Operating Systems Reading material

Reading Material

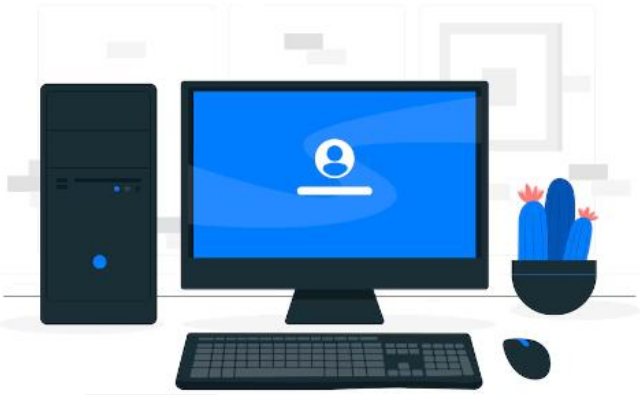
PW SKILLS

Topics

- Introduction to Operating System
- Understanding the concept of Operating system
- Operating system Architecture
- Process and threads
- Process synchronization and Concurrency
- Inter-Process Communication (IPC)
- File Systems and I/O system

Introduction to Operating System

What is computer ?
A computer is an electronic device that process data according to a set of instructions(program)
It performs task such as calculation, data processing and automated reasoning



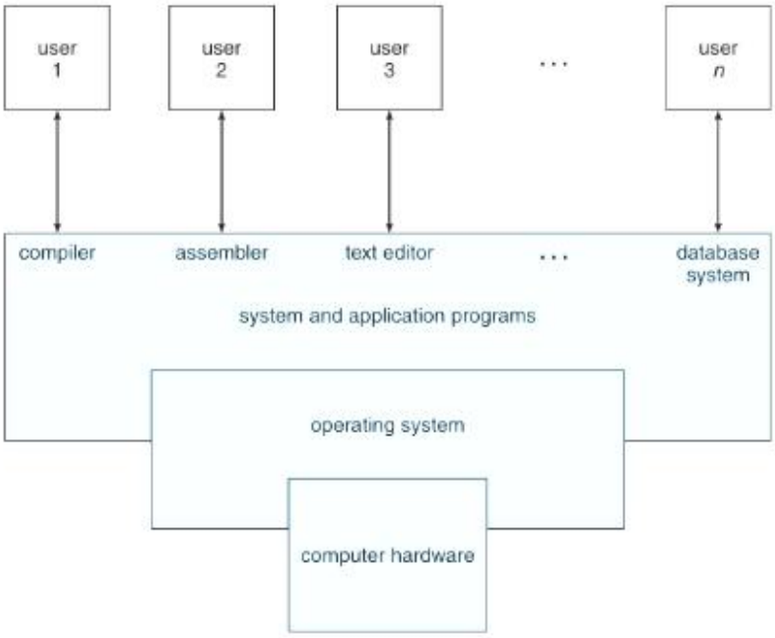
Component of a computer

Overall view component of a computer can be -
Hardware - central processing unit(CPU), the memory and the input/output devices, providing basic computing resources for the system.

The operating system - it controls the hardware and coordinates its use among the various application program for various users.

The application program - it can be of word processors, spreadsheets, compilers, web-browser which define the ways in which these resources are used to solve user's computing problem.

The user - a person using the computer.



The Hardware component of computers

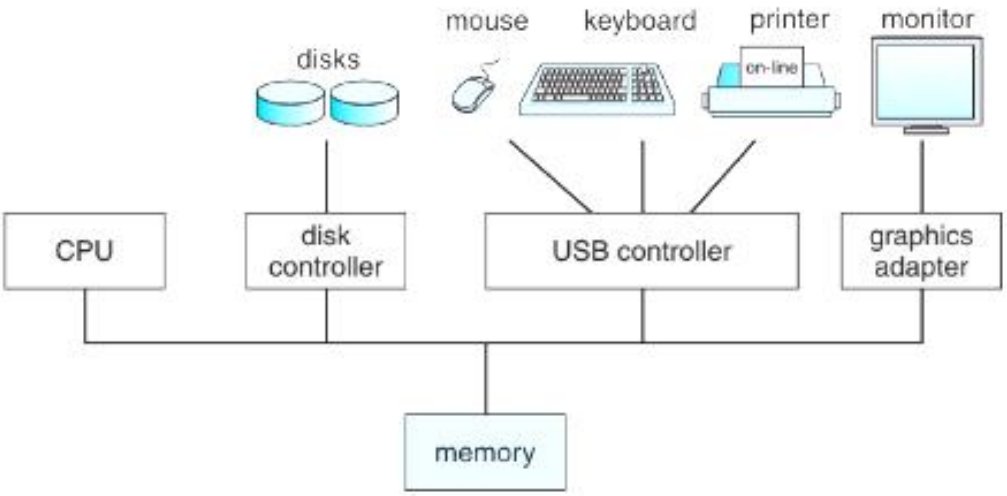
- **Central Processing Unit (CPU)** - the brain of the computer
- **Memory** - store data or instructions temporarily or permanently mainly RAM and ROM
- **Storage Devices** - store data long-term (hard disk drive & solid state drive)
- **Input Devices** - used to send data to a computer (e.g., keyboard, mouse).
- **Output Devices** - used to receive data from a computer (e.g., monitor, printer).
- **Motherboard** - main circuit board in a computer that houses the CPU, memory, and other essential components.
- **Power supply** - A component that provides electrical power to the computer.
- **Peripheral devices** - : External devices that connect to a computer and provide additional functionality (e.g., USB drives, printers).

The software component of computers

- **Operating System** - Manages hardware and software resources (eg. Window,MacOs and Linux
- **Application Software** - Program that perform specific tasks for users (eq. Microsoft office, web browsers
- **System Software** - supports applications software and manage hardware (eg. device drivers)

What exactly does a modern computer system look like

- A modern general purpose computer system consist of one or more CPUs an a number of device controllers connected through a common bus that provide access to shared memory
- Each device controller is in charge of a specific type of device (like disk drives, audio devices or video displays)
- The CPU and the device controllers can execute in parallel,competing for memory cycles
- To ensure orderly access to the shared memory,a memory controller synchronizes access to the memory.



The need to understand the Operating system

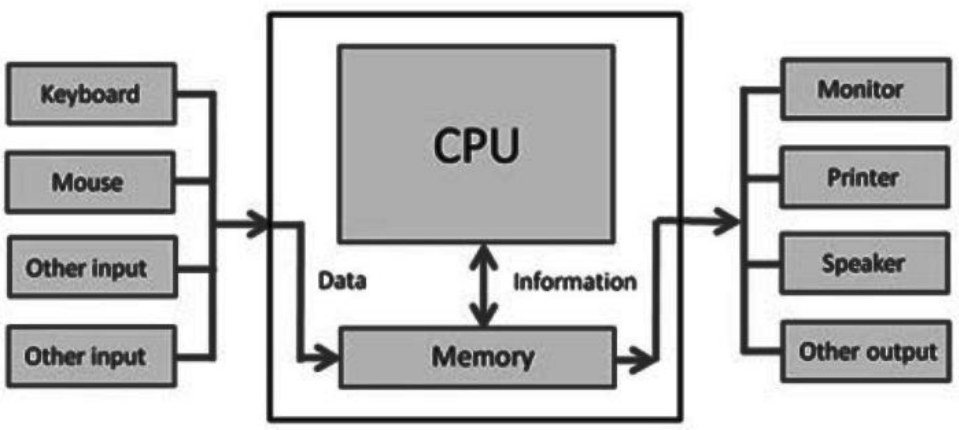
Here are some of the key reason why it's important to understand Operating systems are

- Foundation of computing knowledge
- Efficient use of computer Resources
- Software development
- System Administration and Troubleshooting
- Security Awareness

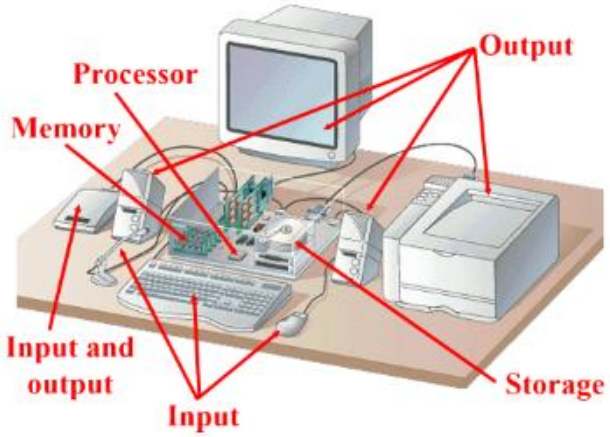
Types of Computers

- **Personal computers** - desktops and laptop for individual use
- **Servers** - Powerful systems providing resources and services to other computer over a network
- **Mainframes** - Large powerful computers used by large organizations for bulk processing
- **Supercomputer** - Extremely powerful system used for complex computations and simulation
- **Embedded system** - Computer integrated into other devices like cars, home appliances
- **Mobile devices** - Smartphone and tablets with computing capabilities.

How does computer works



Good knowledge to have before learning an Operating system.



- Basic Computer Knowledge
- Knowledge of how a computer system is organized and functions.
- Understanding of concepts like CPU operation, memory hierarchy, and I/O devices.
- Understanding of basic networking concepts (IP addresses, protocols).

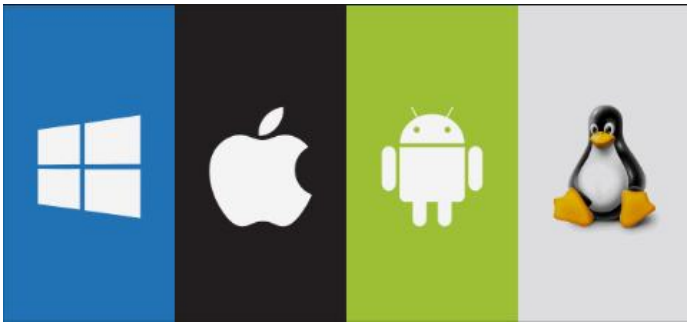
Question Time!

Question: why do we need to learn about the operating system ?

Understanding the concept of Operating system

What is an Operating System

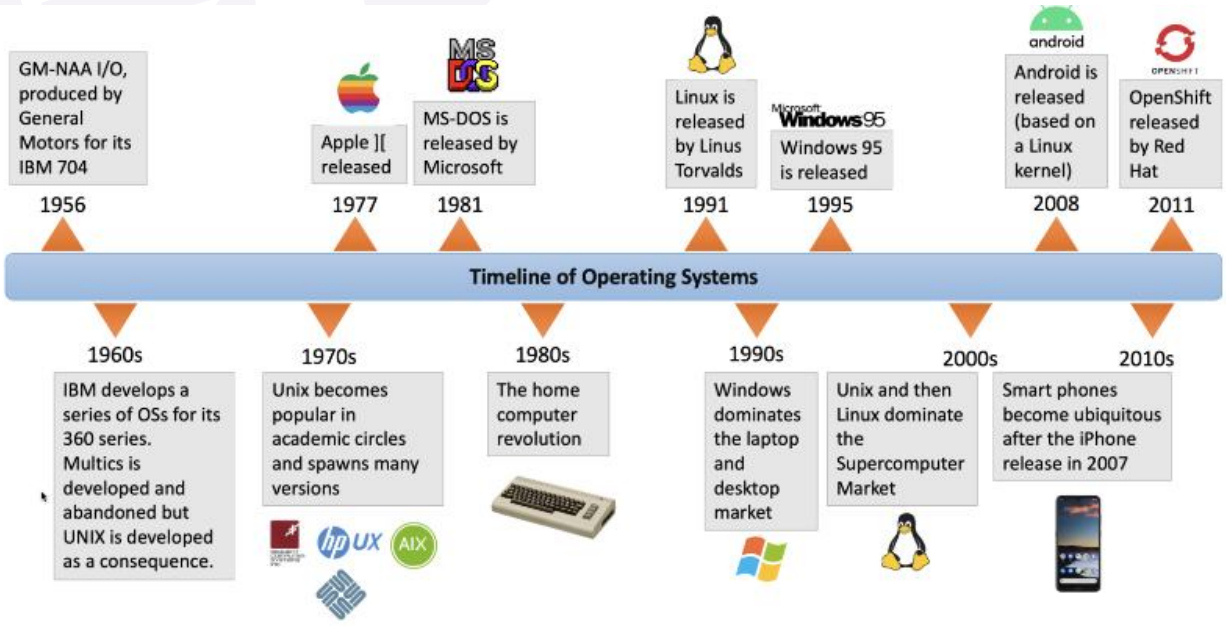
The Operating system can be defined as
system software that manages computer hardware resources and provinces common services for computer programs.
It acts as an interface between the user and the computer hardware.
Some popular operating systems include Windows, macOS, Linux, Android and iOS



