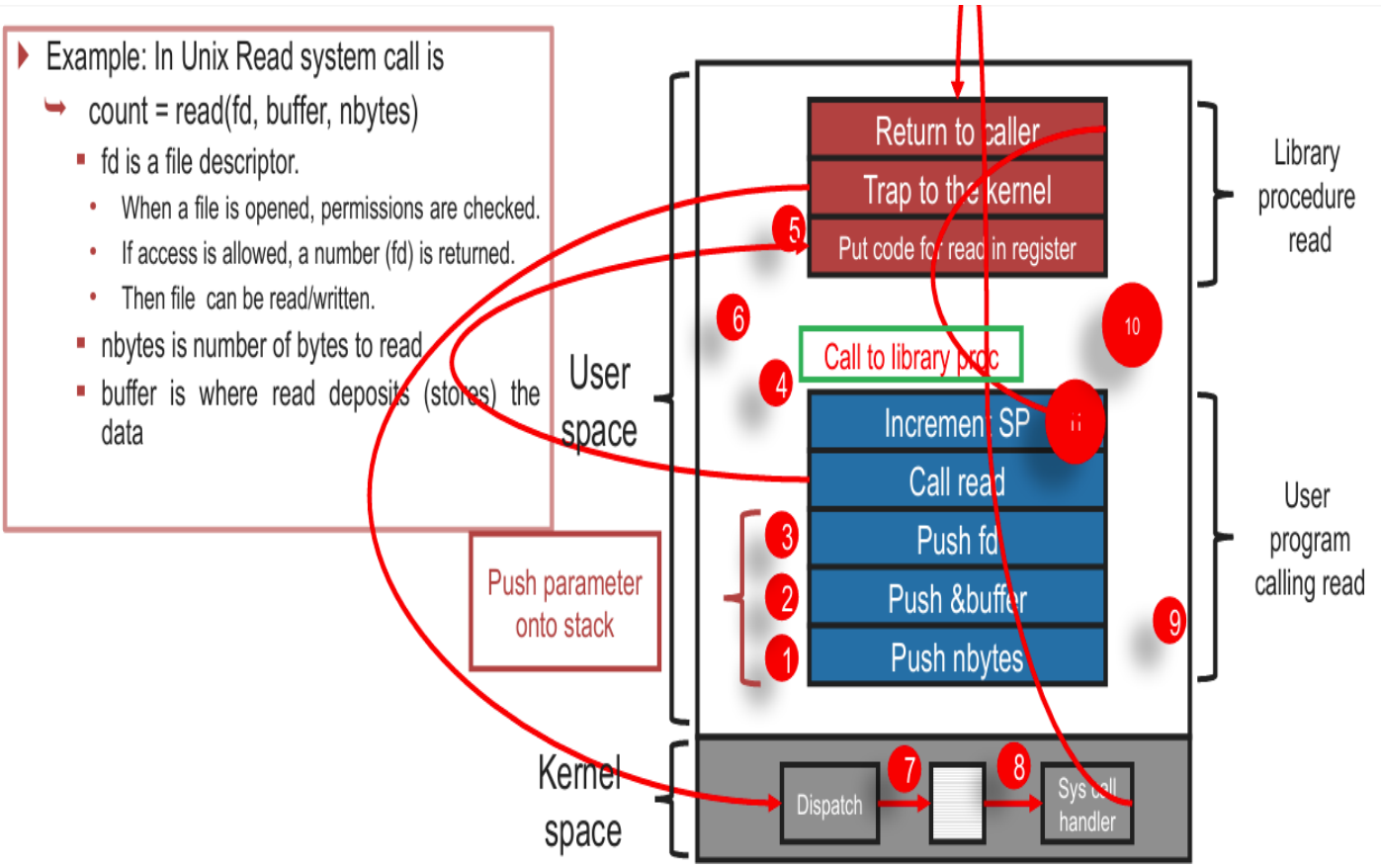
4. Information Maintenance: It handles information and its transfer between the OS and user program.

- Functions:
  - ✓ Get or set time and date
  - ✓ Get process and device attributes

5. Communication: These types of system calls are specially used for interprocess communications (IPC).

- Functions:
  - ✓ Create, delete communications connections
  - ✓ Send, receive message
  - ✓ Help OS to transfer status information
  - ✓ Attach or detach remote devices

Example of system calls (Read system call)



count = read(fd, buffer, nbytes);

- **fd** → File descriptor (an integer returned by open())
- **buffer** → Memory location where data will be stored
- **nbytes** → Number of bytes to read
- **count** → Number of bytes actually read (returned by OS)

Steps in the Diagram

User Space (Application side)

1. **Push parameters onto stack**
   - ✓ The program pushes arguments (fd, buffer, nbytes) onto the process stack.
2. **Call read()**
   - ✓ The program calls the **C library function read()**, which is not the actual OS function but a wrapper.
3. **Library procedure read()**
   - ✓ This library code prepares the request for the OS.
4. **Trap to the Kernel**
   - ✓ A special CPU instruction (**trap**) switches the CPU from **User Mode → Kernel Mode**.
   - ✓ Control is passed to the OS.

Kernel Space (Operating System side)

5. **System Call Handler**
   - ✓ The kernel identifies which system call (here read) was invoked.
   - ✓ Validates file descriptor and permissions.

6. **Dispatch**
   - ✓ The kernel dispatches control to the appropriate device driver or file system handler.
7. **Execution**
   - ✓ The OS performs the actual read:
     - Reads from file/device
     - Copies data into the user's buffer
8. **Return to Kernel**
   - ✓ The number of bytes read (count) is placed in a register.

Return to User Space

9. **Return from System Call**
   - ✓ Control switches back from Kernel Mode → User Mode.
10. **Return to Caller**
    - ✓ The program resumes execution with the result of the system call (count).

Note:

- ✓ User program cannot access hardware directly → it calls read().
- ✓ The OS kernel takes over, reads from the device/file, and returns the data into the program's buffer.
- ✓ System calls act like a **bridge between user programs and OS kernel**.

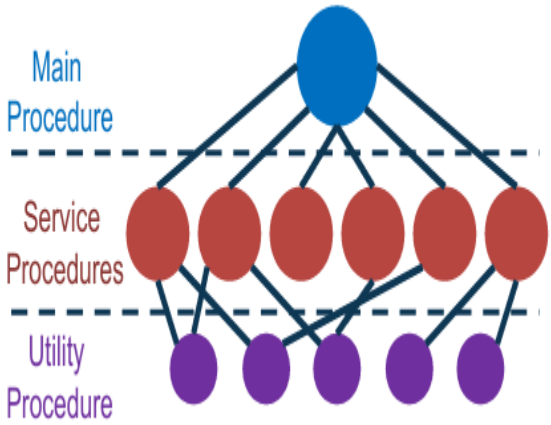
Operating Systems (OS) structure

Operating Systems (OS) structure

1. Monolithic systems
2. Layered systems
3. Microkernel
4. Client-server model
5. Virtual machines
   I. VM/370
   II. Virtual machines rediscovered
   III. The java virtual machine
6. Exokernels

1. Monolithic systems

- ✓ The entire OS runs as a single program in kernel mode.
- ✓ OS is written as a collection of procedures, linked together into a single large executable binary program.
- ✓ Each procedure has well defined interface in terms of parameter and results, and each one is free to call any other one
- ✓ A main program that invoke the requested service procedure.
- ✓ A set of service procedures that Carry out the system calls.
- ✓ A set of utility procedures that Help the service procedure.



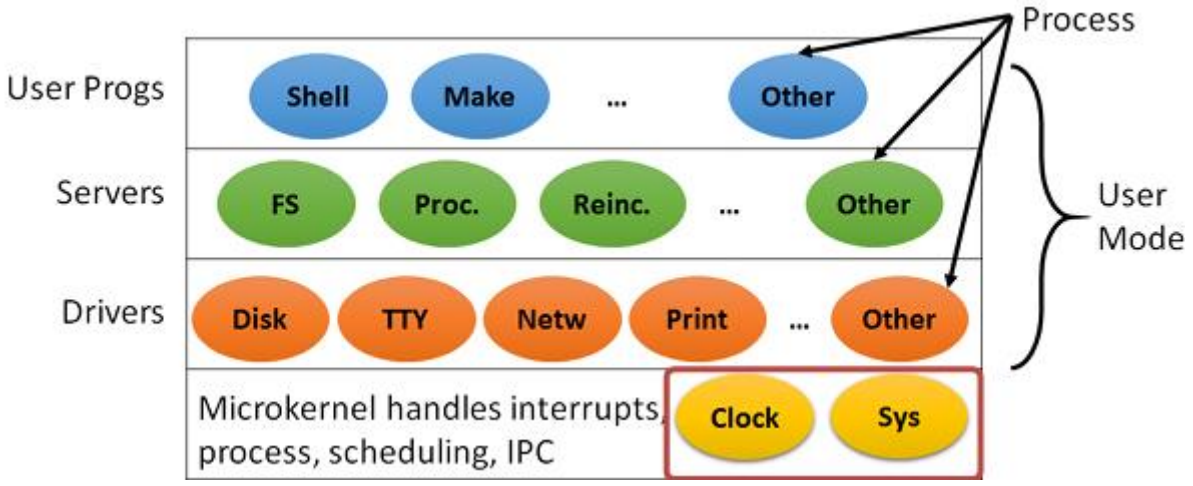
2. Layered Systems

- In this system, the OS is organized as a hierarchy of layers.
- The first system constructed in this way was THE system.

Layer	Function	Description
5	Operator	Operator was located
4	User programs	User programs were found
3	Input / Output management	Takes care of managing the I/O devices. Buffering the information
2	Operator-process communication console (i.e. user).	Handles communication between each process and the operator
1	Memory and drum management	Did the memory management. Allocated space for process in main memory and on a 512K word drum used for holding parts of processes for which there was no room in memory.
0	Processor allocation and multi-programming	Provided the basic multiprogramming of the CPU. Dealt with allocation of the processor, switching between processes when interrupts occurred or timers expired.

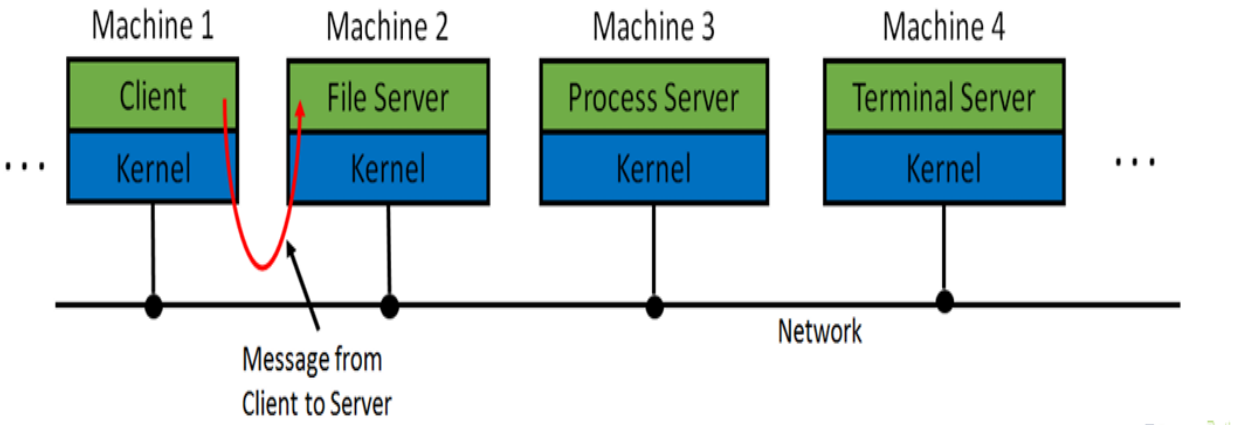
3. Microkernel

- In layered approach, the designer have choice where to draw the kernel and user mode boundary.
- It is better to put as little as possible in kernel mode because bugs in the kernel can bring down the system instantly.
- The microkernel design provides high reliability by splitting OS up into small well defined modules, only one module run in kernel and rest of all run in user mode.
- As each device driver runs as a user process, a bug in audio driver will cause the sound to be stop, but not crash the computer.
- Examples: Integrity, K42, QNX, Symbian and MINIX 3.
- Kernel contains only
  - Sys (Kernel call handler)
  - Clock (because scheduler interact with it)



