* 333333323UGFHG

**Unit 3 Process Management**

### Process -:

In an operating system, a process is a unit of activity that represents the execution of a program or a task. It is an instance of a computer program that is being executed and managed by the operating system.

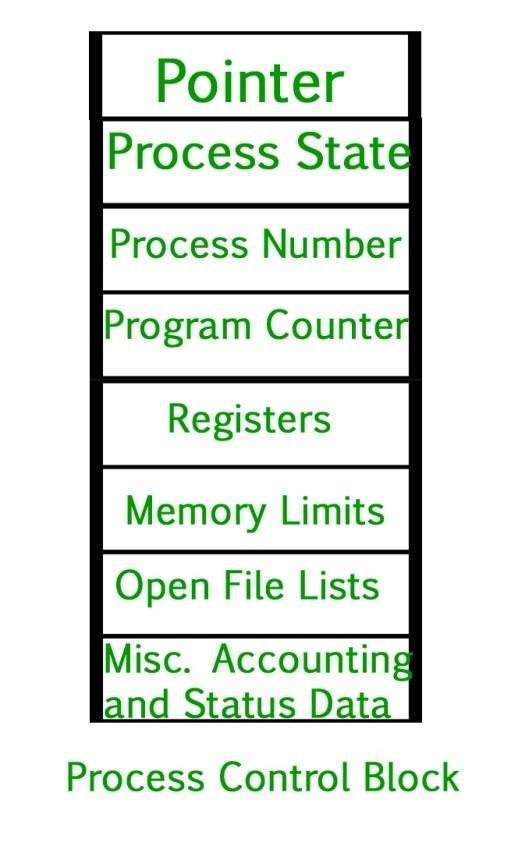
### Process States -:

A process has several stages that it passes through from beginning to end. There must be a minimum of five states.

1. **New :-**A new process is Create
2. **Ready:- :** New -> Ready to run. After the creation of a process,
3. **Waiting**:- A process is waiting from such as I/O complete be assigne a new process
4. **Running:-**A process is running sates
5. **Terminated:**-A process is finish

**Process Control Block**

The Process Control Block (PCB), also known as the Task Control Block (TCB), is a data structure used by an operating system to manage and track information about a specific process. . It allows the operating system to maintain and retrieve the necessary information for process scheduling, context switching, resource allocation, and interprocess communication



1. **Process ID (PID):** A unique identifier assigned to each process in the system
2. **Process State:** Indicates the current state of the process, such as running, ready, blocked, or terminated
3. **Program Counter (PC):** The Program Counter holds the address of the next instruction to be executed by the process. It allows the operating system to keep track of the execution progress of the process.
4. **CPU Registers**: The PCB contains the values of various CPU registers associated with the process, such as the accumulator, stack pointer,

**What is Process Scheduling ?**

Process scheduling is a fundamental function of an operating system that involves determining the order and allocation of CPU time to processes in a multitasking environment. It is the process by which the operating system selects and assigns processes for execution on the CPU.

Process scheduling aims to achieve the following goals:

**1.Fairness**: The scheduler ensures that each process gets a fair share of CPU time,

**2.Efficiency**: The scheduler aims to maximize CPU utilization,

**3.Responsiveness**: The scheduler strives to provide a responsive system by minimizing the time it takes for processes to start execution.

**4.Resource Optimization**: Process scheduling considers various system resources, such as CPU, memory, and I/O devices,

* **Scheduling Queue**

### Ready Queue:

The ready queue is a fundamental concept in process scheduling within an operating system. It is a data structure that holds the processes that are ready and waiting to be executed on the CPU.

Purpose, Organization, Scheduling Decision, Process State

1. **Device Queue:**

The device queue, also known as the I/O queue or waiting queue, is a data structure used by the operating system to manage processes that are waiting for access to I/O devices. It holds processes that are waiting for I/O operations to complete before they can proceed with their execution.