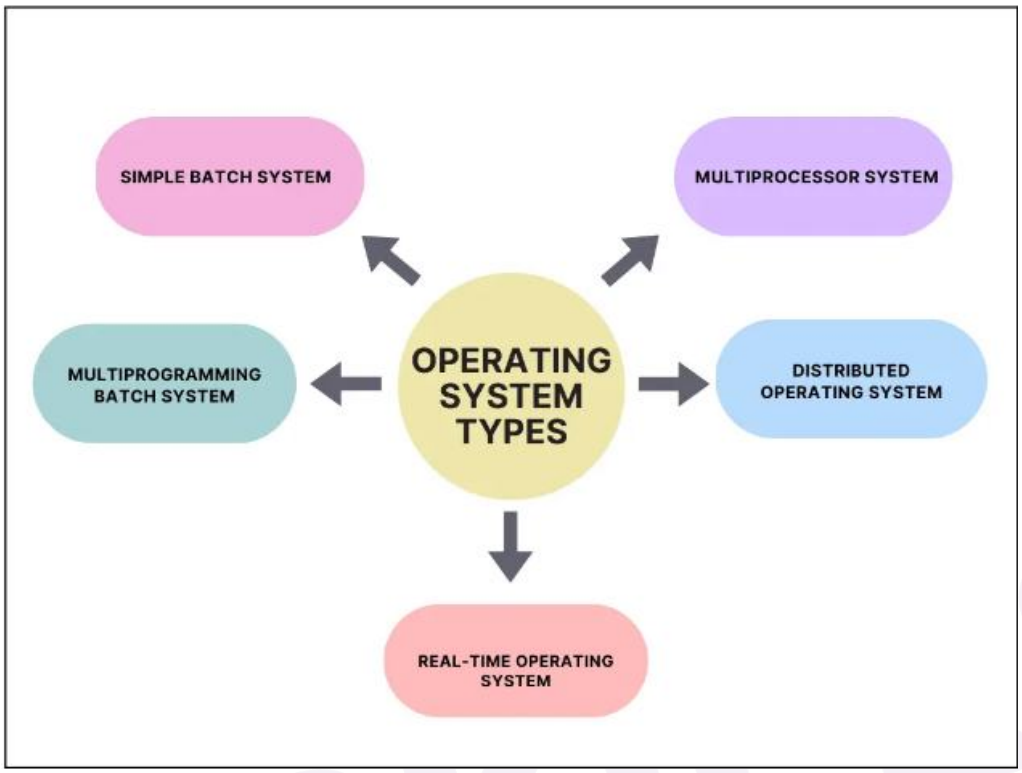
Objective of Operating System

- **objective of operating system include –**
  - Convenience
  - Efficiency
  - Hardware abstraction
  - System Resource Management

History of Operating Systems

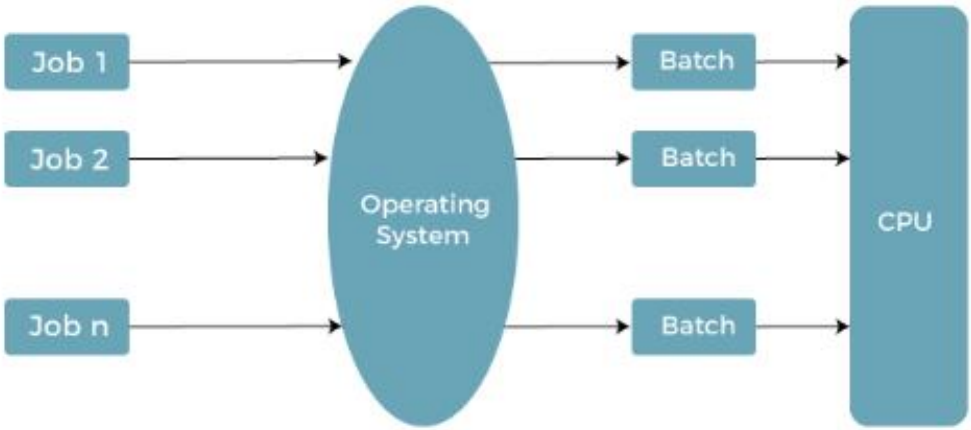


Types of Operating Systems



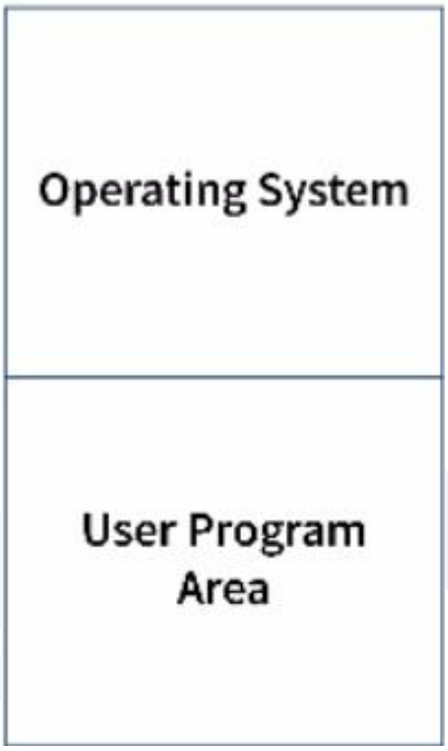
Batch Operating system

- In Operating system, the batch operating system can be defined as
  - a type of operating system that processes jobs or tasks in batches, without direct user interaction.
  - Batch operating systems can be of -
    - Simple batch system
    - Multi program batch system



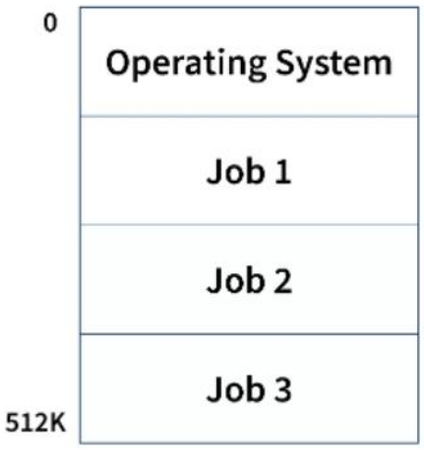
Simple Batch operating system

- **In simple batch operating system –**
  - Jobs with similar requirements are grouped together and submitted as a batch
  - The system processes one job after another without human intervention until all jobs are completed
  - Advantages include simplifying administration, reducing user response time, and efficiently processing large job batches
  - Disadvantages are potential resource inefficiency and lack of job scheduling or prioritization

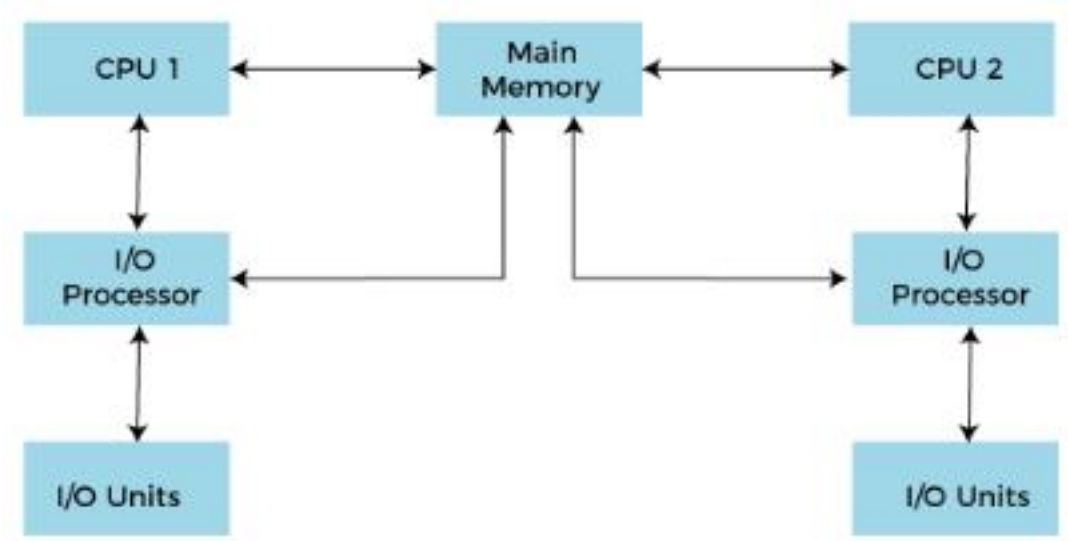


Multiprogramming Operating system

- **In Multiprogramming operating system –**
  - An advanced version that processes multiple jobs concurrently to maximize resource utilization
  - The operating system keeps several jobs in memory at a time and executes them one at a time, switching between jobs when one requires I/O
  - Supports job prioritization and can handle a large number of jobs
  - Advantages include high CPU utilization, shorter response time for short jobs, and increased throughput
  - Disadvantages are difficulty tracking all processes, need for CPU scheduling, and inability for user interaction during execution



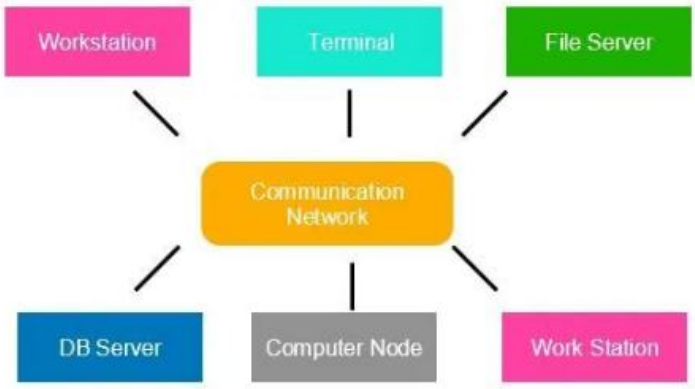
Multi Processor Operating System



Working of Multiprocessor System

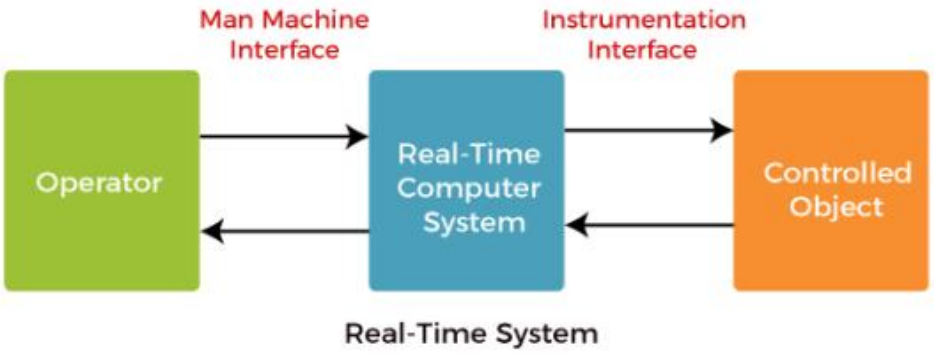
- Multiprocessor operating systems utilize multiple CPUs or processor cores within a single computer system.
- The operating system can distribute tasks and workloads across the available processors to improve overall system performance.
- This allows multiple programs or parts of a program to execute simultaneously on different processors.

Distribution Operating System



- A distributed operating system is a type of operating system where applications run on multiple interconnected computers or nodes.
- It allows the sharing of resources like CPUs, memory, storage, and network interfaces across the distributed system.
- Each node has its own local resources and operating system components, but they work together as a single, integrated system.
- Distributed OSes aim to provide the illusion of a single, centralized system to users, even though the underlying architecture is decentralized.

Real time Operating System



Real-Time System

- A real-time operating system (RTOS) is an operating system designed to handle time-critical tasks and applications.
- The key defining characteristic of an RTOS is its ability to provide a guaranteed response time, known as determinism.
- RTOS prioritize predictable and consistent execution times over maximum throughput, unlike general-purpose operating systems

Question Time

Question: which among the following is correct function of an operating system

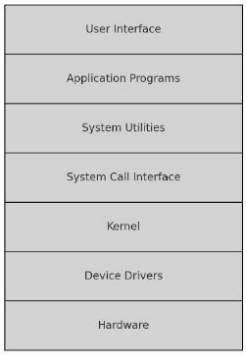
Operating system Architecture

Operating System Architecture

- Operating system architecture refers to the fundamental structure and design principles of an operating system.
- It describes how the system is organized and the key components that make up the OS.

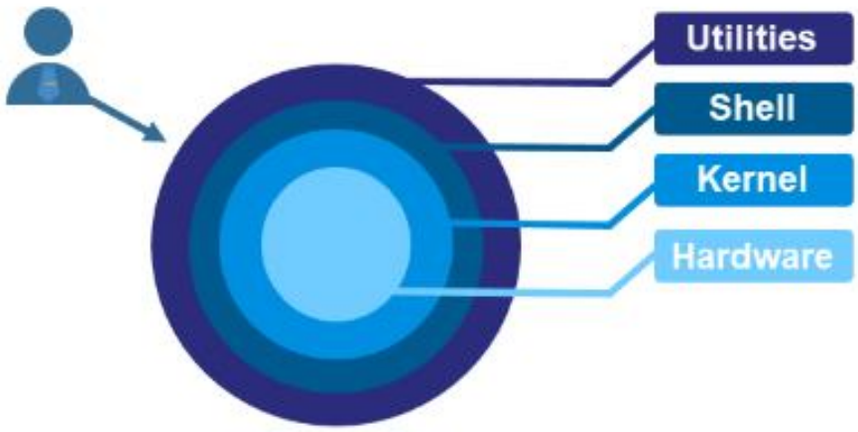
Layers Architecture of an Operating system

Layered Architecture of an Operating System



- **The layer of an Architecture of an Operating System include -**
  - **Hardware**: The physical components of the computer.
  - **Device Drivers:** Software modules that control and interact with hardware devices.
  - **Kernel:** The core part of the operating system that manages hardware resources.
  - **System Call Interface:** Provides the interface for application programs to interact with the operating system.
  - **System Utilities:** Programs that perform individual, specialized management tasks.
  - **Application Programs:** Software applications used by the user.
  - **User Interface:** The interface through which users interact with the computer system.