4. Client-Server model

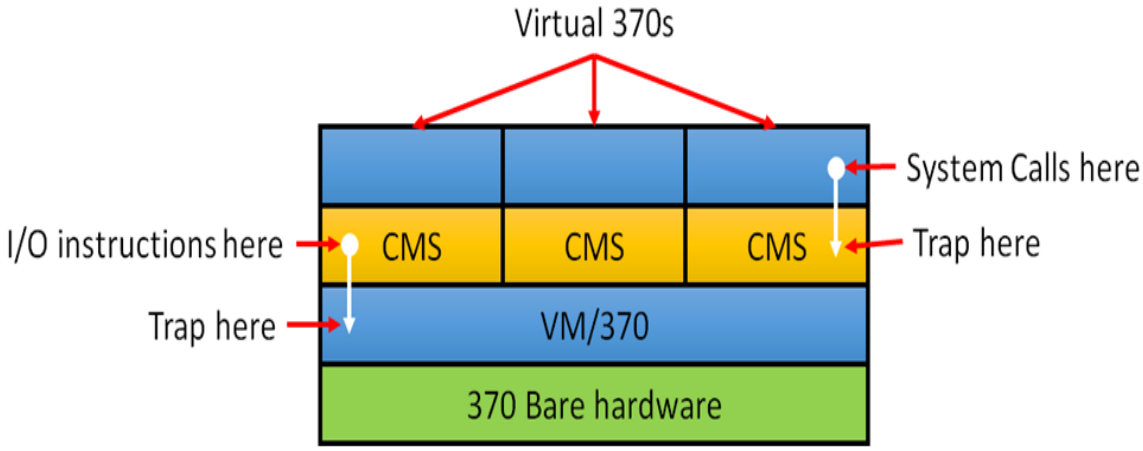
- Processes are divided into two categories
  - ✓ **Servers**: provide services
  - ✓ **Clients:** uses services
- Client and server run on different computers, connected by LAN or WAN and communicate via message passing.
- To obtain a service, a client construct a message saying what it wants and send it to server.
- The server then does the work and send back the answer.



5. Virtual Machines

- A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system.
- The initial releases of OS/360 were strictly batch systems.
- But many users wanted to be able to work interactively at a terminal, so OS designers decided to write timesharing systems for it.
- Types of Virtual machines are:
  i. VM/370
  ii. Virtual Machines Rediscovered
  iii. The Java Virtual Machine

i. VM/370

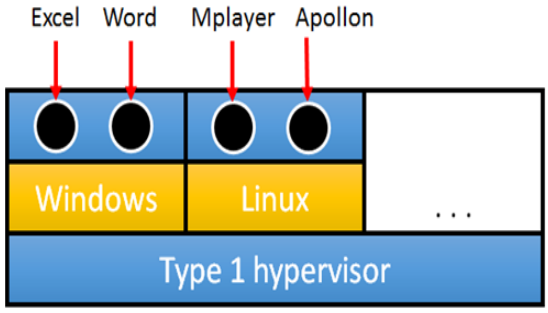


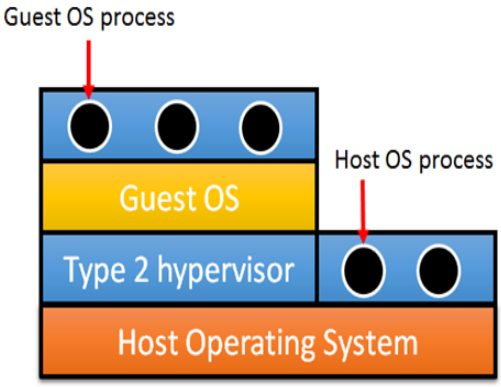
- ✓ Virtual machine monitor run on the bare hardware and does the multiprogramming.
- ✓ Each virtual machine is identical to the true hardware; each one can run any OS (may be different) that will run directly on the bare hardware.
- ✓ On VM/370, some run OS/360 while the others run single user interactive system called CMS

(Conversational Monitor System) for interactive time sharing users.

- ✓ When CMS program executed a system call, a call was trapped to the operating system in its own virtual machine, not on VM/370.
- ✓ CMS then issued the normal hardware I/O instruction for reading its virtual disk or whatever was needed to carry out the call.
- ✓ These I/O instructions were trapped by VM/370 which then performs them.

ii Virtual Machines Rediscovered



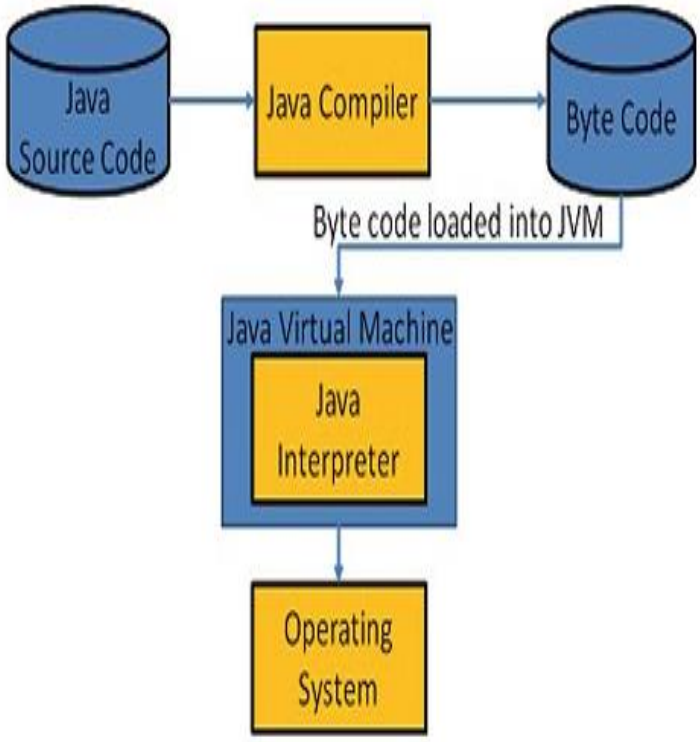


- ✓ Companies can run their mail servers, web servers, FTP servers and other servers on the same machine without having a crash of one server bring down the rest.
- ✓ Web hosting company offers virtual machines for rent, where a single physical machine can run many virtual machines; each one appears to be a complete machine.
- ✓ Customers who rent a virtual machine can run any OS or software.

- ✓ Another use of virtualization is for end users who want to be able to run two or more operating systems at the same time, say Windows and Linux.
- ✓ Type 1 hypervisors runs on the bare hardware.
- ✓ Type 2 hypervisors run as application programs on top of Windows, Linux, or some other operating system, known as the host operating system.

iii Java Virtual Machine

- ✓ When Sun Microsystems invented the Java programming language, it also invented a virtual machine called the JVM (Java Virtual Machine).
- ✓ The Java compiler produces code for JVM, which then typically is executed by a software JVM interpreter.
- ✓ The advantage is that the JVM code can be shipped over the internet to any computer that has a JVM interpreter and run there.

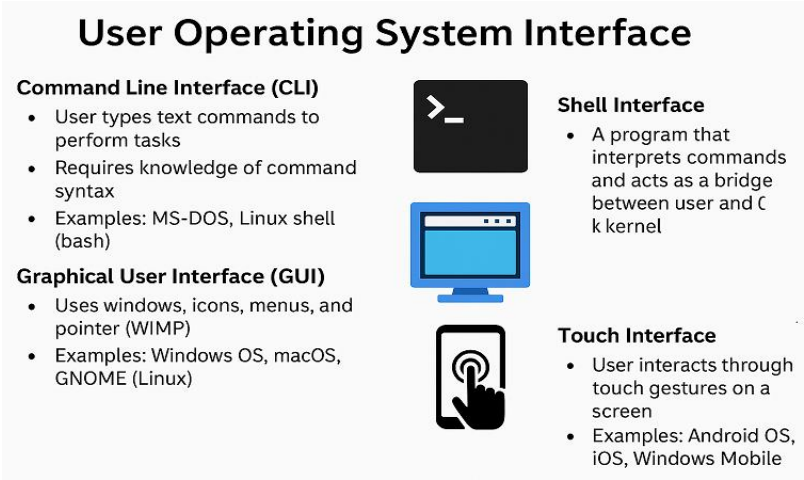


6. Exokernels

- ✓ Rather than cloning (copying) the actual machine, another strategy is partitioning it (giving each user a subset of the resource).
- ✓ For example one virtual machine might get disk blocks 0 to 1023, the next one might get block 1024 to 2047, and so on.
- ✓ Program running at the bottom layer (kernel mode) is called the exokernel.
- ✓ Its job is to allocate resources to virtual machines and then continuously check to make sure no machine is trying to use somebody else's resources.
- ✓ The advantage of the exokernel scheme is that it saves a layer of mapping.

User Operating System Interface

User Operating System Interface



❖ The **User Interface (UI)** is the part of the OS that allows the user to interact with the computer.

❖ Common interfaces:

1. **Command Line Interface (CLI)**

   ✓ Text-based interface.
   ✓ User types commands (e.g., dir, ls, copy).
   ✓ Advantage: Powerful for experts.
   ✓ Disadvantage: Hard for beginners.
   ✓ **Example**: MS-DOS, UNIX shell.

2. **Graphical User Interface (GUI)**

   ✓ Provides **windows, icons, menus, and pointers (WIMP model)**.
   ✓ Easy to use; supports multitasking.
   ✓ **Examples**: Windows OS, macOS, Ubuntu desktop.

3. **Shell**

   - ✓ A **program that interprets commands** from user to OS.
   - ✓ Types:
     - ➢ **Command-line shells** (Bash, PowerShell).
     - ➢ **Graphical shells** (Windows Explorer, GNOME shell).

4. **Touch Interface**
   - Designed for **mobile & tablet devices**.
   - Users interact using **gestures, taps, swipes, pinch-zoom**.
   - **Examples**: Android OS, iOS.