Purpose, Organization, Process State, I/O Scheduling, Device Completion

**Schedulers**

Schedulers in an operating system are responsible for making decisions about process execution, resource allocation, and process management. They determine which processes should run, in what order, and for how long. Schedulers play a crucial role in achieving efficient utilization of system resources, responsiveness, fairness,

* **Type of process schedulers**

There are three type of scheduler

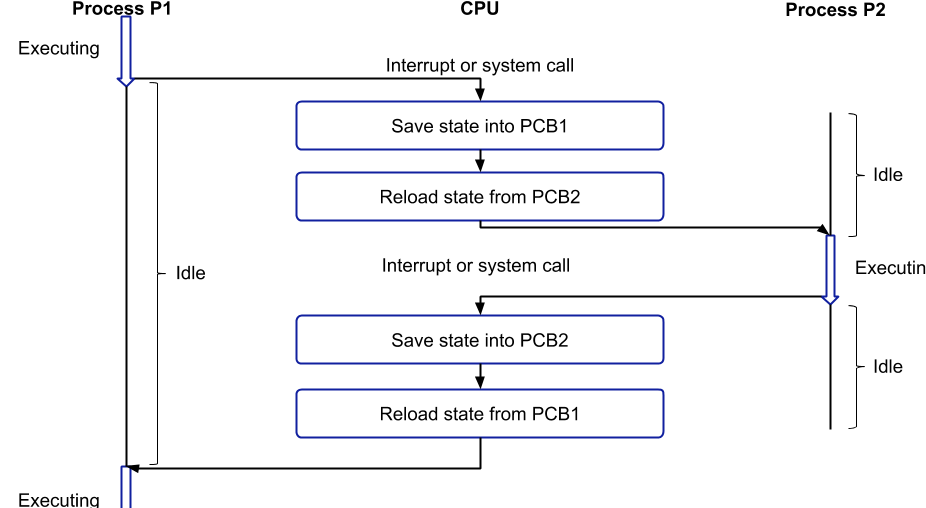
1**.Long-team-scheduler** :- The long-term scheduler decides which processes are admitted into the system from the pool of waiting processes

**2.Median-team-scheduler** :- The medium-term scheduler, also known as the swapping scheduler, is an optional scheduler found in some operating systems.

3.**Short-team-scheduler** :- The short-term scheduler determines which processes in the ready queue should be allocated CPU time for execution. It is responsible for making frequent and fast scheduling decisions,

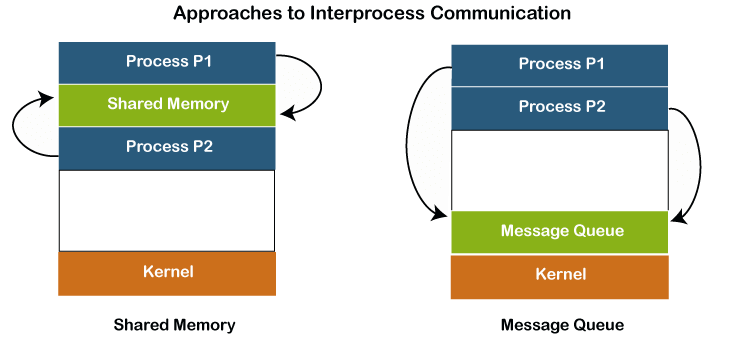
**Context Switch**

A context switch is a fundamental operation performed by an operating system to save and restore the state of a process or thread when there is a need to switch the CPU's execution from one process to another. It involves storing the current execution context of a process, i ncluding its register values, program counter, and other relevant information

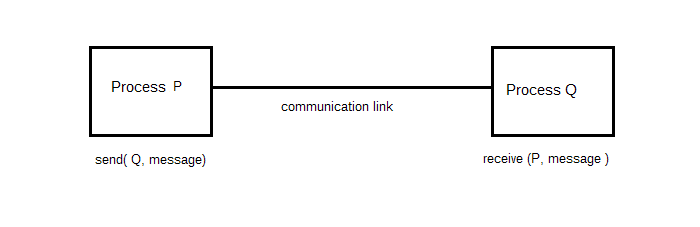


**Inter-process Communication**

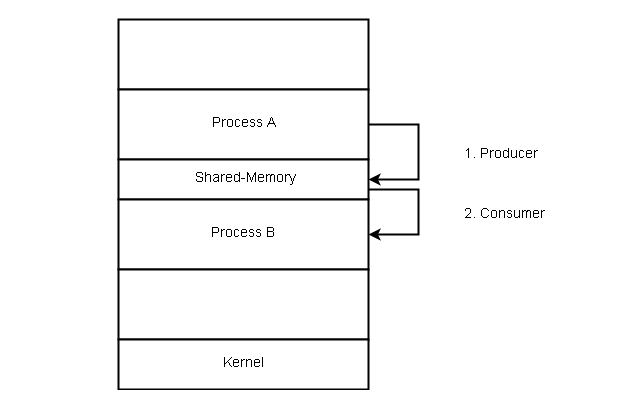
Inter-process communication (IPC) refers to the mechanisms and techniques used by operating systems to enable communication and data exchange between different processes running concurrently. IPC plays a vital role in coordinating and synchronizing the activities of various processes, **Shared Memory, Message Passing, Pipes and FIFOs, Sockets**