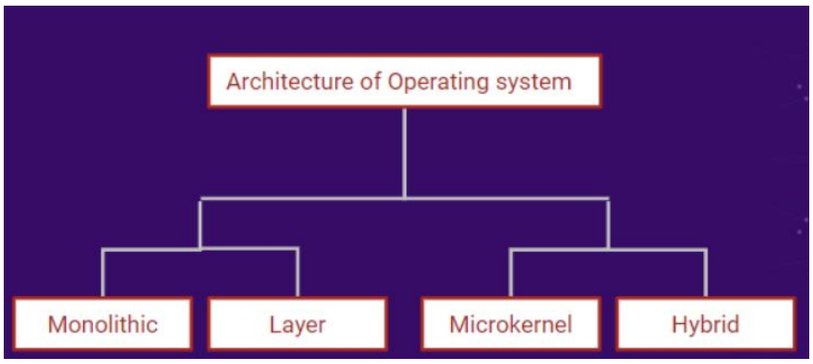
System components



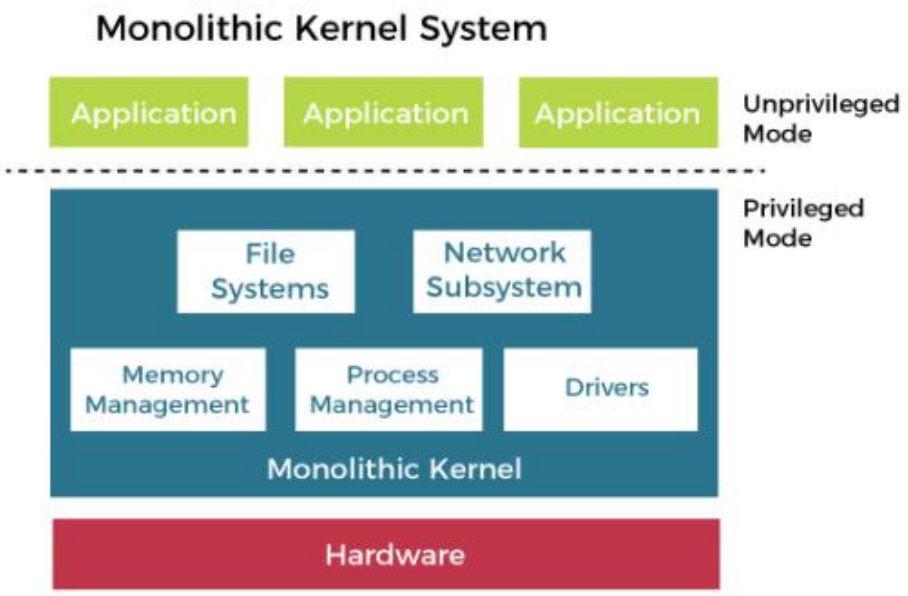
- The concept of system components in an operating system consist of three core components, they are
- **Kernel** - The kernels is the core part of an operating system and responsible of managing system resources and facilitating communication between hardware and software
- **Shell** - The shell acts as an intermediary between the user and the kernel. It provides a command-line interface (CLI) or a graphical user interface (GUI) that allows users to interact with the operating system by entering commands or using visual elements
- **User Interface** - The user interface (UI) is the part of the operating system that users interact with directly.

Types of Architectures of Operating System

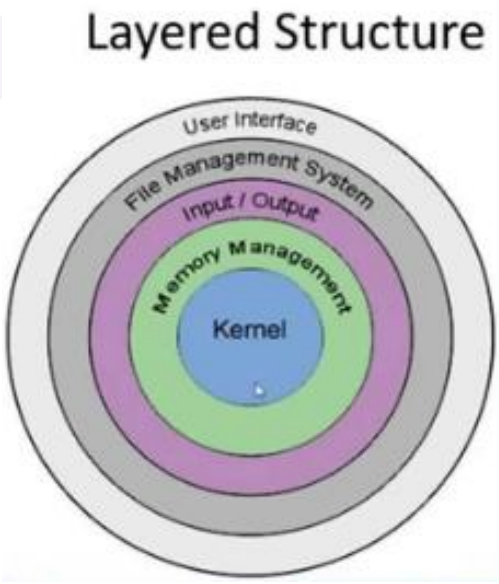
- **The architecture of an operating system can take different forms, including:**
- Monolithic Architecture
- Layer Architecture
- Microkernel Architecture
- Hybrid architecture



Monolithic Architecture

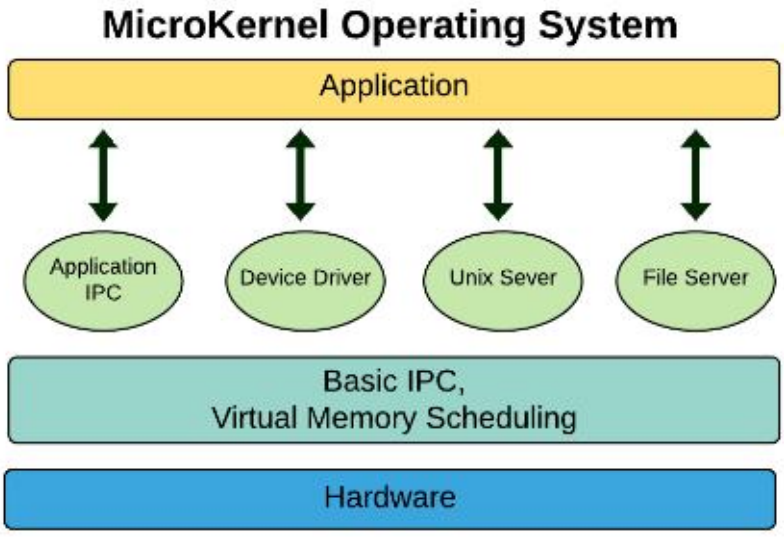


Layer Architecture



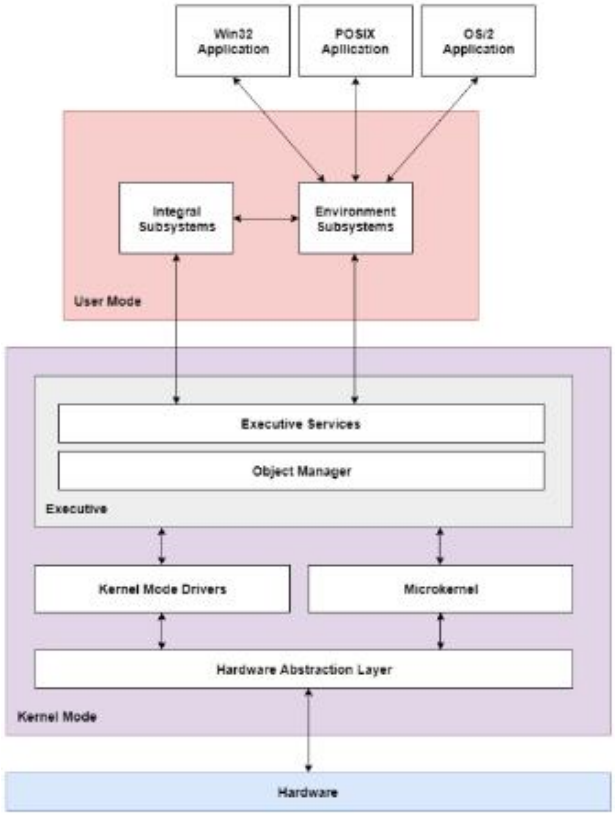
- Organizes the operating system into a hierarchy of layers
- Each layer provides services to the layer above it and uses services from the layer below it
- Provides modularity and ease of debugging and testing
- Can result in performance overhead due to the need to pass through multiple layers

Microkernel Architecture



- Moves as much of the operating system as possible into user space
- Kernel provides only minimal services like memory management, process scheduling, and inter-process communication
- Provides modularity, flexibility, and security as services run in user space
- Can suffer from performance overhead due to increased inter-process communication

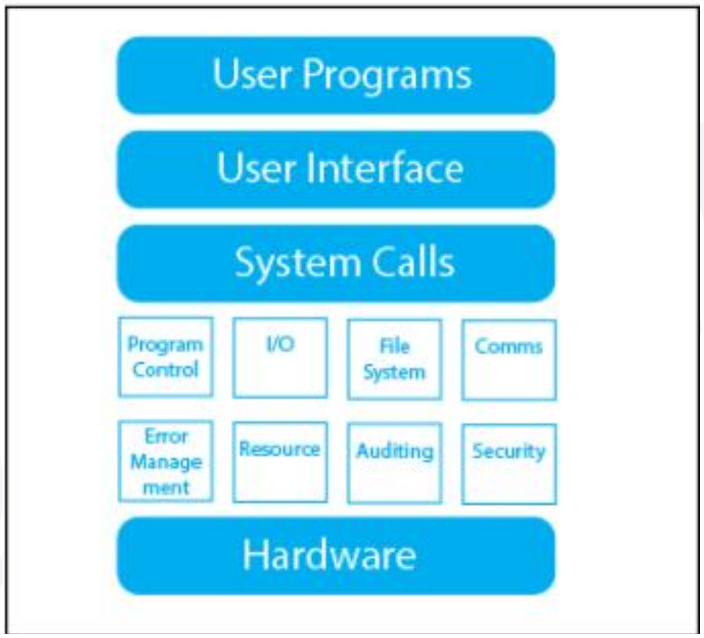
Hybrid Architecture



Combines features of both monolithic and microkernel architectures
Has a small microkernel that provides basic services, with additional services provided by kernel modules
Aims to provide the performance of a monolithic system with the modularity and flexibility of a microkernel
Can be more complex to design and implement compared to pure monolithic or microkernel systems

Overview of System calls and APIs

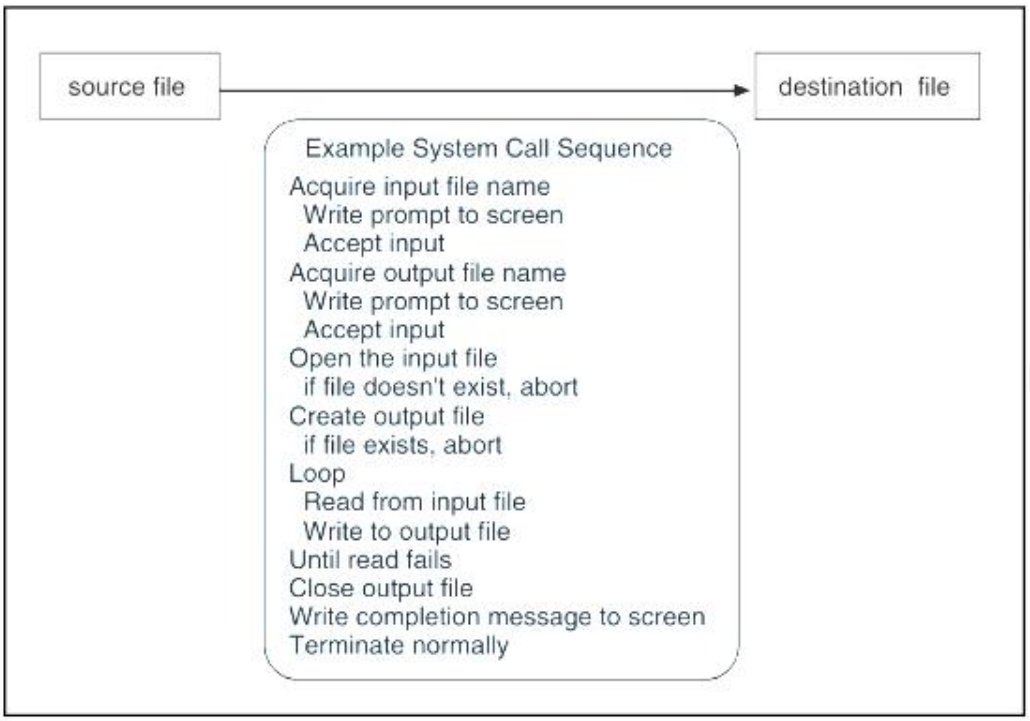
- System calls are the interface between a computer and the operating system
- They provide a way for applications to request services from the kernel, the core of the operating system
- When a program needs to access hardware resources, manage files, or perform other low-level operations, it cannot do so directly.
- Instead, it makes a system call, which is a request to the operating system to perform the desired task on its behalf.
- These calls are generally available as routine written in C and C++



Types of System calls

- The different types of system calls can be of
- **Process control** - calls related to process creation, termination, synchronization, and status.
- **File Management** - calls for creating, deleting, reading, writing, and managing files and directories.
- **Device management** - calls for controlling and accessing hardware devices like disks, printers, and networks.
- **Information Maintenance** - calls for retrieving and setting information about system resources and configuration.
- **Communications** - calls for establishing and managing communication between processes, both locally and across a network.

Example of System calls



Operating System Services

- Process Management
- Memory Management
- File System Management
- I/O System Management
- Security and Protection

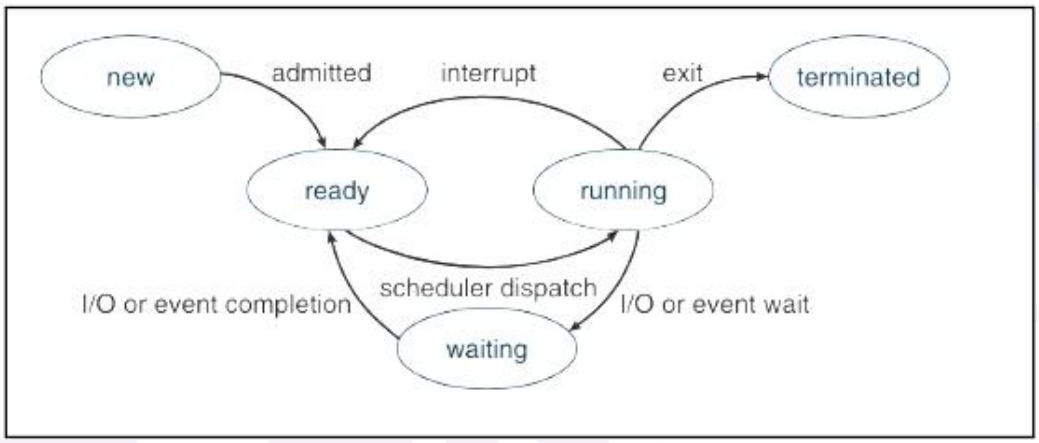
Question Time!