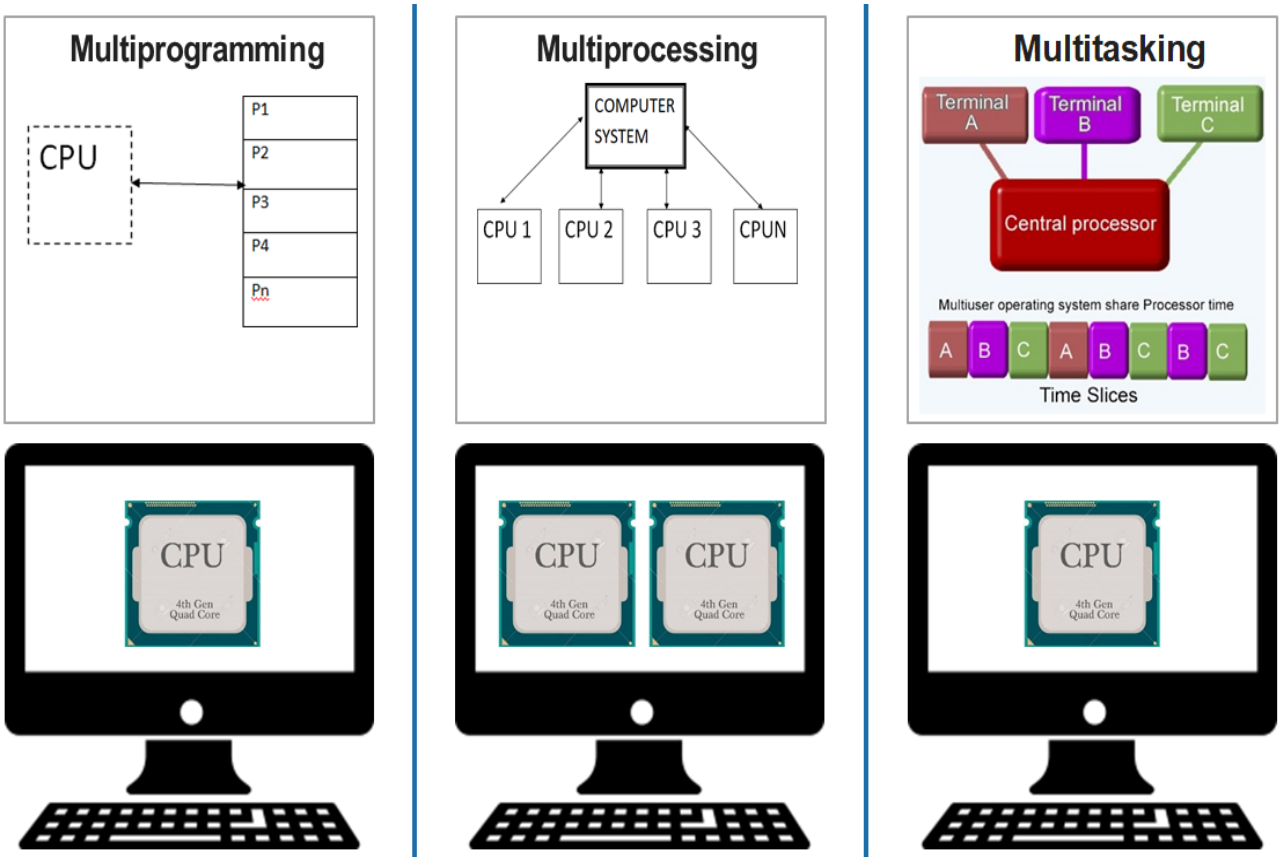
Interprocess Communication (IPC)

Processes often need to **communicate and synchronize** with each other. IPC provides mechanisms for this.

- **Shared Memory Model**
  - ✓ A region of memory is shared by multiple processes.
  - ✓ Processes read/write directly to this memory.
  - ✓ Very fast, but requires synchronization (semaphores, locks).
- **Message Passing Model**
  - ✓ Processes communicate by sending/receiving messages via the OS.
  - ✓ Safer (no direct memory access), but slower than shared memory.
  - ✓ Examples: send(), receive(), pipes, sockets.

Multiprogramming v/s Multiprocessing v/s Multitasking


OPERATING
SYSTEM
SERVICES
RESOURCES
DESIGN
KERNEL
SERVER
TIME
SOFTWARE
CALL
APPLICATION
HARDWARE
GUI
MANAGEMENT
THREAD
PROCESSOR


Multiprogramming
CPU
P1
P2
P3
P4
Pn
Multiprocessing
COMPUTER SYSTEM
CPU 1
CPU 2
CPU 3
CPUN
Multitasking
Terminal A
Terminal B
Terminal C
Central processor
Multiuser operating system share Processor time
A B C A B C B C
Time Slices
CPU
4th Gen Quad Core

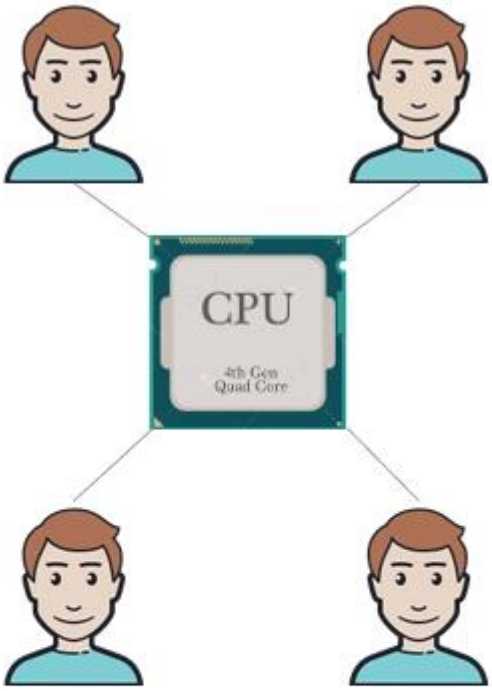
Multiprogramming v/s Multiprocessing v/s Multitasking

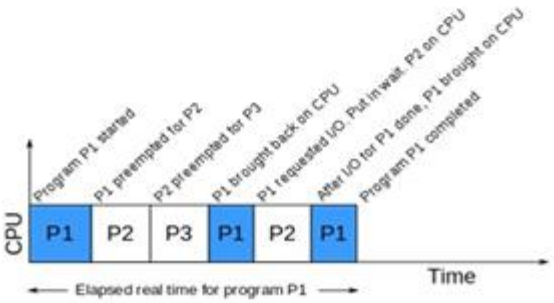
Multiprogramming	Multiprocessing	Multitasking
The concurrent residency of more than one program in the main memory is called as multiprogramming.	The availability of more than one processor per system, which can execute several set of instructions in parallel is called as multiprocessing	The execution of more than one task simultaneously is called as multitasking
Number of processor: one	Number of processor: more than one	Number of processor: one
One process is executed at a time	More than one process can be executed at a time.	One by one job is being executed at a time.

Multitasking is a logical extension of multi programming. The major way in which multitasking differs from multi programming is that multi programming works solely on the concept of context switching whereas multitasking is based on time sharing alongside the concept of context switching.

Time Sharing Operating System

- A time sharing operating system allows many users to share the computer resources simultaneously.
- In other words, time sharing refers to the allocation of computer resources in time slots to several programs simultaneously.
- For example a mainframe computer that has many users logged on to it.
- Each user uses the resources of the mainframe i.e. memory, CPU etc.

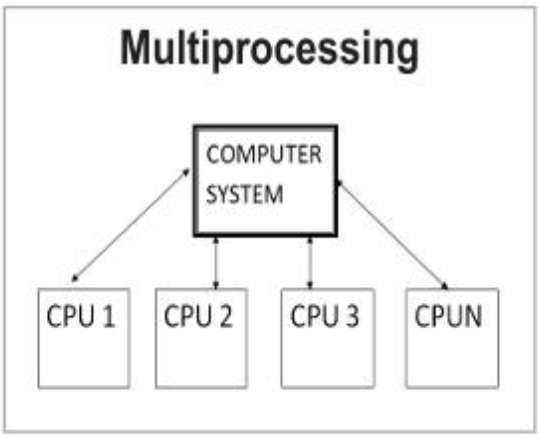




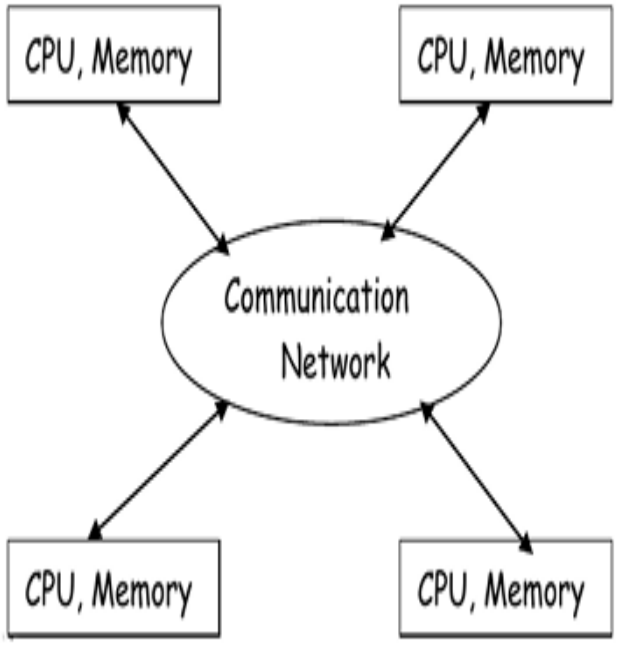
Parallel Processing

Operating System

- Parallel Processing Operating Systems are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously.
- Such systems are multiprocessor systems.
- Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors.



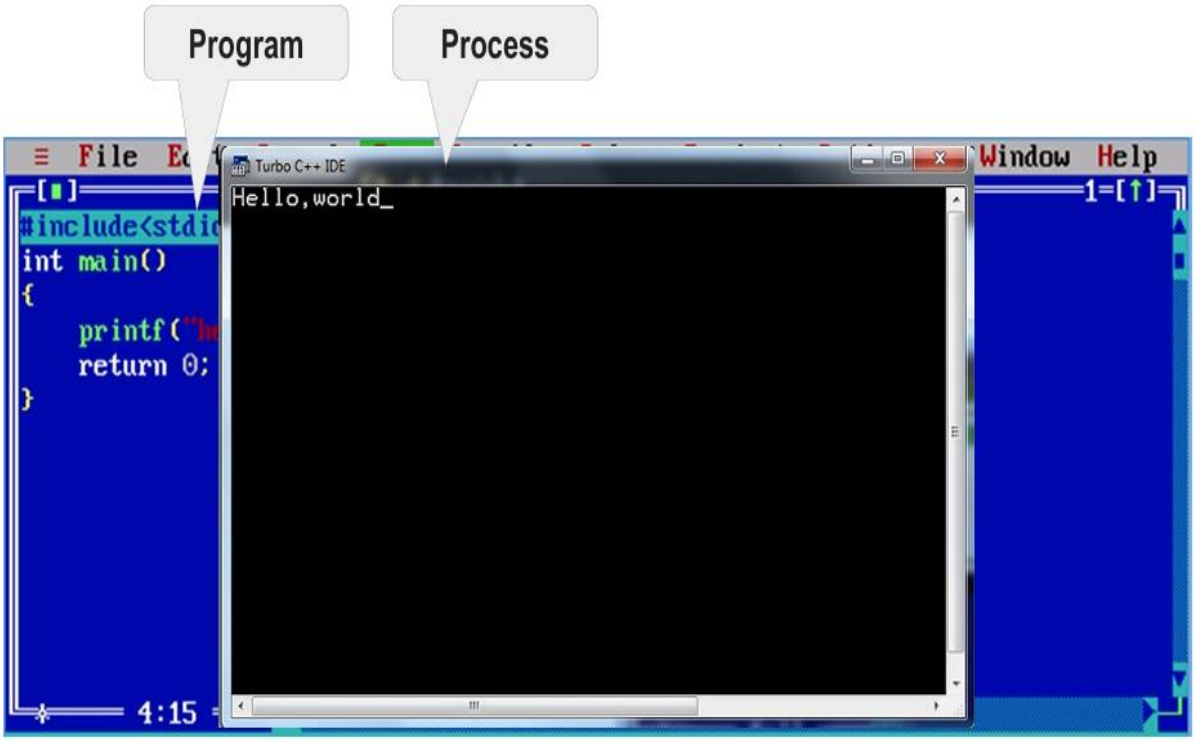
Distributed Operating System



- Distributed Operating System is a model where distributed applications are running on multiple computers linked by communications.
- A distributed operating system is an extension of the network operating system that supports higher levels of communication and integration of the machines on the network.