**Message passing** is a form of inter-process communication (IPC) that involves the exchange of messages between processes. In this method, processes communicate by sending and receiving messages



**Shared memory** is an IPC mechanism that allows multiple processes to access and share a common region of memory. In shared memory communication



**Threads**

In operating systems, a thread refers to a lightweight unit of execution within a process. A thread is also known as a "lightweight process" because it shares the same memory space as other threads within the same process, allowing them to access and modify the same data.

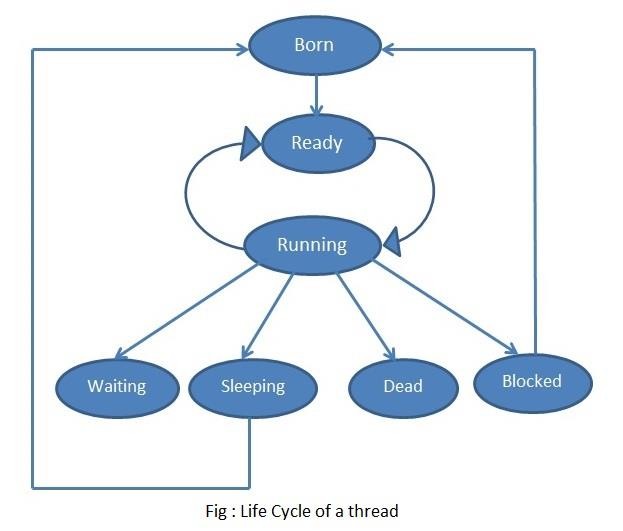
Threads are used to achieve concurrent execution within a single process. Unlike processes, **Concurrency, Shared Resources, Lightweight, Scheduling**

* Advantage:-

Responsiveness ,Efficiency ,Resource Sharing ,Modularity

**Thread Lifecycle –**

The life cycle of a thread describes the different stages that a thread goes through during its existence. The thread life cycle typically includes the following states:



**New :** In the new state,

**Runnable :** Once the thread is ready to execute,

**Running :** When a thread is selected by the scheduler to execute on a CPU core,

**Blocked :** Threads can transition to the blocked (or waiting) state if they need to wait for a particular event or condition to occur

**Terminated :** A thread enters the terminated state when it finishes its execution

* **Types of Threads**

**1.User-Level Threads (ULT) :** User-level threads are managed entirely by user-level thread libraries or programming language runtimes, without direct support from the operating system kernel.

**2.Kernel-Level Threads (KLT) :** Kernel-level threads are managed by the operating system kernel. The kernel provides explicit support for thread creation, scheduling,

**Multi-threading Model**

A multithreading model in an operating system (OS) allows multiple threads of execution to exist within a single process. Multithreading enables concurrent execution of multiple threads,

**Many-to-One (User-Level Threads) :** In this model, multiple user-level threads are managed by a single kernel-level thread

**One-to-One (Kernel-Level Threads) :** In this model, each user-level thread is mapped to a separate kernel-level thread by the operating system. Kernel- level threads are scheduled and managed by the OS scheduler.

**Many-to-Many (Hybrid Model) :** The many-to-many model is a hybrid approach that combines aspects of the previous two models. In this model, multiple user-level threads are mapped to an equal or smaller number of kernel- level threads

**Process Commands**

**ps command:-**

The `ps` command is a commonly used command in operating systems, especially UNIX-like systems, to provide information about the currently running processes.

ps -u username

ps -p PID

**wait command:-**

The "wait" command in an operating system is used to make a process wait or pause its execution until a specific event or condition occurs

**sleep command:-**

The "sleep" command is a utility found in various operating systems and programming languages that allows for a specified delay or pause in the execution of a program or script.

Syntax: sleep NUMBER[SUFFIX]…

sleep OPTION

**kill command:-**

the `kill` command is a system utility used to terminate or send signals to processes. Its primary purpose is to allow users or system administrators to manage and control running processes. The `kill` command is available in various Unix-like operating systems, including Linux.

The basic syntax of the `kill` command is as follows

#### kill [options] <PID>