Question: Select the correct statement of the a system calls in an Operating system.

Process and threads

Process state

- As a process executes, it changes state.
- The state of a process is defined in part by the current activity of that process.
- A process may be in one of the following states
- New state. The process is being created.
- Running state. Instructions are being executed.
- Waiting state. The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- Ready state. The process is waiting to be assigned to a processor.
- Terminated state. The process has finished execution.

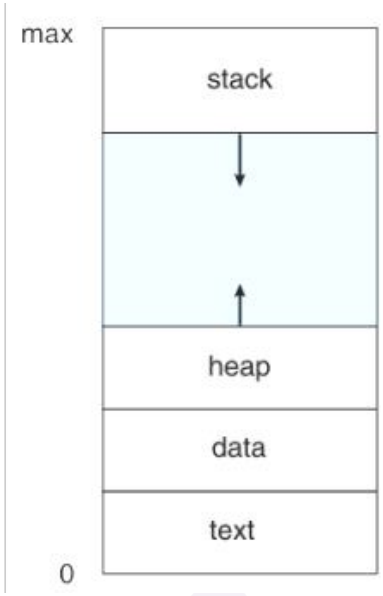


Process state - things to remember

- A process necessarily goes through minimum 4 states.
- A single processor can execute only one process at a time.
- Moving a process from wait state to suspend wait state is a better alternative.

State	Present in Memory
New state	Secondary Memory
Ready state	Main Memory
Run state	Main Memory
Wait state	Main Memory
Suspend wait state	Secondary Memory
Suspend ready state	Secondary Memory
Terminate state	–

Process and Attributes



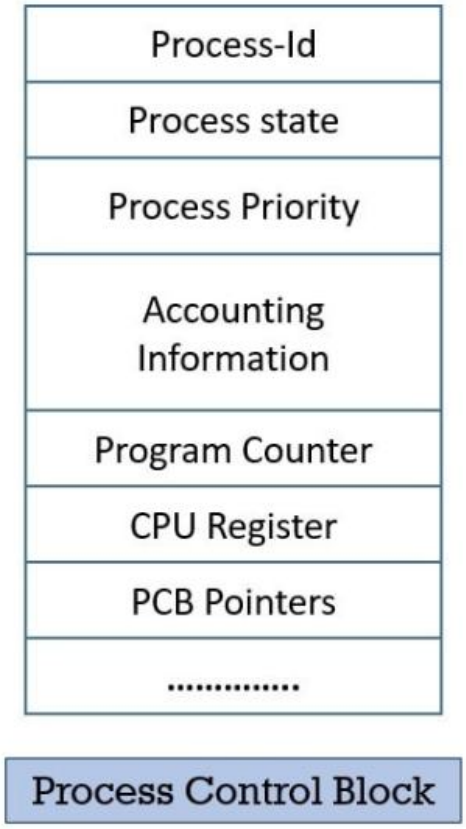
- Process can be defined as the execution of computer programs.
- It is an active entity that executes a series of instructions to perform a specific task or function.

- **The main attribute of of a process are -**
- Program code/Text: The instructions that the process will execute.
- **Data:** The data that the process will use during its execution.
- **Stack:** A data structure used to store temporary data like function parameters and return addresses.
- **Heap:** A data structure used to store dynamically allocated memory.
- **Process control block (PCB):** A data structure that contains information about the process, such as its state, priority, and memory usage.

Process Control Block (PCB)

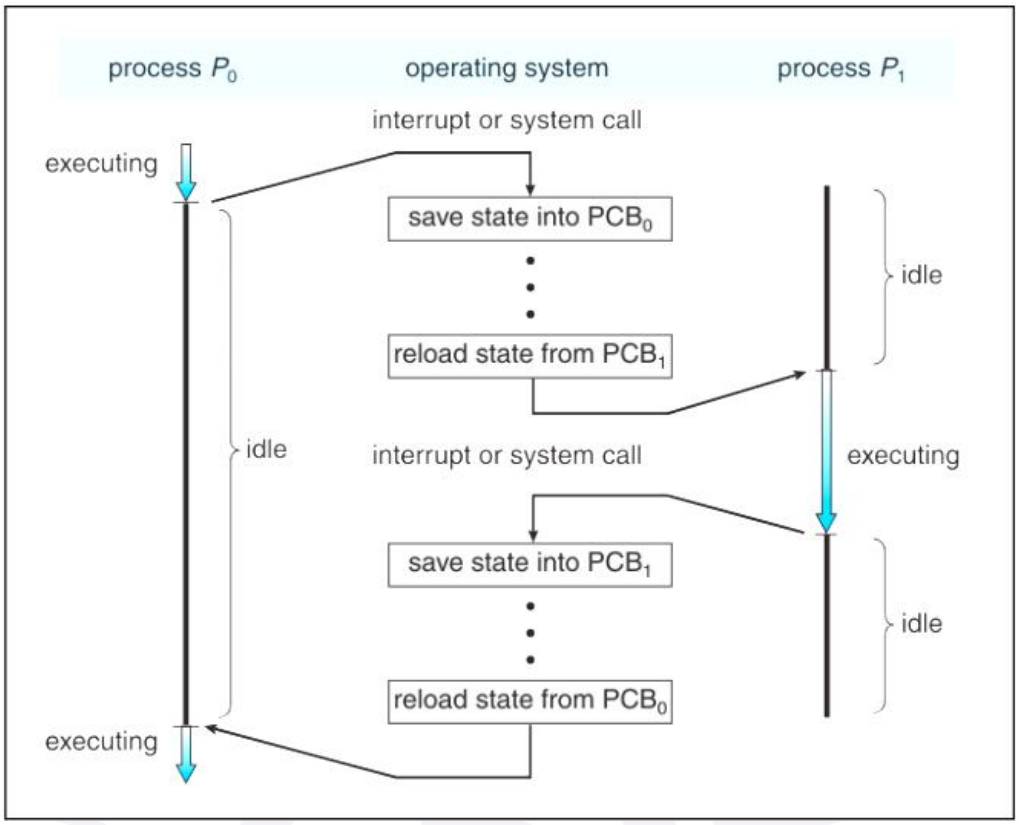
- Each process is represented in the operating system by a process control block
- Also called Task control Block
- It is a data structure that stores information about a particular process
- It consist of
- Process state
- Program counter
- CPU registers
- CPU-scheduling information
- Memory-management information

Various attribute of process stored in PCB

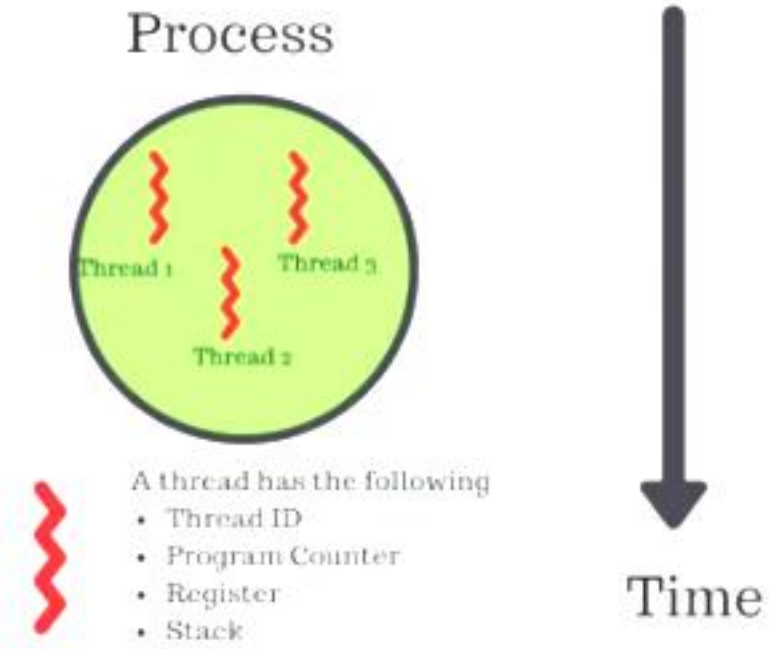


- **Process Id** - a unique Id that identifies each process of the system uniquely.
- **Program Counter** - specifies the address of the instruction to be executed next.
- **Process State** - process goes through different states during its lifetime
- **Priority** - specifies how urgent is to execute the process.
- **General Purpose Registers** - registers are used to hold the data of process generated during its execution.
- **List of Open Files** - maintains a list of files used by the process during its execution
- **List of Open Devices** - maintains a list of open devices used by the process during its execution.

Overview of Process execution



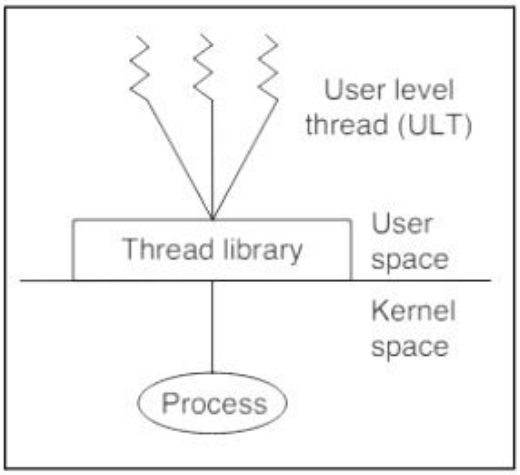
What is Threads

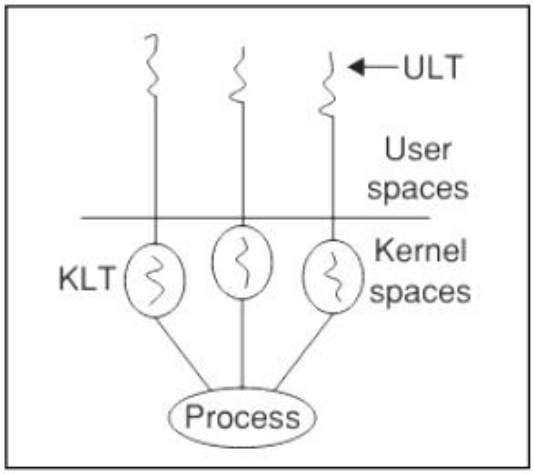


- A Thread in an operating system is a lightweight process that is a basic unit of CPU utilization.
- A thread consists of a program counter, a register set, and a stack. It is a single sequential flow of execution within a process.
- Threads allow a process to perform multiple tasks concurrently by dividing the process into multiple, independently executing sub processes or threads.
- Threads share the same memory and resources of the process they belong to, which allows for efficient communication and resource sharing between them.

Type of Thread

- **There are two main types of threads -**
- **User-level threads** - Managed by a user-level library, not recognized by the operating system kernel.
- **Kernel level threads** - Managed directly by the operating system kernel.

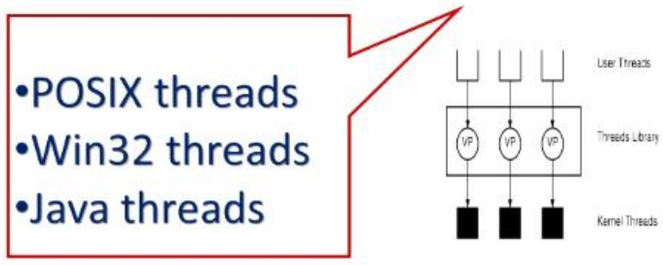




The need for Threads in OS

- **Threads are used in operating systems for several key reasons:**
  - Improved Responsiveness
  - Resource Sharing
  - Faster Context Switching
  - Multiprocessor Utilization
  - Enhanced Throughput

Thread libraries



- Thread libraries provide an API (Application Programming Interface) that allows developers to create and manage threads within their applications.
- There are two main approaches to implementing thread libraries:
- **User**-level thread libraries - implemented entirely in user space, without direct kernel support.
- **kernel** -level thread libraries - implemented with direct support from the operating system kernel.

Examples of thread library APIs –

- **POSIX threads (pthreads)** - available in Unix systems
- **Windows threads** - available in windows operating systems
- **Java threads** - Part of the java programming language

Question Time!
Question: A Thread in an operating system is a lightweight process that is a basic unit of ?

Process synchronization and Concurrency

Various Times Related to process

Arrival time - is the point of time at which a process enters the ready queue.
Waiting time - the amount of time spent by a process waiting in the ready queue for getting the CPU.
Response time - the amount of time after which a process gets the CPU for the first time after entering the ready queue.
Burst time - amount of time required by a process for executing on CPU.
Completion time - is the point of time at which a process completes its execution on the CPU and takes exit from the system.
Turn Around Time - the total amount of time spent by a process in the system.