Process concept

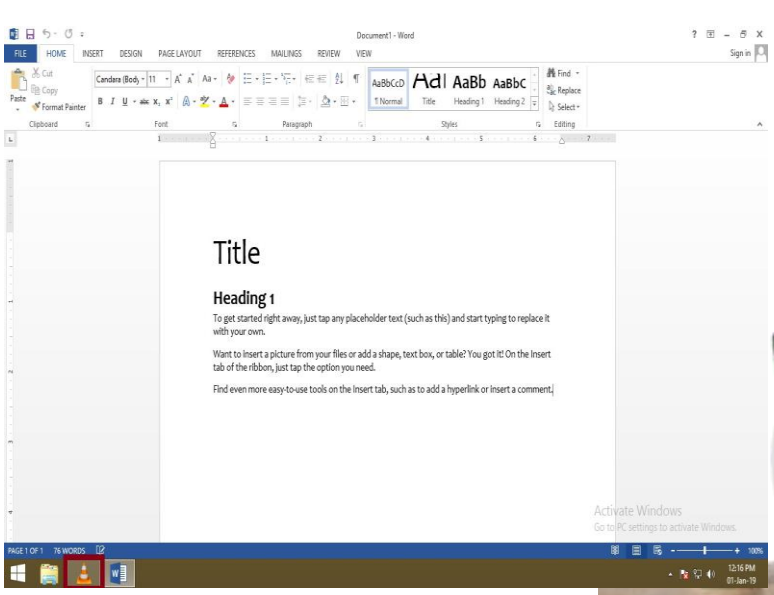
What is Process?

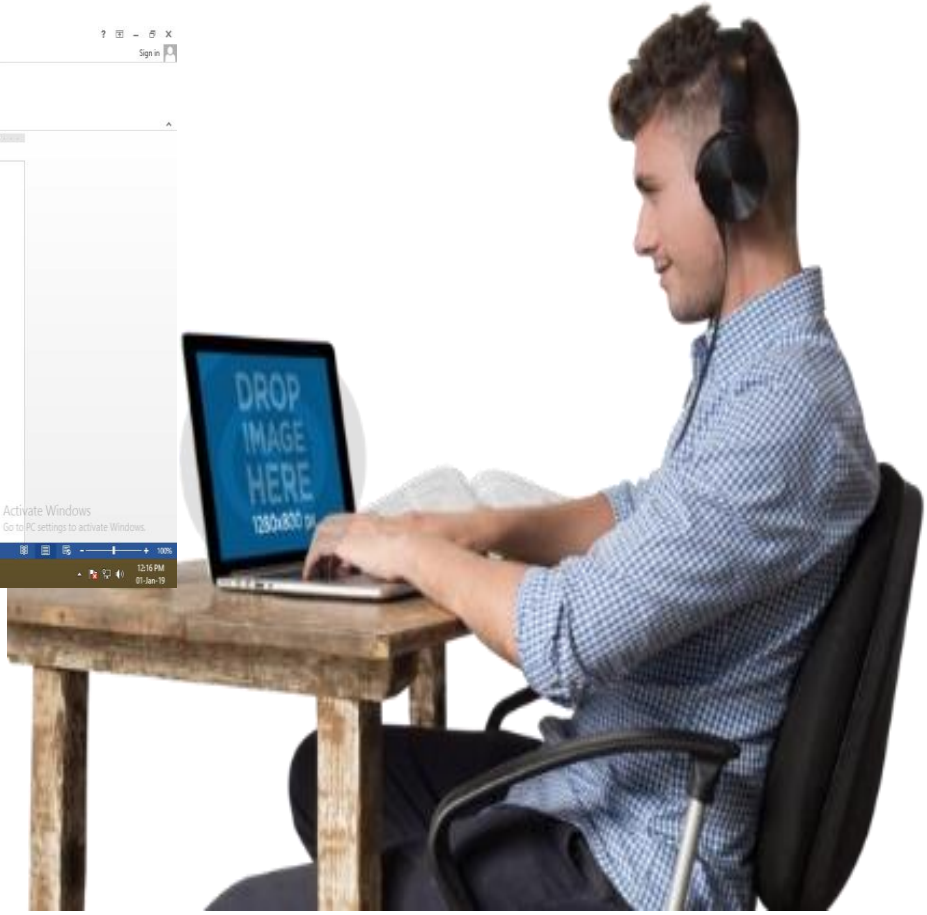


- Process is a program under execution.
- Process is an abstraction of a running program.
- Process is an instance of an executing program, including the current values of the program counter, registers & variables.
- Each process has its own virtual CPU.

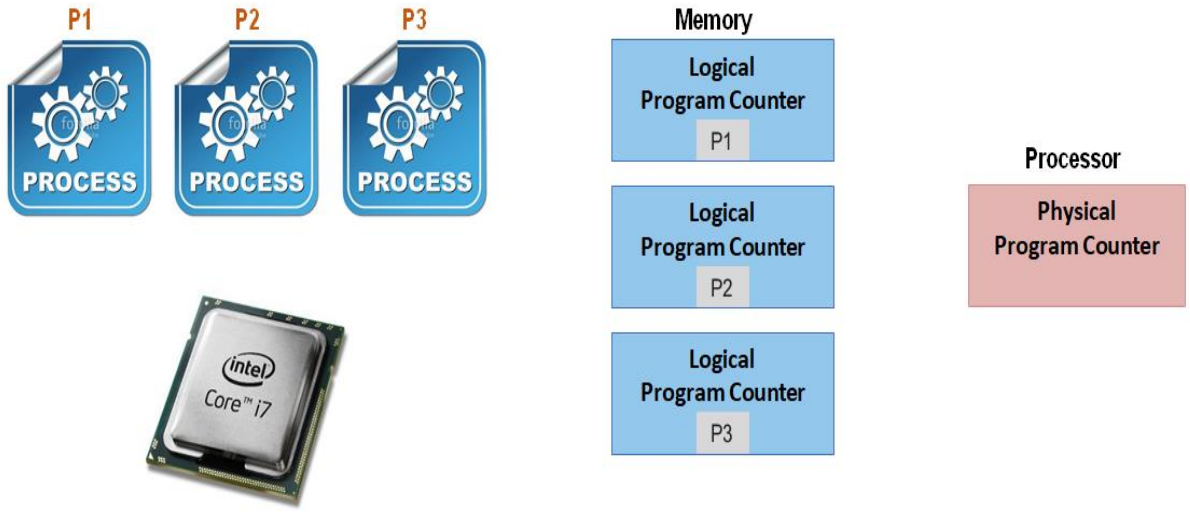
Multiprogramming

- The real CPU switches back and forth from process to process.
- This rapid switching back and forth is called multiprogramming.
- The number of processes loaded simultaneously in memory is called degree of multiprogramming.

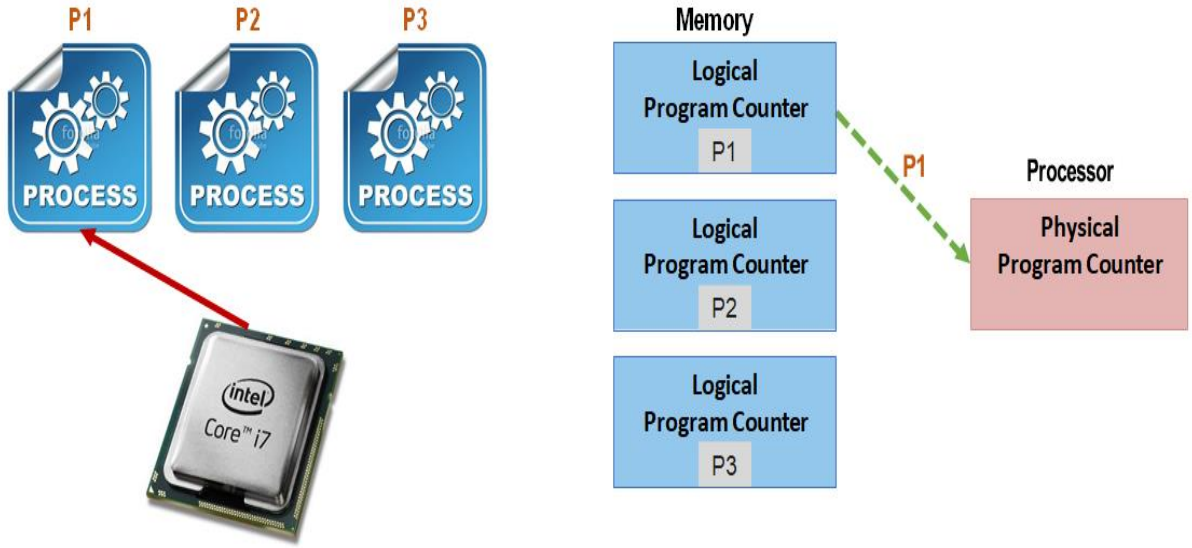




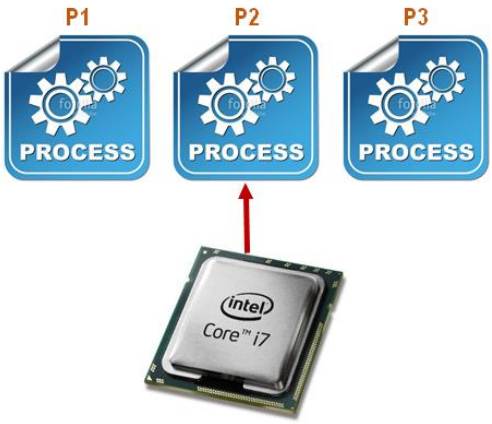
Multiprogramming execution

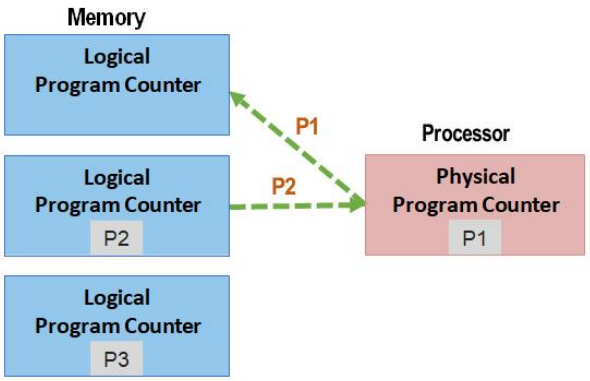


- There are three processes, one processor (CPU), three logical program counter (one for each processes) in memory and one physical program counter in processor.
- Here CPU is free (no process is running).
- No data in physical program counter.

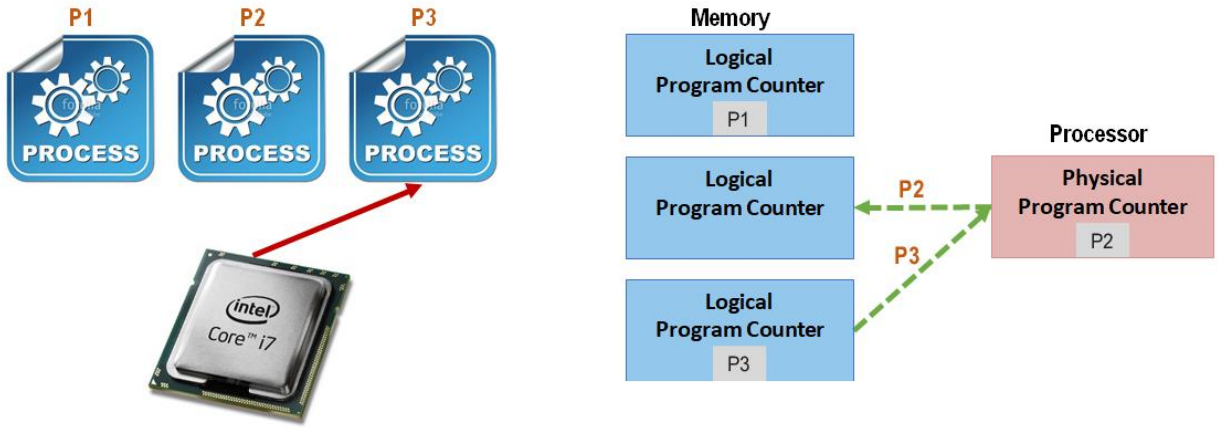


- CPU is allocated to process P1 (process P1 is running).
- Data of process P1 is copied from its logical program counter to the physical program counter.



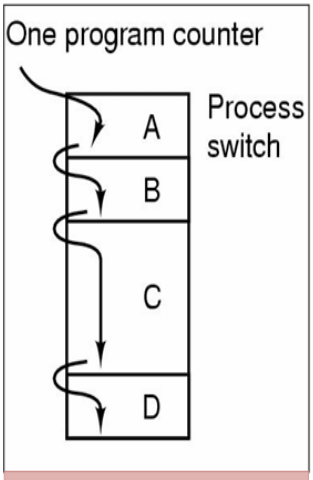


- CPU switches from process P1 to process P2.
- CPU is allocated to process P2 (process P2 is running).
- Data of process P1 is copied back to its logical program counter.
- Data of process P2 is copied from its logical program counter to the physical program counter.

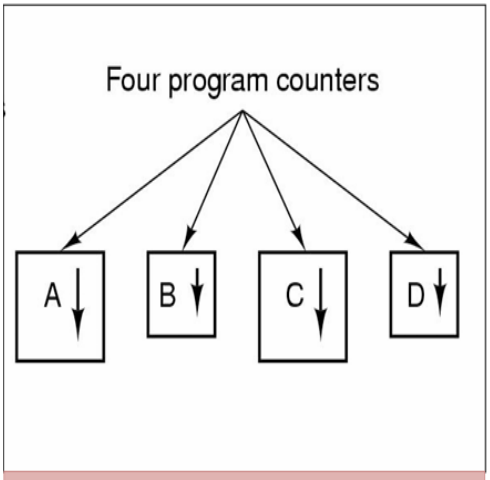


- CPU switches from process P2 to process P3.
- CPU is allocated to process P3 (process P3 is running).
- Data of process P2 is copied back its logical program counter.
- Data of process P3 is copied from its logical program counter to the physical program counter.

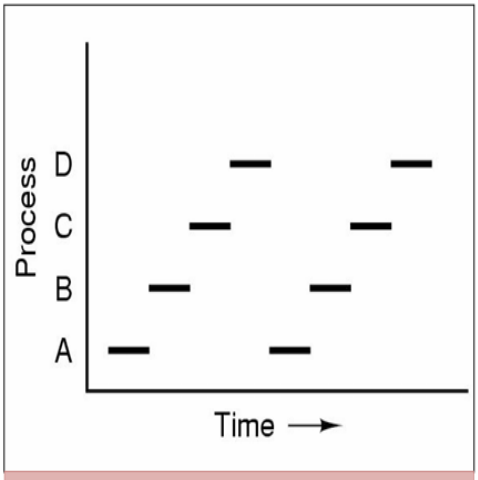
Process Model



Multiprogramming of four programs in memory



Conceptual model of 4 independent, sequential processes, each with its own flow of control (i.e., its own logical program counter) and each one running independently of the other ones.



Over a long period of time interval, all the processes have made progress, but at any given instant only one process is actually running.

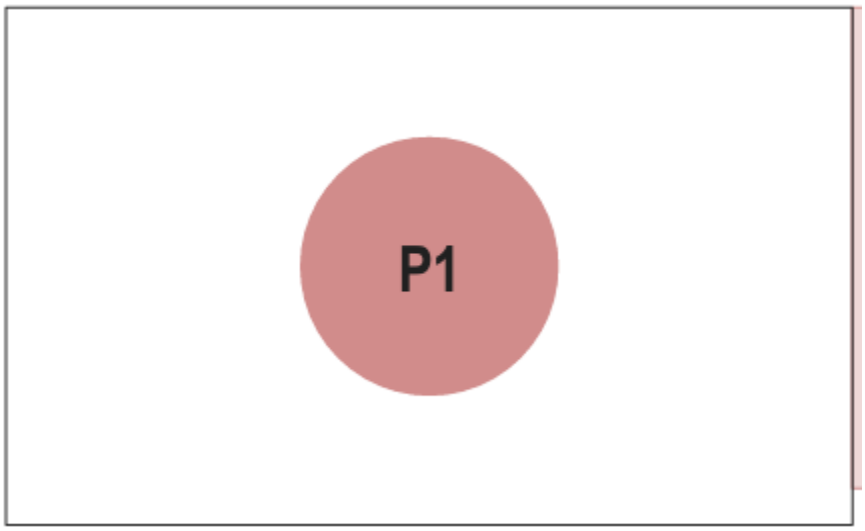
Process Creation

1. System initialization

- ✓ At the time of system (OS) booting various processes are created
- ✓ Foreground and background processes are created
- ✓ Background process – that do not interact with user e.g. process to accept mail
- ✓ Foreground Process – that interact with use

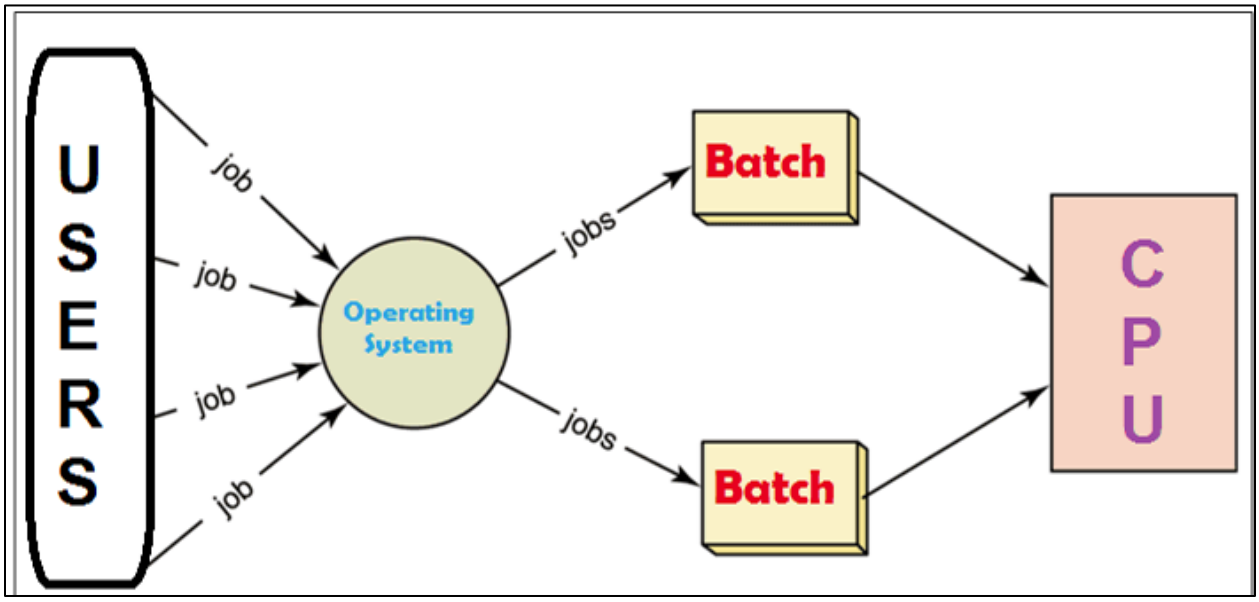
2. Execution of a process creation system call (fork) by running process

- ✓ Running process will issue system call (fork) to create one or more new process to help it.
- ✓ A process fetching large amount of data and execute it will create two different processes one for fetching data and another to execute it.



3. A user request to create a new process

- ✓ Start process by clicking an icon (opening word file by double click) or by typing command.



4. Initialization of batch process

- ✓ Applicable to only batch system found on large mainframe

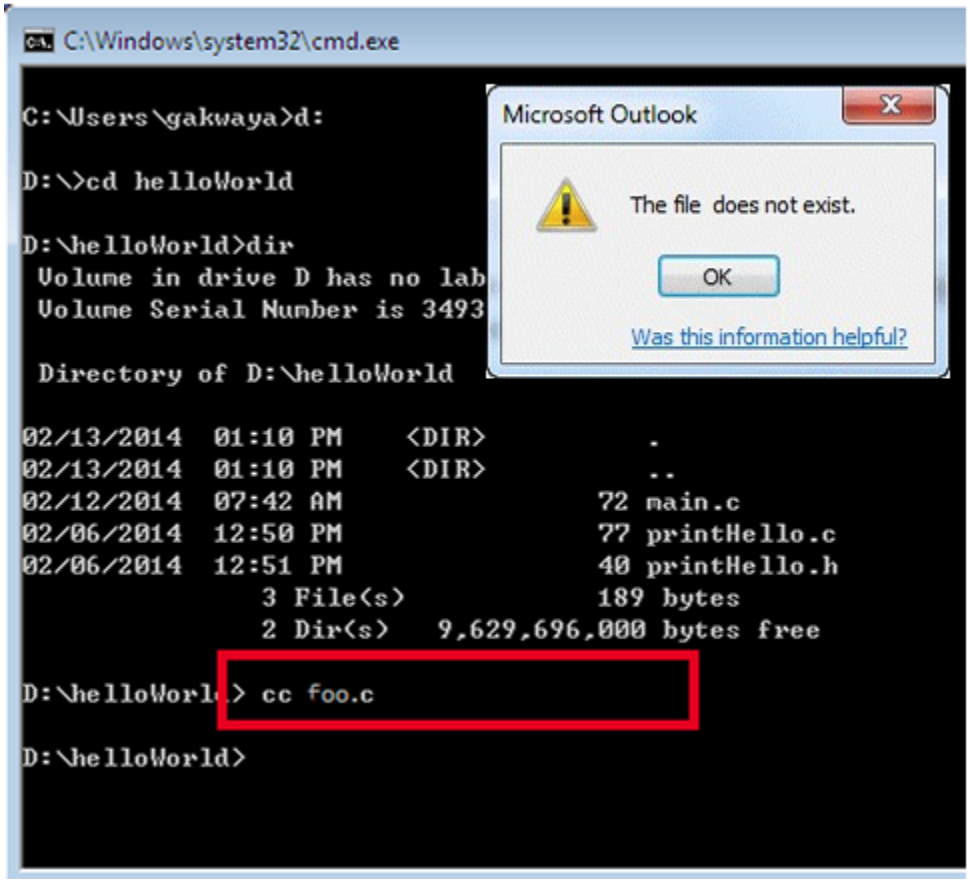
Process Termination

1. Normal exit (voluntary)

- ✓ Terminated because process has done its work.

2. Error exit (voluntary)

- ✓ The process discovers a fatal error e.g. user types the command cc foo.c to compile the program foo.c and no such file exists, the compiler simply exit.