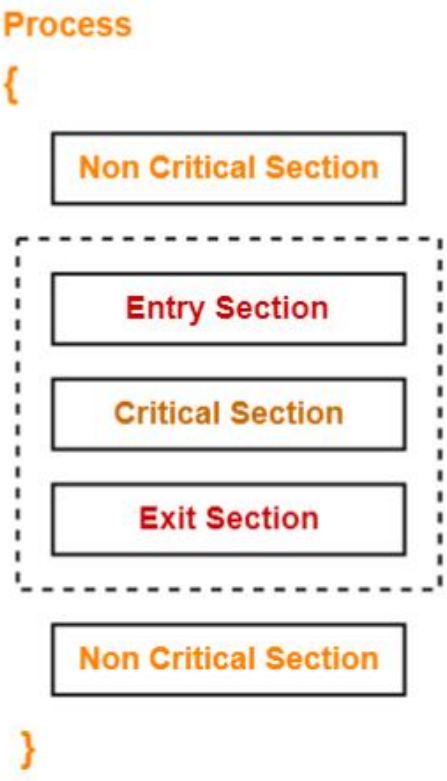
Critical Section Problem

- The Critical Section Problem is a fundamental concept in operating systems that deals with the synchronization of concurrent processes accessing shared resources.

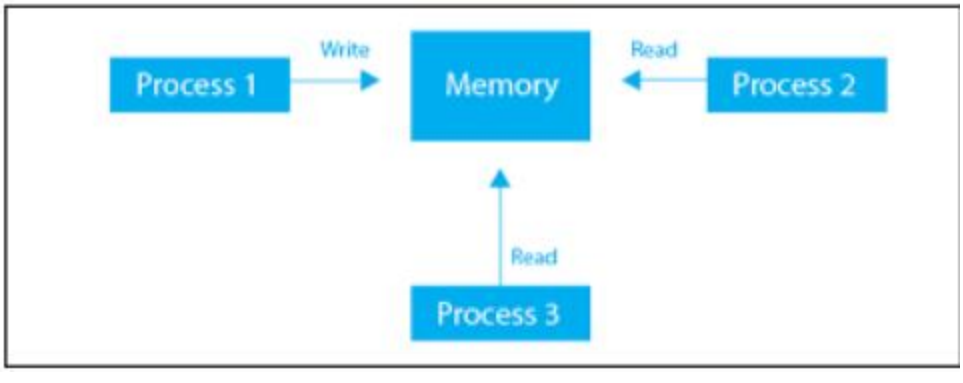
- **The key points are:**
- **Critical section** - refers to part of a program where shared resources are accessed.
- **The Critical section problems** - a design protocol that ensures mutual exclusion, so that only one process can be in its critical section at a time

- **Requirement for solution** -
- Only one process can be in its critical section at a time
- If No process is in the section, a waiting process should be allow to enter
- There should be limit on the number of times a process has to wait before it can enter its critical section.

- **Solutions –**
- Software base (using locks, semaphores and atomic operations) and hardware base (CPU instructions for synchronization)



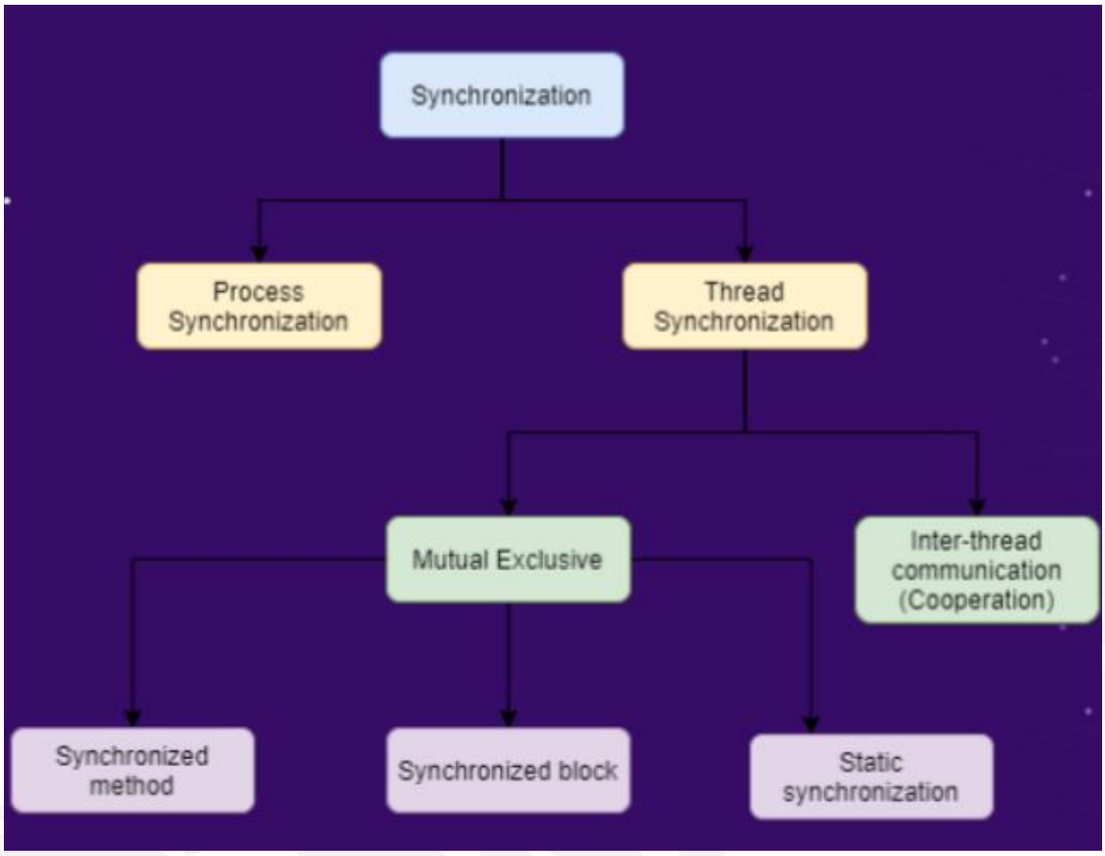
Synchronization



- In operating system, Synchronization refers to the coordination and control of multiple processes or threads that are accessing shared resources or data.

- **The need for Synchronization –**
- **Concurrent Execution:** In modern operating systems, multiple processes and threads can execute concurrently. Without synchronization, these processes and threads might interfere with each other, leading to inconsistent data or unexpected behavior.
- **Shared Resources:** When multiple processes or threads access shared resources (e.g., memory, files), synchronization ensures that these resources are used correctly and without conflicts.

Types of Synchronization



The different types of Synchronization are –

- **Process Synchronization:** Deals with the coordination between different processes.
- **Thread Synchronization:** Deals with the coordination between different threads within the same process.

Synchronization Mechanisms

The synchronization mechanisms are techniques used to coordinate the execution of processes and thread ensure that resources are in controlled and consistent manner.

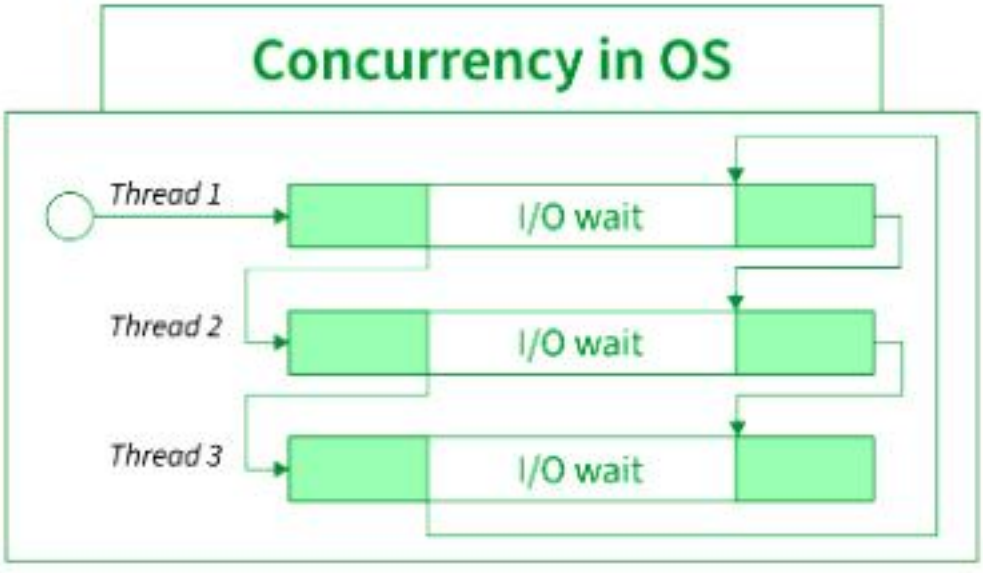
Some of the Synchronization Mechanisms are

- Locks
- Semaphores
- Monitors
- Condition variables
- Barriers

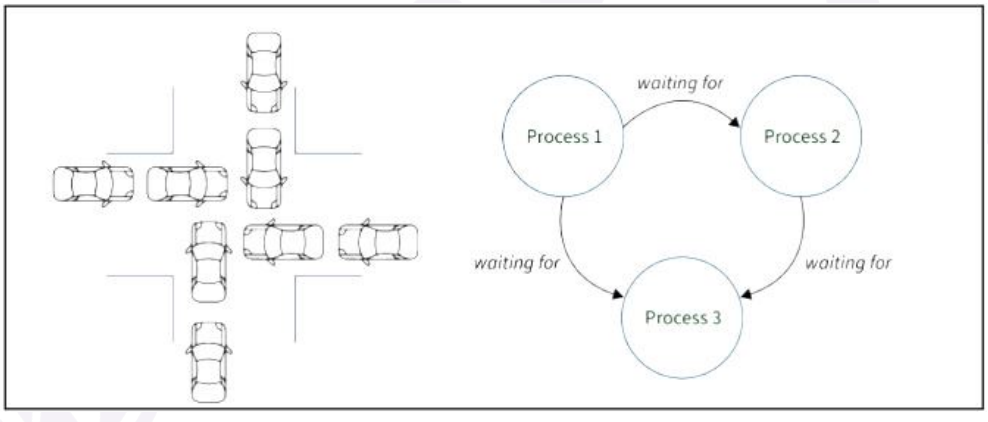
Concurrency

- Concurrency is the fundamental design of an Operating System and encompasses a host of design issues, including communication among processes, sharing of and competing for resources,
- synchronization of the activities of multiple processes, and allocation of processor time to processes.

- **There are two examples of concurrent processing are as follows**
  - In a single-processor multiprogramming system, processes are interleaved in time to yield the appearance of simultaneous execution.
  - In a multiprocessor system, it is possible not only to interleave the execution of multiple processes but also to overlap them



Challenges of concurrency



- The challenges of Concurrency include the following
- Critical section
- Deadlock
- Mutual exclusion
- Race Condition
- Starvation

CPU Scheduling

In an Operating system -

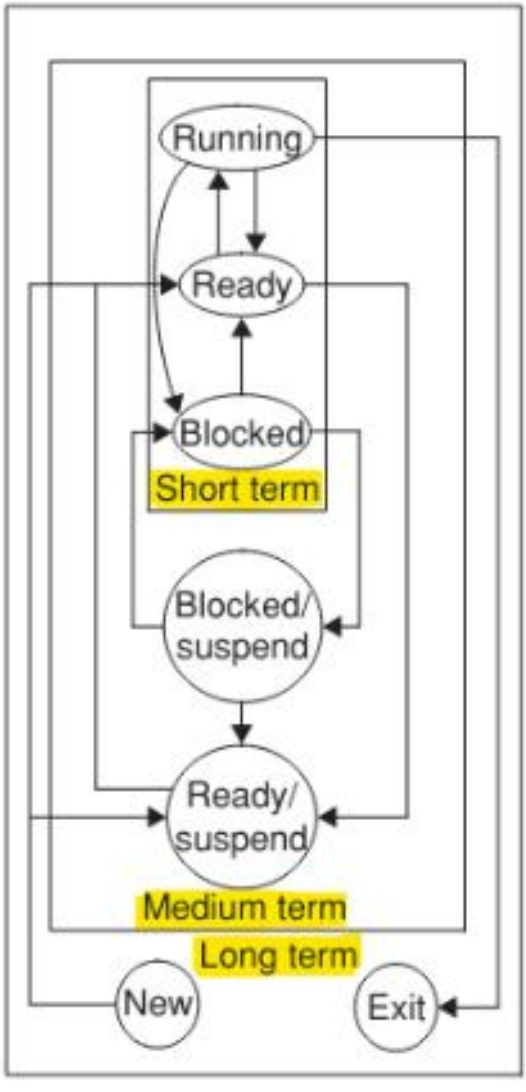
- The objective of the multiprogrammed OS is to maximize CPU utilization by having some processes running at all times
- The objective of time-shared OS is to switch the CPU among processes so frequently that the users can interact with each program while it is executing
- When there is more than one process ready to execute with the processor, a selection decision needs to be made to pick a process for execution from among the ready processes. This activity is called process scheduling.

Scheduling Queue

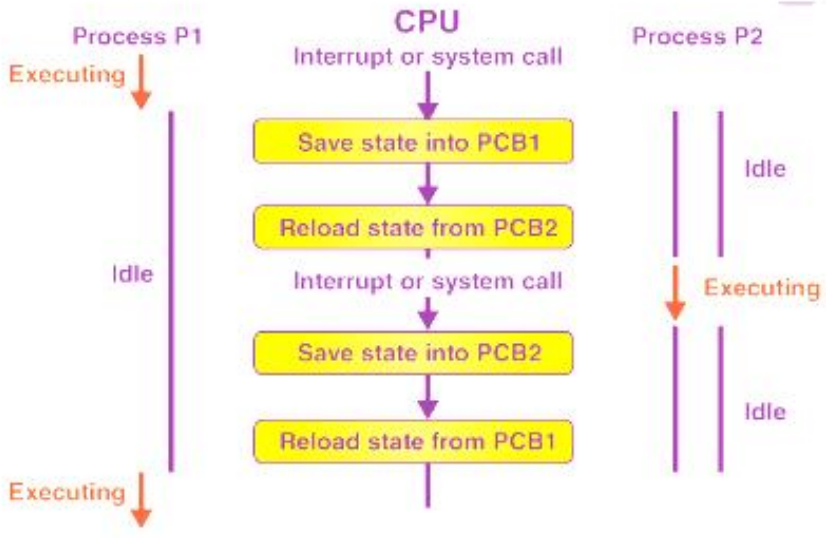
- It maintains information on all ready processes for CPU devices. It is maintained as a linked list.
- Types of scheduling queues can be
- **Job queue** - It consists of all processes in the system
- **Ready queue** - It consists of all processes that are residing in the main memory and are ready but waiting to execute on the CPU
- **Device queue** - It consists of processes waiting for a particular I/O device. Each device has its queue.