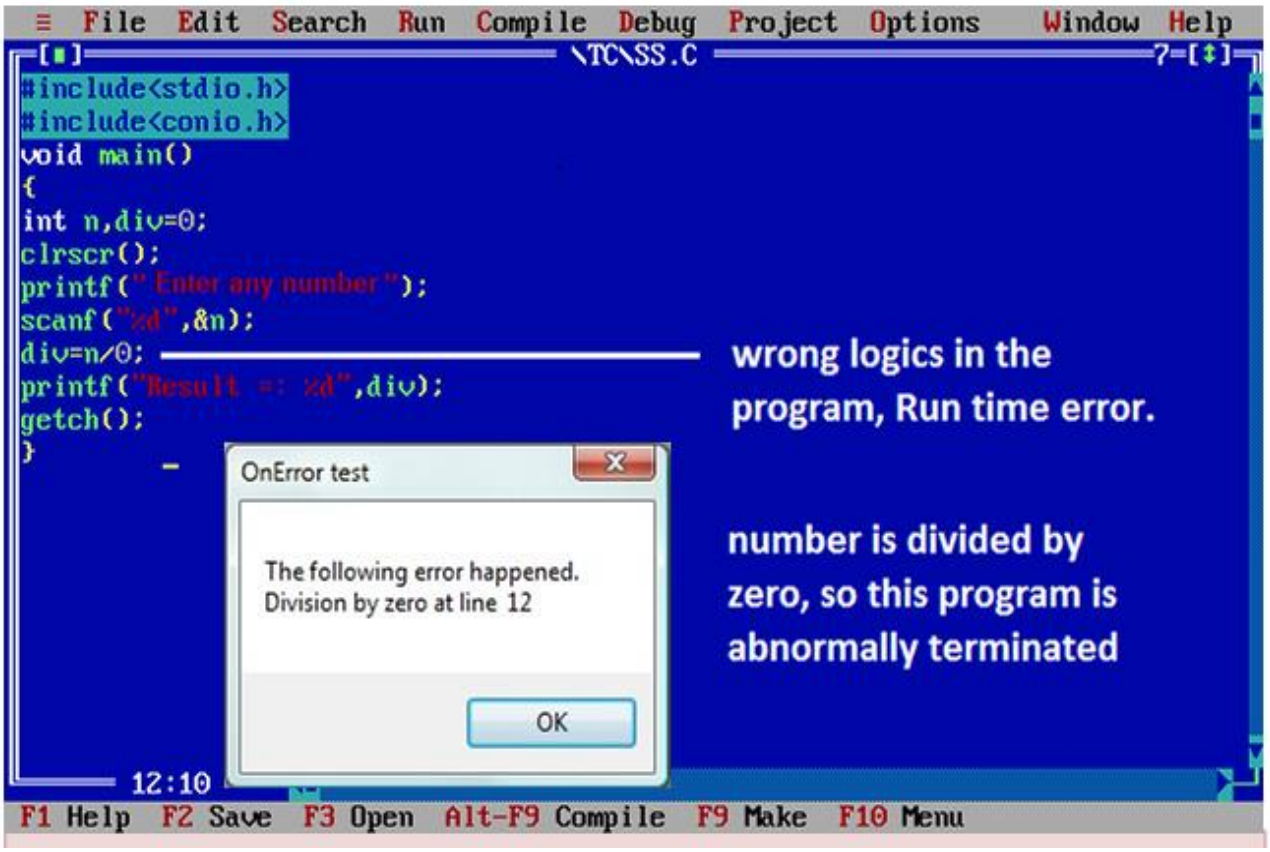


3. Fatal error (involuntary)

- ✓ An error caused by a process often due to a program bug e.g. executing an illegal instruction, referencing nonexistent memory or divided by zero.

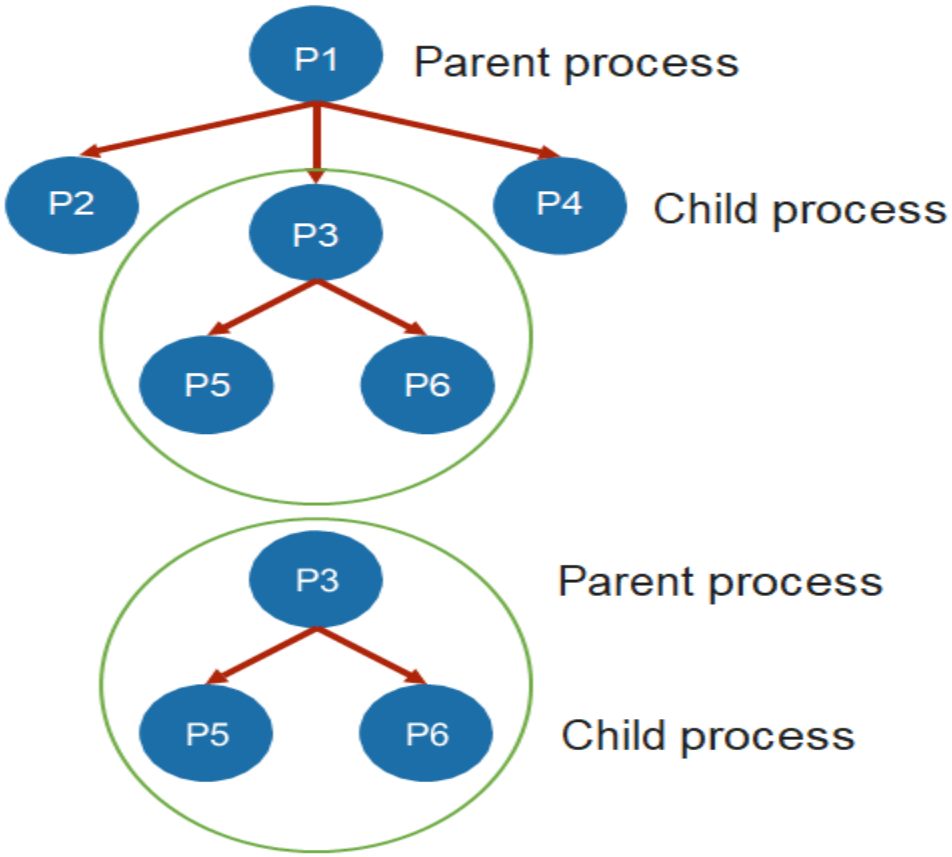
4. Killed by another process (involuntary)

- ✓ A process executes a system call telling the OS to kill some other process using kill system call.



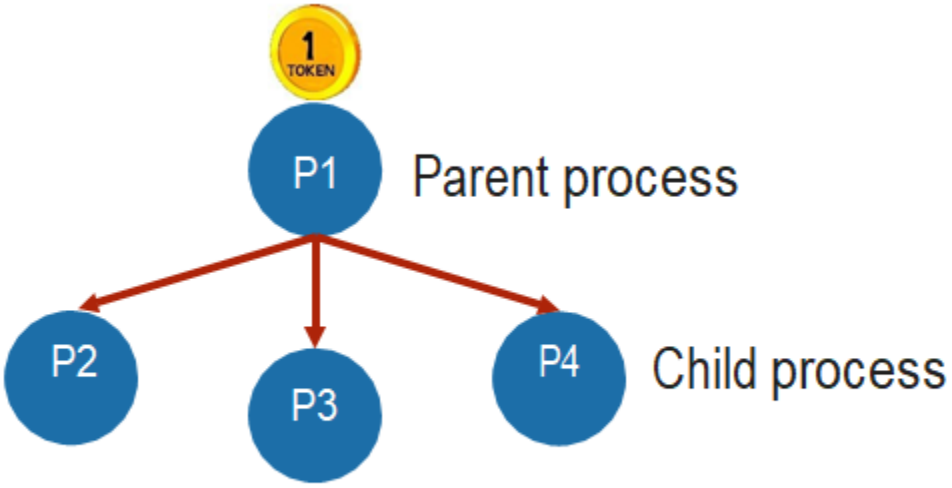
Process Hierarchies

- ✓ Parent process can create child process, child process can create its own child process.
- ✓ UNIX has hierarchy concept which is known as process group
- ✓ Windows has no concept of hierarchy
  - ⍰ All the process as treated equal (use handle concept)



Handle

- ✓ When a process is created, the parent process is given a special token called handle.
- ✓ This handle is used to control the child process.
- ✓ A process is free to pass this token to some other process.



Process states

Running:

Process is actually using the CPU

Ready:

Process is runnable, temporarily stopped to let another process to run

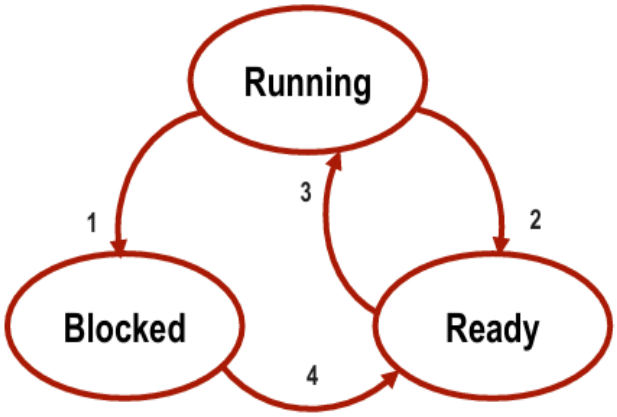
Blocked:

Process is unable to run until some external event happens

Process states transitions

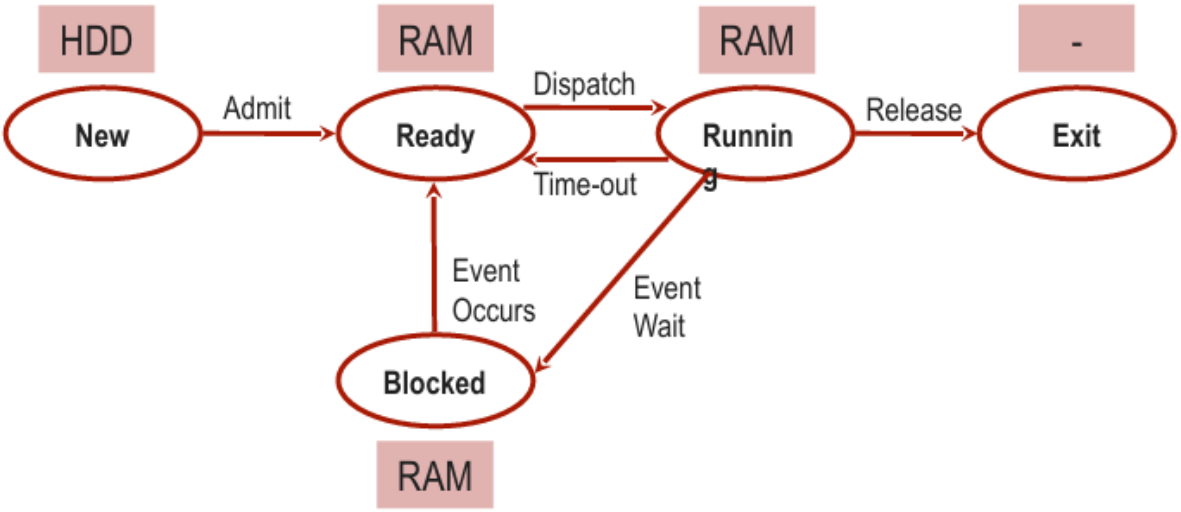
- ✓ **When and how these transitions occur (process moves from one state to another)?**

1. Process blocks for input or waits for an event (i.e. printer is not available)
2. Scheduler picks another process
   - ✓ End of time-slice or pre-emption.
3. Scheduler picks this process
4. Input becomes available, event arrives (i.e. printer become available)



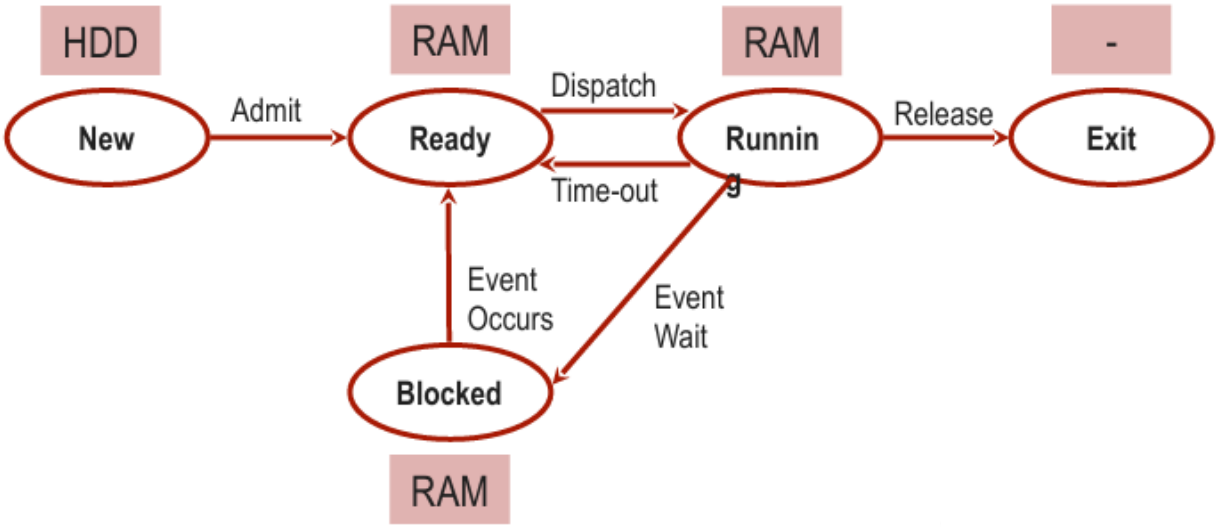
Processes are always either executing (running) or waiting to execute (ready) or waiting for an event (blocked) to occur.

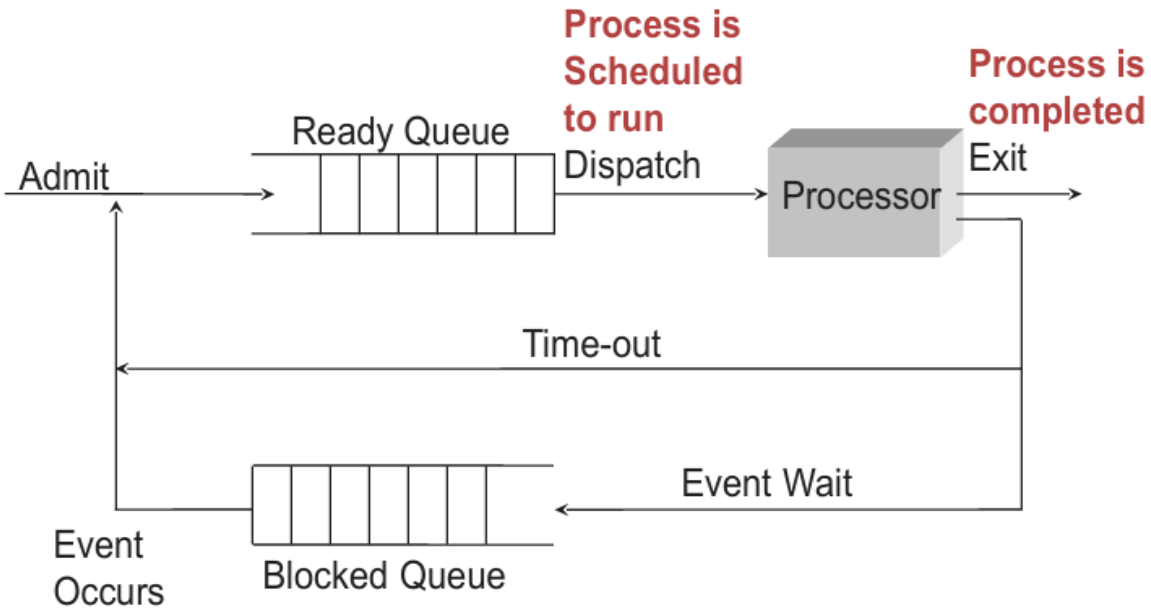
Five State Process Model and Transitions



- ✓ New – process is being created
- ✓ Ready – process is waiting to run (runnable), temporarily stopped to let another process run
- ✓ Running – process is actually using the CPU
- ✓ Blocked – unable to run until some external event happens
- ✓ Exit (Terminated) – process has finished the execution

Queue Diagram



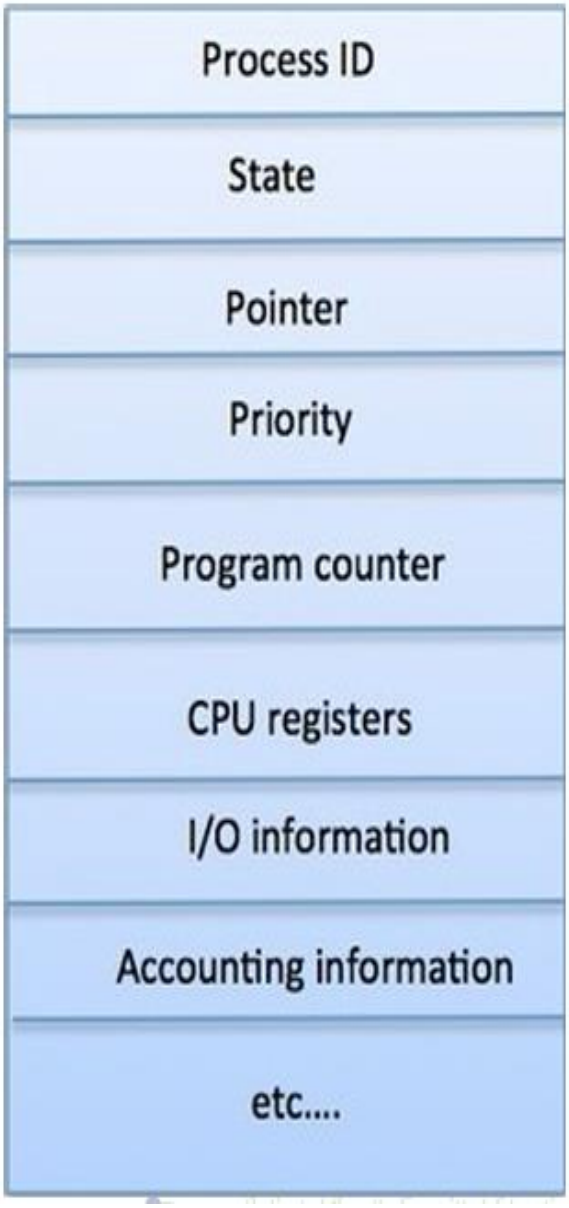


Process Control Block (PCB)

What is Process Control Block (PCB)?

- ✓ A Process Control Block (PCB) is a data structure maintained by the operating system for every process.
- ✓ PCB is used for storing the collection of information about the processes.
- ✓ The PCB is identified by an integer process ID (PID).
- ✓ A PCB keeps all the information needed to keep track of a process.
- ✓ The PCB is maintained for a process throughout its lifetime and is deleted once the process terminates.
- ✓ The architecture of a PCB is completely dependent on operating system and may contain different information in different operating systems.
- ✓ PCB lies in kernel memory space.

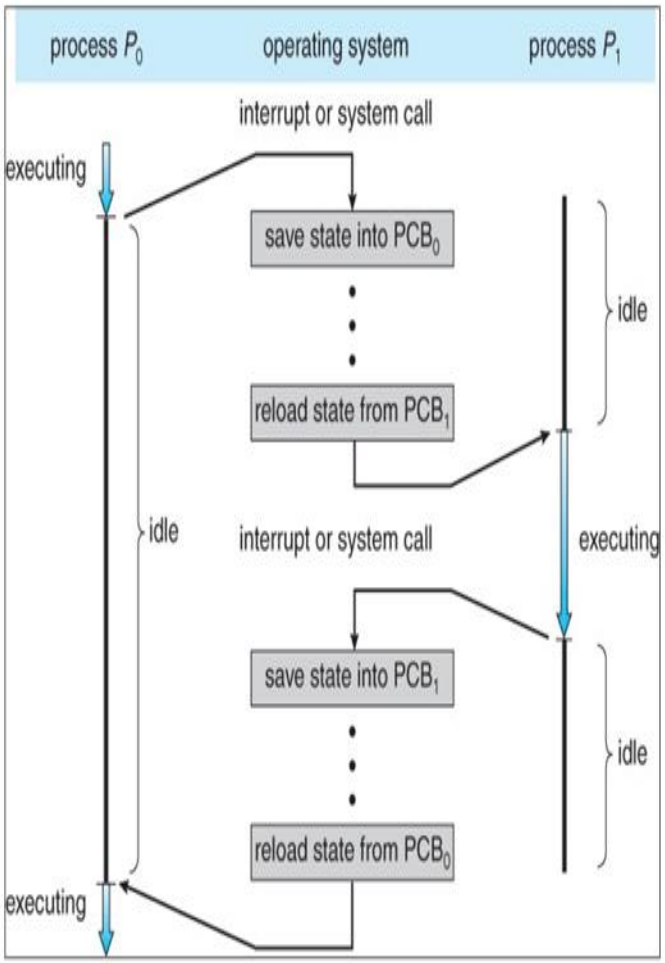
Fields of Process Control Block (PCB)



- ✓ **Process ID -** Unique identification for each of the process in the operating system.
- ✓ **Process State -** The current state of the process i.e., whether it is ready, running, waiting.
- ✓ **Pointer -** A pointer to parent process.
- ✓ **Priority -** Priority of a process.
- ✓ **Program Counter -** Program Counter is a pointer to the address of the next instruction to be executed for this process.
- ✓ **CPU registers -** Various CPU registers where process need to be stored for execution for running state.
- ✓ **IO status information -** This includes a list of I/O devices allocated to the process.
- ✓ **Accounting information -** This includes the amount of CPU used for process execution, time limits etc.

Context switching

- ✓ Context switch means stopping one process and restarting another process.
- ✓ When an event occur, the OS saves the state of an active process (into its PCB) and restore the state of new process (from its PCB).
- ✓ Context switching is purely overhead because system does not perform any useful work while context switch.

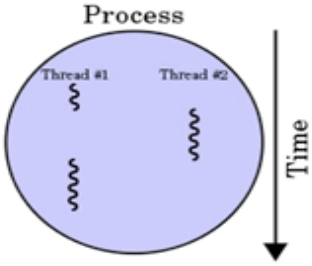


- ✓ **Sequence of action:**
  1. OS takes control (through interrupt)
  2. Saves context of running process in the process PCB
  3. Reload context of new process from the new process PCB
  4. Return control to new process


OPERATING
SYSTEM
SERVICES
RESOURCES
CALL
THREAD
DESIGN
HARDWARE
SOFTWARE
KERNEL
TIME
SERVER
GUI
COMPUTER
APPLICATION

Threads

What is Threads?



✓ Thread is light weight process created by a process.

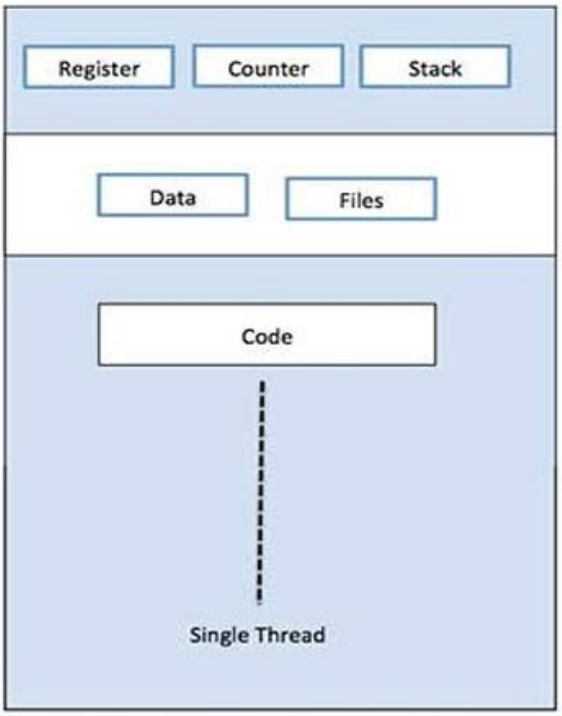
Processes are used to execute large, 'heavyweight' jobs such as working in word, while threads are used to carry out smaller or 'lightweight' jobs such as auto saving a word document.

✓ Thread is a single sequence stream within a process.