Scheduler

- There are three types of processor scheduling
- Long-term scheduling -Long-term scheduling, also known as job scheduling, is a type of process scheduling in operating systems that decides which processes are admitted to the system for processing.
- Medium-term scheduling - Medium-term scheduling is a part of the swapping function in operating systems that handles swapped-out processes
- Short-term scheduling - Short-term scheduling, also known as CPU scheduling, is a type of process scheduling that selects from a group of ready processes to allocate the CPU to one of them.



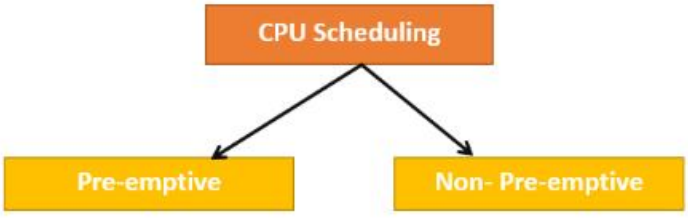
Context switching



- the process of saving the state of a currently running process or thread and restoring the state of the next scheduled process or thread.
- includes the program counter, CPU registers, memory management information, and other process-specific data.
- Steps in Context Switching
- Saving State
- Loading state
- Switching Execution

Types CPU scheduling algorithms

- **Some of the CPU scheduling algorithms are -**
  - **Non-Preemptive Scheduling** - Once a process start it CPU burst, it cannot be preempted until it finishes.
    - **First-Come-First-Serve (FCFS):** Serves processes in the order they arrive, like a line at a food truck.
    - **Shortest Job Next (SJN):** Prioritizes processes with the shortest execution time, similar to choosing the shortest queue at a buffet.
  - **Preemptive Scheduling** - A running process can be interrupted and moved to the ready queue.
    - **Priority Scheduling:** Assigns priority levels to processes, with higher-priority tasks getting CPU attention first, like accommodating urgent guests at a busy restaurant.
    - **Round Robin:** Allocates a fixed time slice to each task, ensuring fairness, like a carousel where each rider gets a turn before the next.
    - **Multilevel Queue Scheduling:** Divides tasks into different queues based on their characteristics, with each queue having its scheduling algorithm.

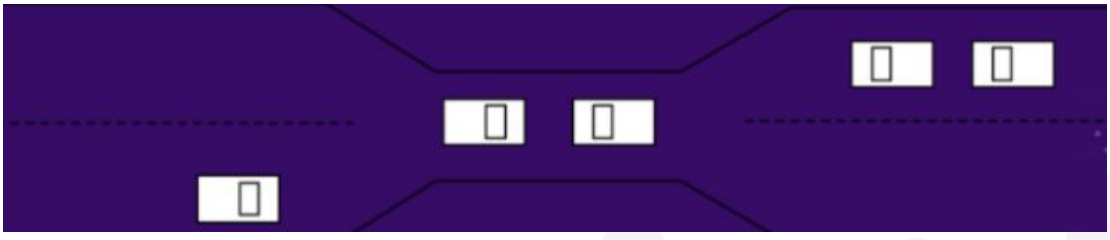


Deadlock

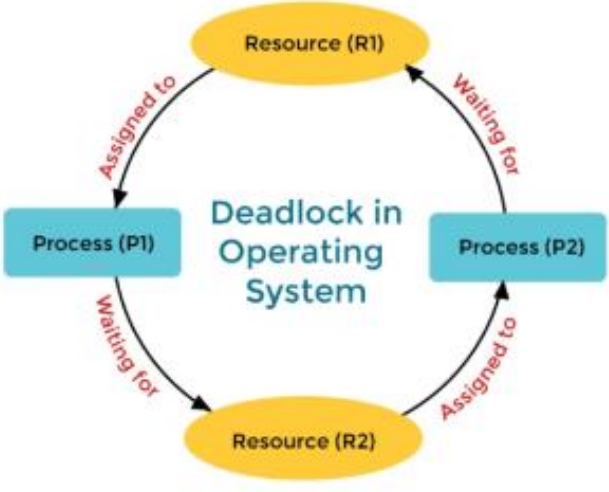
Deadlock can be defined as a situation where a process or set of processes is blocked waiting for some resource that is held by another waiting.

Example of illustration of deadlock -
Consider a Bridge crossing

- Traffic is allowed only in one direction. Each section of a bridge can be viewed as a resource.
- If a deadlock occurs, it can be resolved if one car backs up (pre-empt resources and rollback)
- Several cars may have to be backed up if a deadlock occurs
- The problem of starvation (infinite wait) is possible.



Deadlock characteristics



Deadlock is an undesirable state of the system. The following are the four conditions that must hold simultaneously for a deadlock to occur

- **Mutual exclusion** - A resource can be used by only one process at a time. If another process requests for that resource then the requesting process must be delayed until the resource has been released.
- **Hold and wait** - Some processes must be holding some resources in a non-sharable mode and at the same time must be waiting to acquire some more resources, which are currently held by other processes in a non-sharable mode.
- **Circular wait** - Deadlock processes are involved in a circular chain such that each process holds one or more resources being requested by the next process in the chain.

Methods of Handling DeadLocks

There are three approaches to dealing with deadlocks, they are

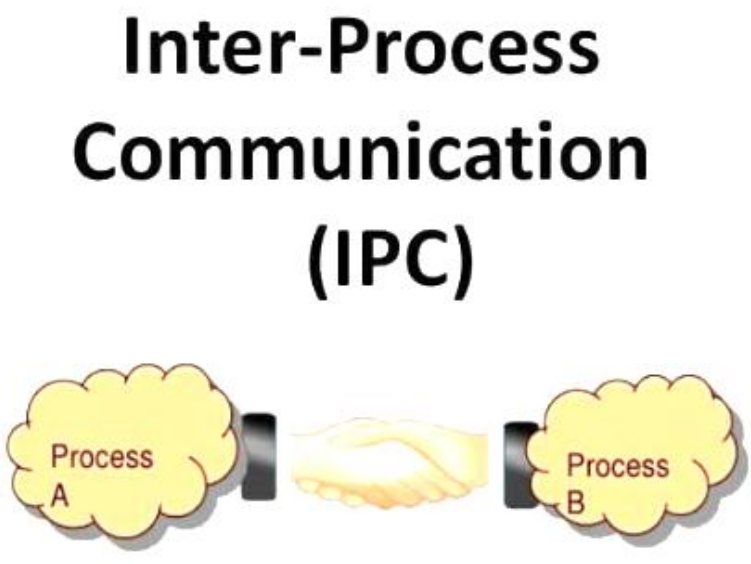
- **Deadlock prevention** - involves designing the system in a way that ensures deadlocks cannot occur by imposing certain restrictions on resource allocation.
- **Deadlock Avoidance** - involves dynamically examines the resource allocation state to ensure that a circular wait condition can never exist, allowing the system to avoid deadlocks.
- **Deadlock detection** - involves periodically checking the system for the existence of a deadlock and then recovering from it by aborting processes or reallocating resources.

Question Time!

Question: What is critical section in process synchronization

Inter-Process Communication (IPC)

What is Inter-process Communication (IPC):



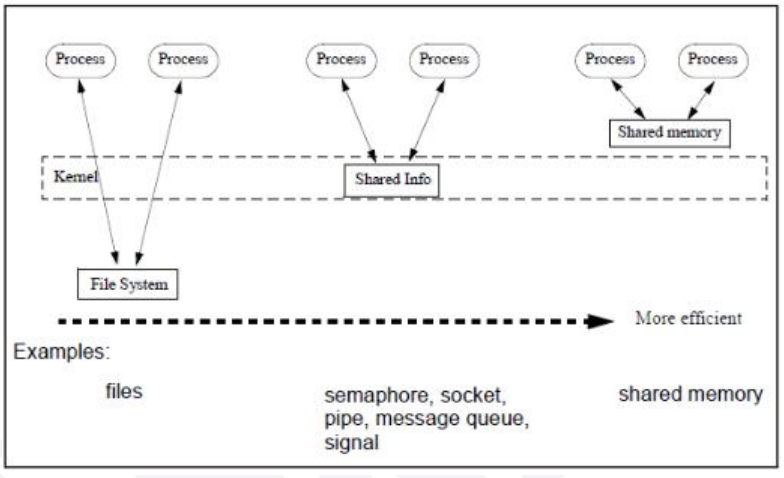
In an Operating system, IPC is the mechanism that allows processes to communicate with each other, it involves processes of sharing data or signaling events to coordinate their actions and exchange information. IPC is necessary to enable processes to cooperate, coordinate, and exchange information, which is vital for modern computing systems and multitasking environments

Process can communicate with each other through both:
Shared Memory
Message passing

Role of Synchronization in IPC

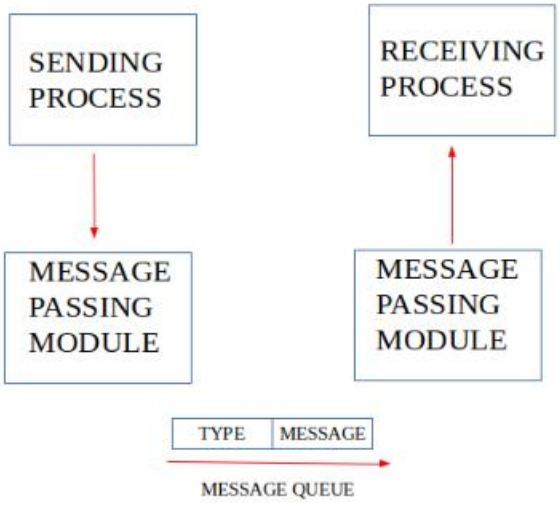
- It is one of the essential parts of inter process communication.
- Typically, this is provided by interprocess communication control mechanisms,
- but sometimes it can also be controlled by communication processes
- Some methods used to provide the synchronization are -
  - Mutual Exclusion
  - Semaphore
  - Barrier
  - Spinlock

Inter Process Communication Mechanism -



- Message passing
- Shared memory
- Pipes
- Sockets
- Signal

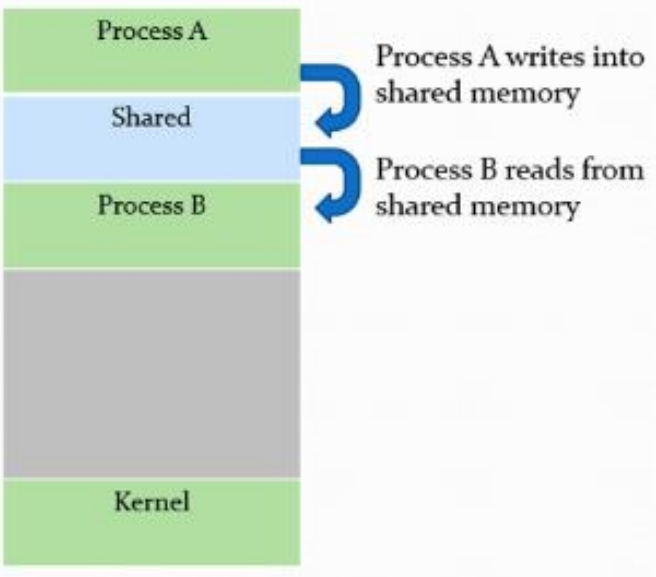
Message Passing



- Allow processes to send and receive messages in a queue.
- Messages are stored in a queue until the receiving process retrieves them.
- Useful for complex communication patterns where messages need to be prioritized or filtered.

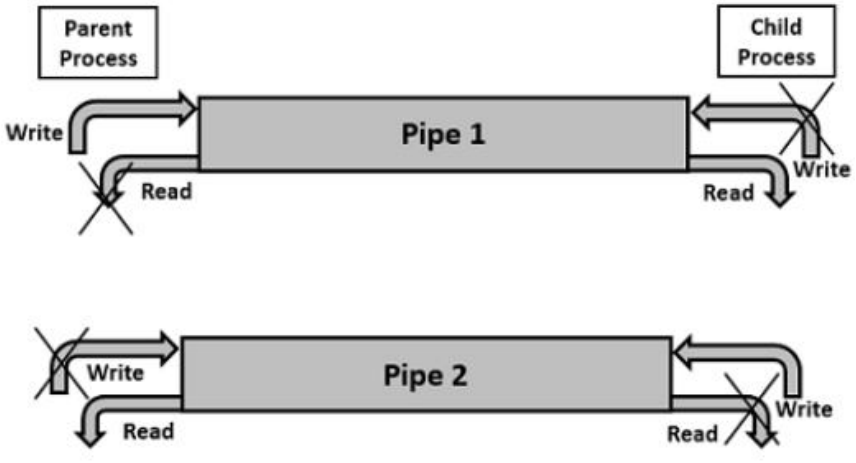
Shared Memory

- Fastest form of IPC, as it allows multiple processes to access the same memory space.
- Requires synchronization mechanisms (like semaphores or mutexes) to avoid data inconsistency due to concurrent access.



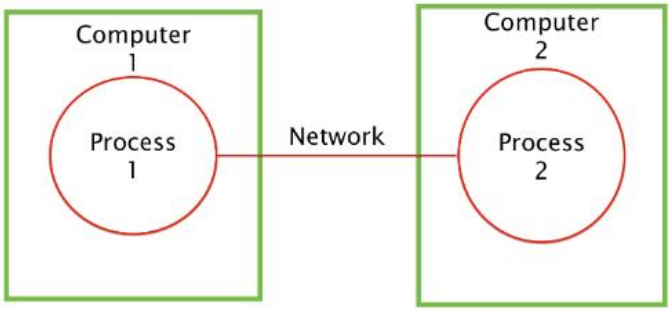
Pipes

- **Anonymous Pipes:** Unidirectional communication channel used for communication between parent and child processes.
- **Named Pipes (FIFOs):** Can be used for bidirectional communication and can exist independently of the processes, allowing communication between unrelated processes.



Sockets

- Provide a communication channel over a network, enabling IPC between processes on different machines.
- Can use different protocols, such as TCP (connection-oriented) or UDP (connectionless).



Signal

- Used to notify a process that a particular event has occurred.
- Can be used for inter-process communication, especially for handling asynchronous events like interrupts.

Wy we need interprocess communication

There are numerous reasons -

- Sharing Data and Resources
- Modularity and Flexibility
- Synchronization and Coordination
- Distributed Computing
- Handling Exceptional Situations

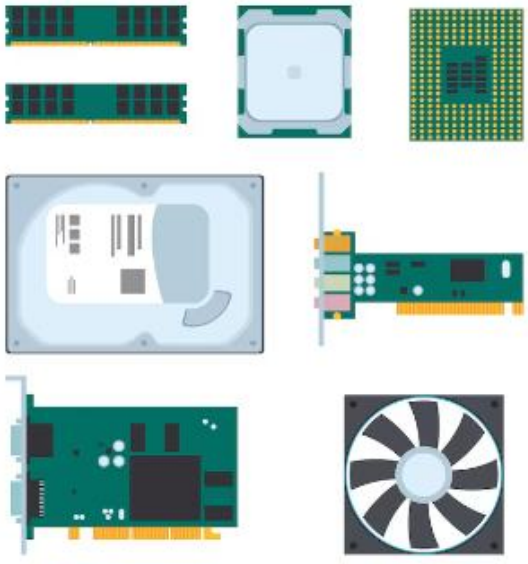
Question Time!

Question: Which IPC mechanism would be most appropriate for communication between processes running on different machines?

Memory management

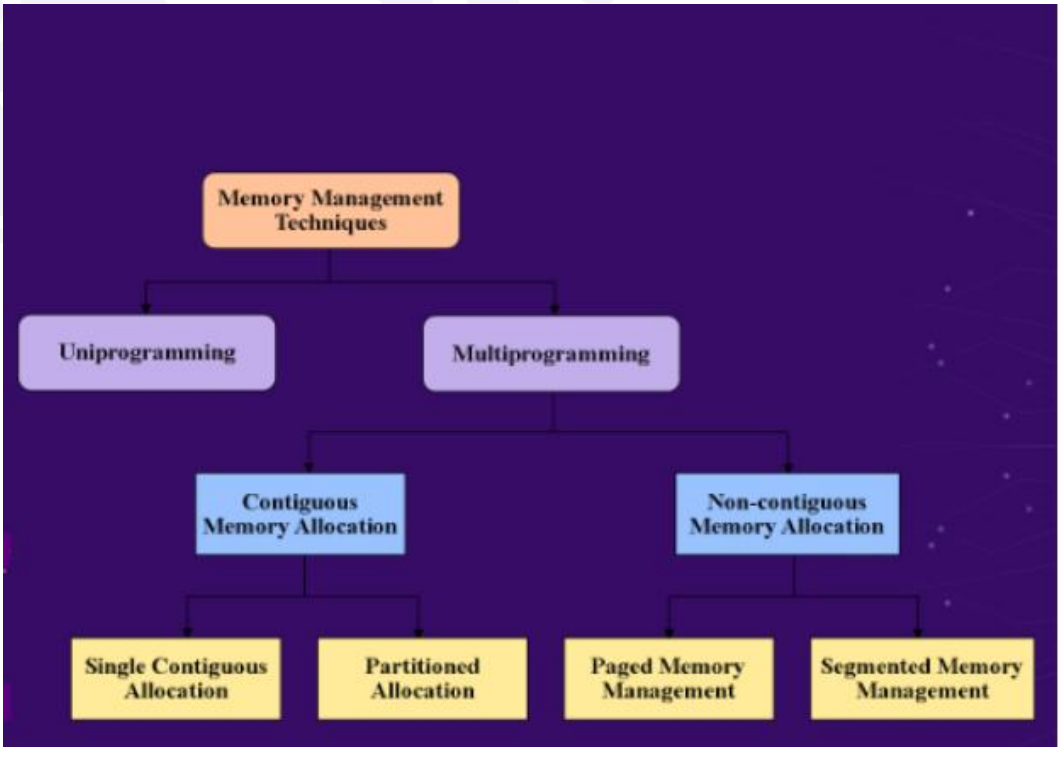
Memory management

- Memory Management is a fundamental function of an operating system (OS) that involves controlling and coordinating the use of the computer's main memory among the running process.
- **The main goals are -**
  - Allocation and Deallocation
  - Protection
  - Sharing
  - Logical Organization



Memory Basic Concept

- The basic concept of memory in operating systems include -
- **Uniprogramming system:** Main memory is divided into two parts as follows:
  - Operating system (OS) part
  - Program part (which is currently being executed)
- **Multiprogramming system:** Here the user part of memory must be further subdivided to accommodate multiple processes. The task of subdivision is carried out dynamically by the OS and is known as memory management.

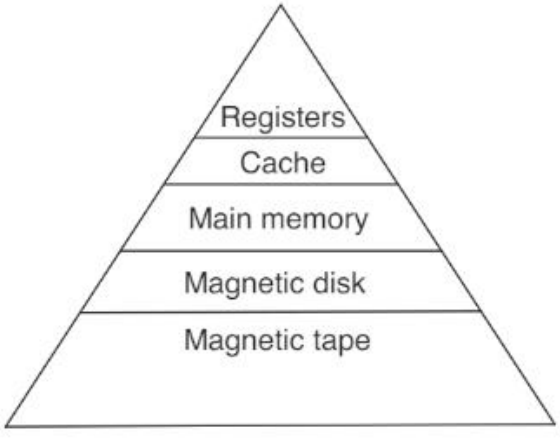


Memory Hierarchy

- The memory hierarchy has different types of storage systems in computers which are arranged in hierarchy, with respect to speed and cost
- From the diagram, if one moves down the hierarchy:
  - access time increases, the cost per bit decreases, the memory capacity increases and memory access frequency by the processor decreases.
  - The registers, cache, and main memory are volatile, whereas magnetic discs and magnetic tapes are non-volatile storage devices.

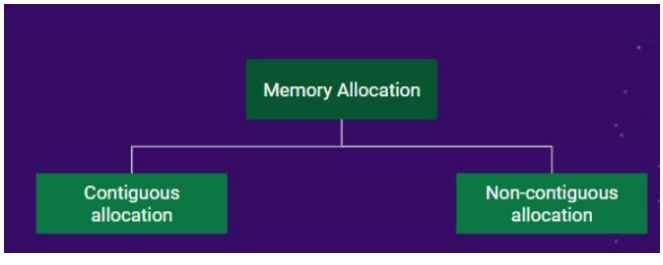
Memory Hierarchy

- The memory hierarchy has different types of storage systems in computers which are arranged in hierarchy, with respect to speed and cost
- From the diagram, if one moves down the hierarchy:
- access time increases, the cost per bit decreases, the memory capacity increases and memory access frequency by the processor decreases.
- The registers, cache, and main memory are volatile, whereas magnetic discs and magnetic tapes are non-volatile storage devices.



Memory Allocation

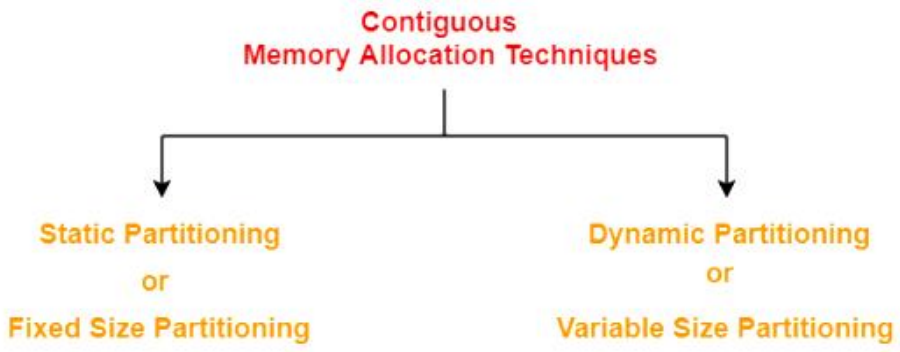
- Memory allocation refers to the process of assigning memory to programs and processes in a computer system. It involves reserving space in computer memory for the storage of data and instructions.
- It can be of -
  - Contiguous Allocation
  - Non-Contiguous Allocation (Paging, Segmentation)



Contiguous Allocation

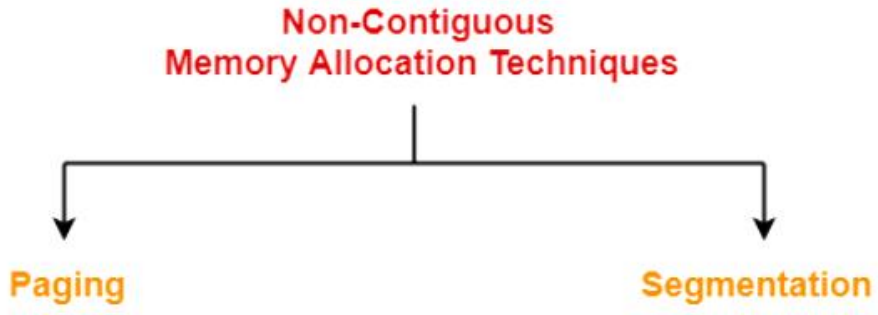
The memory allocation technique –

- It allows to store the process only in a contiguous fashion.
- Thus, entire process has to be stored as a single entity at one place inside the memory.



Non-Contiguous Allocation

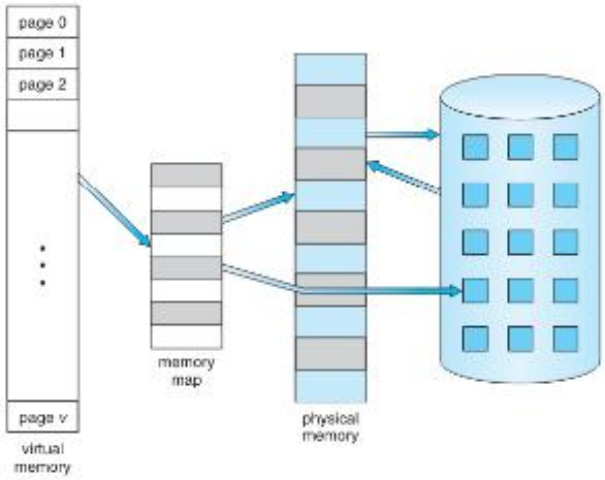
- It allows to store parts of a single process in a non-contiguous fashion.
- Thus, different parts of the same process can be stored at different places in the main memory.



Virtual Memory

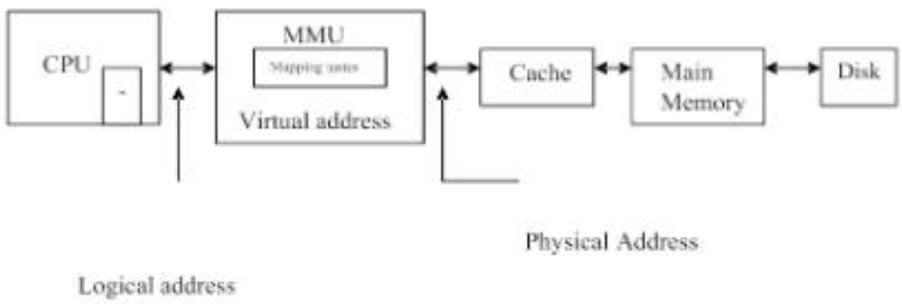
In operating system –

- Virtual memory is a technique that allows the execution of processes that are not completely in memory
- Virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory.



How is Virtual Memory works

- The Virtual Memory is implemented by employing a memory management unit (MMU) to translate every logical address into a physical address reference
- The MMU is imposed between the CPU and the physical memory where it performs these translations under the control of the operating system
- Each memory reference is sued by the CPU is translated from the logical address space to the physical address space.
- Mapping tables guide the translation, again under the control of the operating system.



Concepts on Virtual memory

- **Demand paging** - Demand paging is a memory management technique used in virtual memory systems. In demand paging, pages are loaded into main memory only when they are needed or accessed by a running process.
- **Page Fault** - A page fault occurs when a process references a page that is not currently in main memory
- Page Replacement - Page replacement algorithms are used to select a page to be replaced from memory when a new page needs to be loaded and there are no free frames available.

Question Time!

- **Question:** In Operating system the technique used to increase the effective size of physical memory is ?