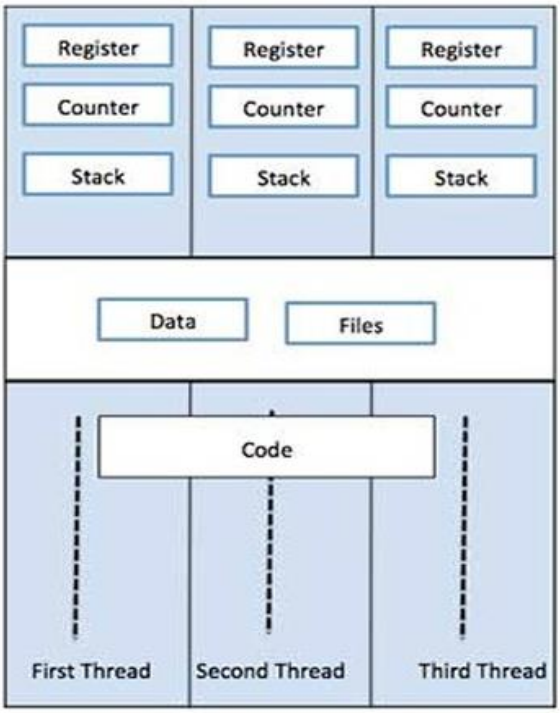
✓ Thread has it own

- program counter that keeps track of which instruction to execute next.
- system registers which hold its current working variables.
- stack which contains the execution history.

Single Threaded Process VS Multiple Threaded Process



Single Process P with single thread



Single Process P with three threads

- ✓ A single-threaded process is a process with a single thread.
- ✓ A multi-threaded process is a process with multiple threads.
- ✓ The multiple threads have its own registers, stack and counter but they share the code and data segment.

Comparison between process and thread

Similarities between Process & Thread

- ✓ Both share the CPU, but only one executes at a time.
- ✓ Both execute sequentially (step-by-step).
- ✓ Both can create children (processes create child processes; threads can spawn new threads).
- ✓ Both can exist in states like Running, Ready, Blocked, or Terminated.
- ✓ Both maintain their own Program Counter (PC), Stack, Registers, and State.

Dissimilarities between Process & Thread

- ✓ **Independence**:
  - Processes are independent of each other.
  - Threads are **not independent**—they belong to a process.
- ✓ **Address Space**:
  - Processes have **separate address spaces**.
  - Threads **share the same address space** within a process.
- ✓ **Memory Access**:
  - A process generally **cannot access another process's memory** directly.
  - Threads **share all memory** of the parent process and can access it.
- ✓ **Cooperation**:
  - Processes **may or may not cooperate**, especially if owned by different users.
  - Threads are **designed to cooperate** to complete tasks together.

Benefits/Advantages of threads

1. **Minimized Context Switching Time**
   - ✓ Switching between threads is faster than switching between processes because threads share the same memory and resources.

2. **Concurrency within a Process**
   - ✓ Multiple threads can run concurrently inside a single process, improving responsiveness.

3. **Efficient Communication**
   - ✓ Since threads share memory, communication between them is faster and easier compared to inter-process communication (IPC).

4. **Easy Creation and Switching**
   - ✓ Threads are lighter than processes; creating and switching threads requires less overhead.

5. **Parallel Execution on Multiprocessors**
   - ✓ On multi-core CPUs, threads can truly run in parallel, increasing performance.

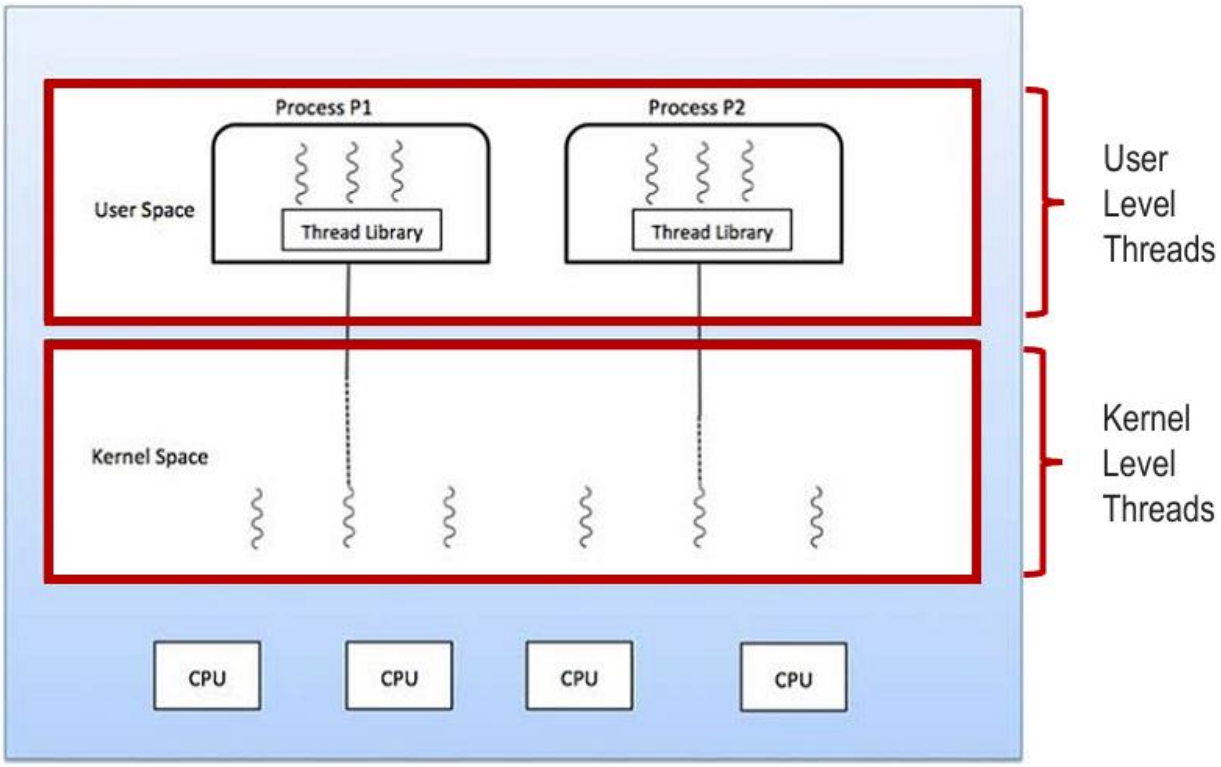
6. **Reduced Overhead compared to Processes**
   - ✓ Threads avoid per-process overhead (like separate address space, PCB, etc.), making them more efficient.

7. **Full Access to Address Space**
   - ✓ All threads of a process share the same memory and resources, making data sharing easier.

Types of threads

1. Kernel Level Thread

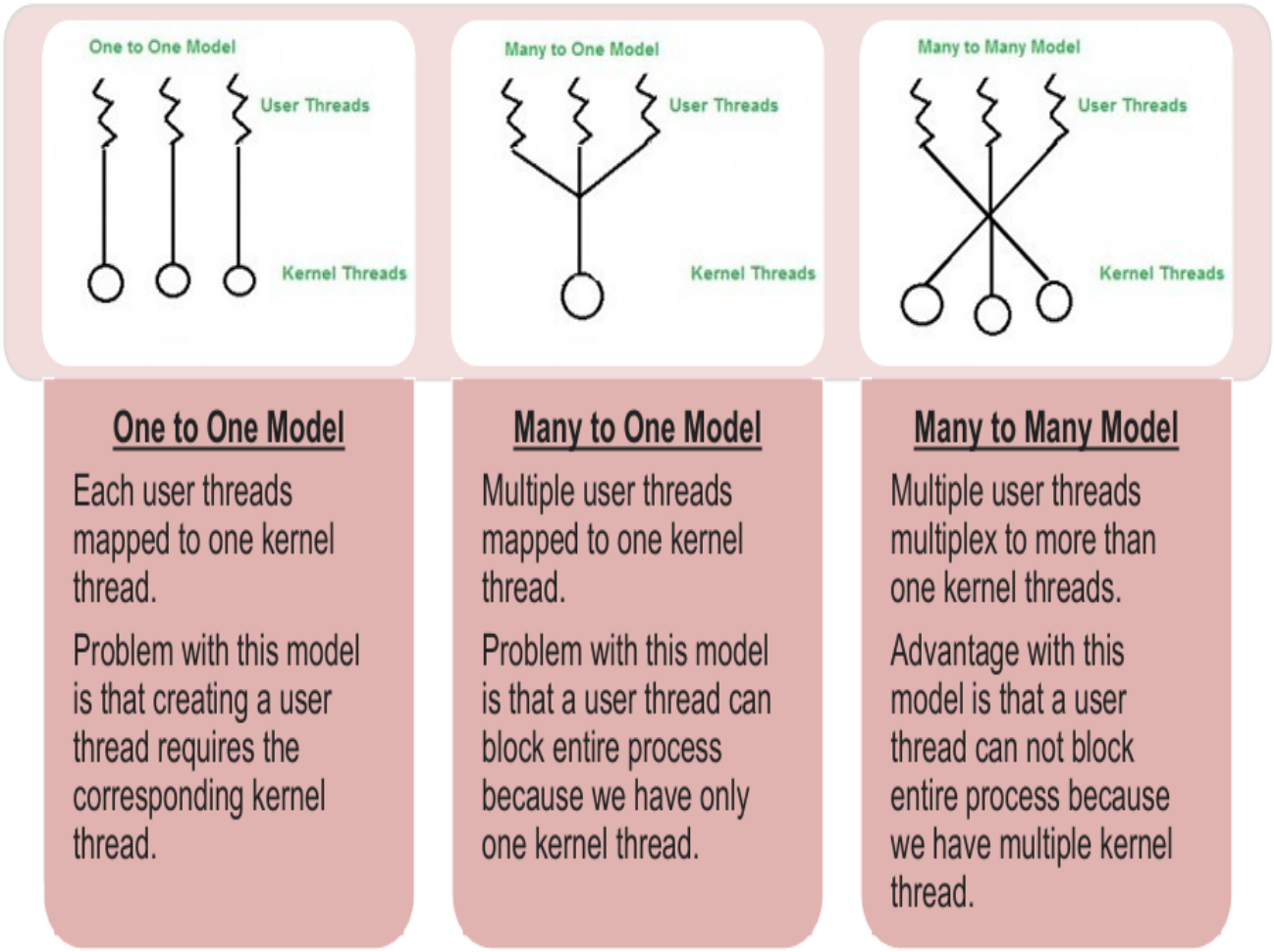
2. User Level Thread



Types of Threads

User Level Thread	Kernel Level Thread
User thread are implemented by users.	Kernel threads are implemented by OS.
OS doesn't recognize user level threads.	Kernel threads are recognized by OS.
Implementation of user threads is easy.	Implementation of kernel thread is complex.
Context switch time is less.	Context switch time is more.
Context switch requires no hardware support.	Context switch requires hardware support.
If one user level thread perform blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread with in same process can continue execution.
Example : Java thread, POSIX threads.	Example : Window Solaris

Multi Threading Models



One to One Model

Each user threads mapped to one kernel thread.

Problem with this model is that creating a user thread requires the corresponding kernel thread.

Many to One Model

Multiple user threads mapped to one kernel thread.

Problem with this model is that a user thread can block entire process because we have only one kernel thread.

Many to Many Model

Multiple user threads multiplex to more than one kernel threads.

Advantage with this model is that a user thread can not block entire process because we have multiple kernel thread.

Pthread function calls

1. Pthread_create:- Create a new thread

2. Pthread_exit:- Terminate the calling thread

3. Pthread_join:- Wait for a specific thread to exit

4. Pthread_yield:- Release the CPU to let another thread run

5. Pthread_attr_init:- Create and initialize a thread's attribute structure

6. Pthread_destroy:- Remove a thread's attribute structure


OPERATING
SYSTEM
SERVICES
RESOURCES
CALL
APPLICATION
THREAD
DRIVER
PROCESSOR
STORAGE
DESIGN
GUI
HARDWARE
COMPONENT
SERVER
MODERN
TIME
MANAGEMENT
KERNEL
INTERRUPT
SOFTWARE
COMPUTER

Thank you