File Systems and I/O system

What is File

The file system consist of two distinct parts -

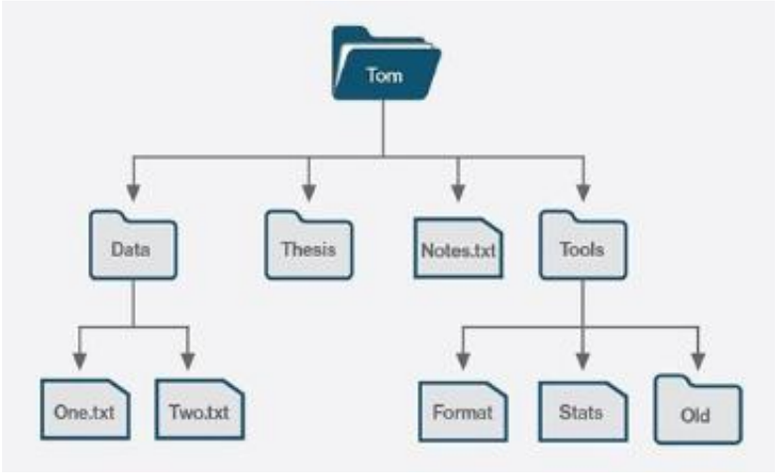
- Collection of files
- Directory structure

File - a file is a name collection of related information that is recorded on secondary storage. The files must have

- Long term existence
- Shareable between processes
- Structure

File Concepts

- **File Attribute** - A typical file attribute are name, identifier, type, location, size, protection, time, date and user identification.
- **File operation** - The operations that are applied on files are creation, deletion, closing, reading and writing



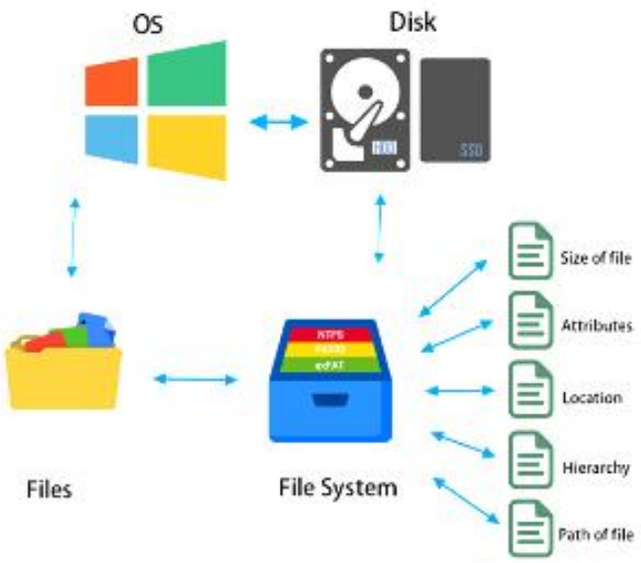
File types

File types - A common technique for implementing file types is to include the type as part of the file name, The is split into two parts:
A name
An extension (usually separated by a period character

The type of a file may be

1. Executable (exe, com, bin)
2. Object (obj, o)
3. Source code (c, cc, java)
4. Batch (bat, sh)
5. Text (txt, doc)
6. Word processor (wp, text, doc)
7. Library (lib)
8. Print or view (ps, pdf, jpg)
9. Archive (zip, tar)
10. Multimedia (mpeg, mov, rm)

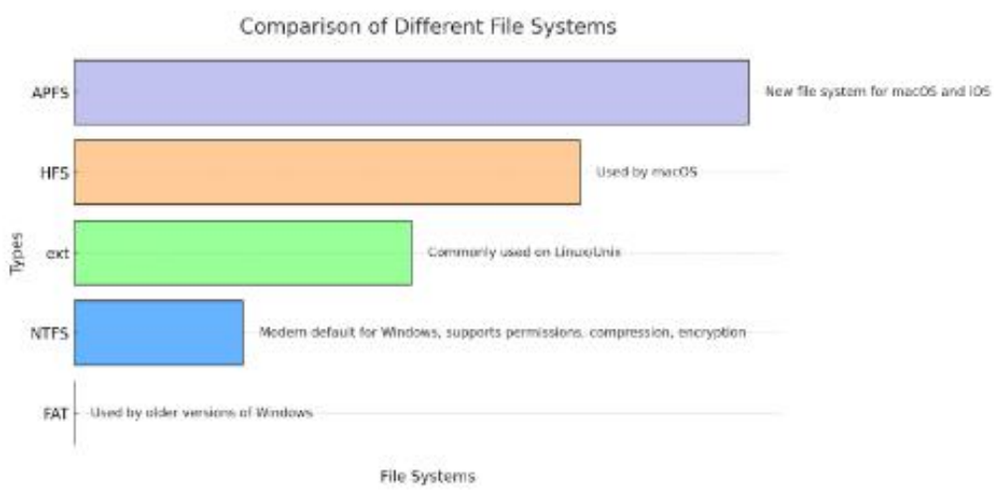
What is File system



- A file system is a method used by an operating system to store, organize, and manage files and directories on storage devices like hard drives, SSDs, and USB drives.
- It acts as a bridge between the OS and the physical storage hardware, allowing users and applications to create, read, update, and delete files in an organized manner.
- Providing a structured way to store and retrieve files
- Handling issues like deciding where to store files on the disk and recovering free space when files are deleted
- Keeping track of which disk blocks belong to each file, even if they are not contiguous
- Defining different file structures for various file types like text files, source code files, and binary executable files

Some common file system types include

- **FAT (File Allocation Table)** - used by older versions of Windows
- **NTFS (New Technology File System)** - modern default for Windows, supports permissions, compression, encryption
- **ext (Extended File System)** - commonly used on Linux/Unix
- **HFS (Hierarchical File System)** - used by macOS
- **APFS (Apple File System)** - new file system for macOS and iOS

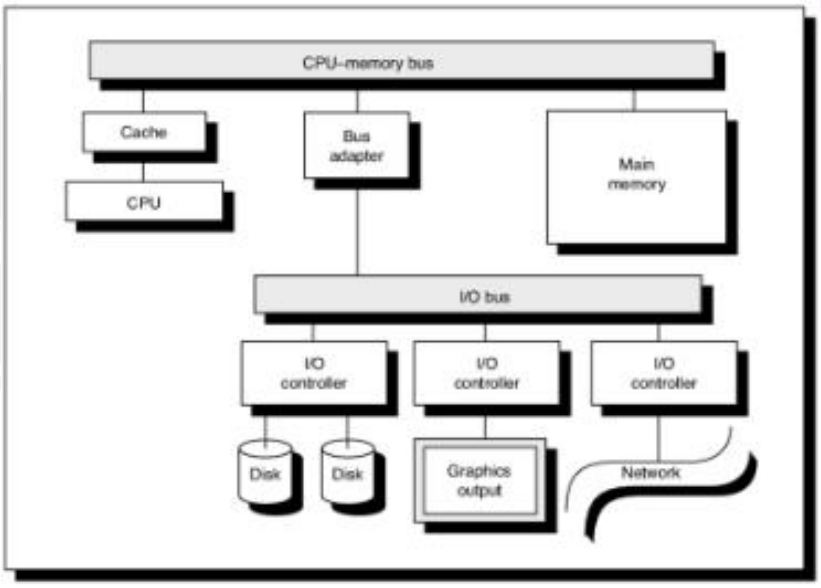


File structure / Types of File system

The four common terms of file systems are -

- **Field**
  - Basic element of data
  - Contains a single value
  - Has a particular length and data type
- **Record**
  - It is a collection of related fields
  - It is treated as a unit
- **File**
  - It is a collection of similar records
  - Treated as a single entity
  - Has file names
  - Access to file may be restricted or unrestricted
- **Database**
  - Collection of related data.
  - Relationship exists among elements.

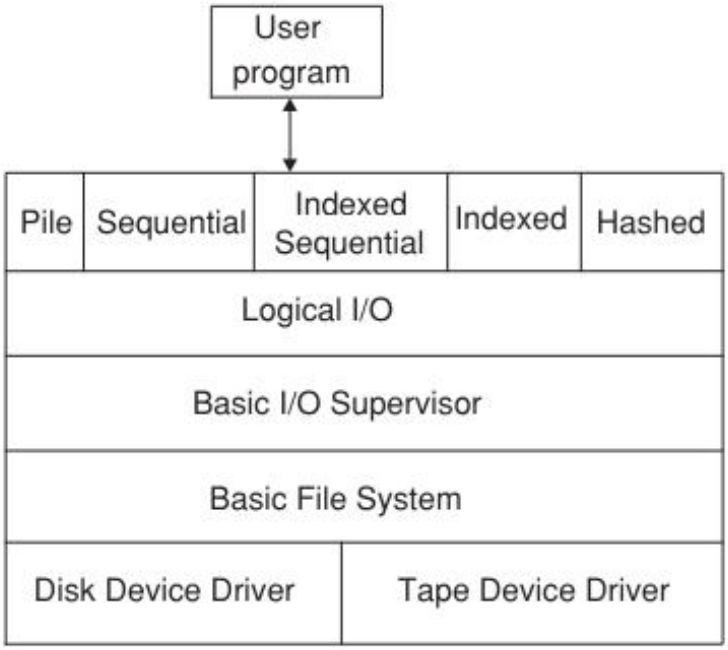
File Management Systems



It is a set of system software that provide services to users and applications in the use of files. Objectives of file management systems are -

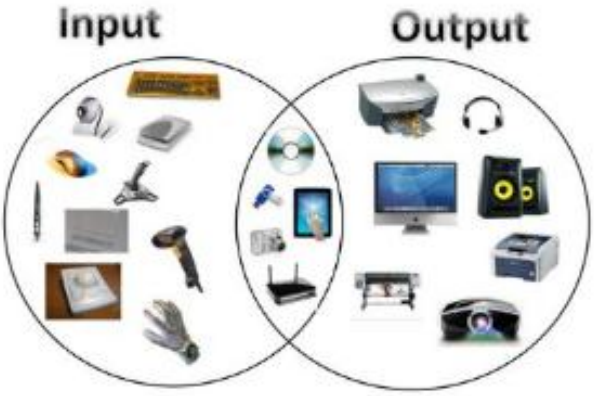
- To meet the data management needs and requirements of the user including storage of data.
- To guarantee, that the data in the file are valid.
- To optimize performance (i.e., throughput, response time).
- To provide Input/Output (I/O) support for a variety of storage device types.
- To minimize or eliminate the potential for lost or destroyed data.
- To provide a standardized set of I/O interface routines.
- To provide I/O support for multiple users.

File System Architecture



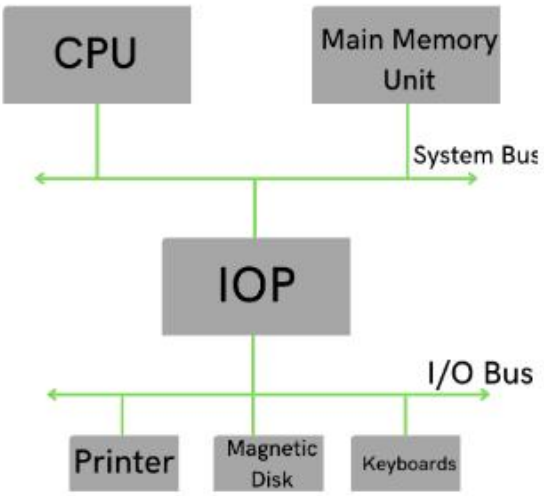
What do you understand by I/O

- I/O (Input/Output) in an operating system refers to the communication between a computer and the outside world.
- It involves the transfer of data between the computer's main memory and external devices like keyboards, mice, printers, scanners, disks, and network interfaces.
- The operating system plays a crucial role in managing and controlling these I/O operations.

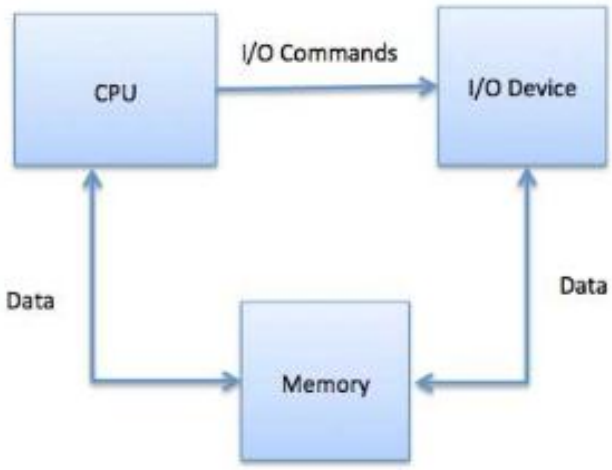


I/O systems

- The I/O (input/output) system in an operating system is responsible for managing the flow of data between the computer's main memory and external devices
- such as storage devices, network interfaces, and input/output peripherals.



Key components of the I/O system



- **Device Drivers** - specialized software components that allow the operating system to communicate with hardware devices, translating OS commands into device-specific instructions.
- **I/O subsystem** - manages the input and output operations of the system, including device communication, data transfer, and request handling.
- **Buffering and Caching** - temporarily holds data during transfer between devices or processes, while caching stores frequently accessed data to speed up future requests.
- **Interrupt handling** - involves detecting and responding to hardware or software events by pausing current processes and executing interrupt service routines (ISRs).
- **Asynchronous I/O** - allows processes to perform other tasks while waiting for I/O operations to complete, improving efficiency and responsiveness.
- **Spooling and Scheduling** - manages multiple I/O requests by queuing them for sequential processing, while scheduling determines the order and timing of these requests.

Question Time!

- **Question:** Which component of the I/O system handles the communication between the CPU and peripherals devices ?