Unit 2: Introduction of Process Management

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process. A process is an 'active' entity instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

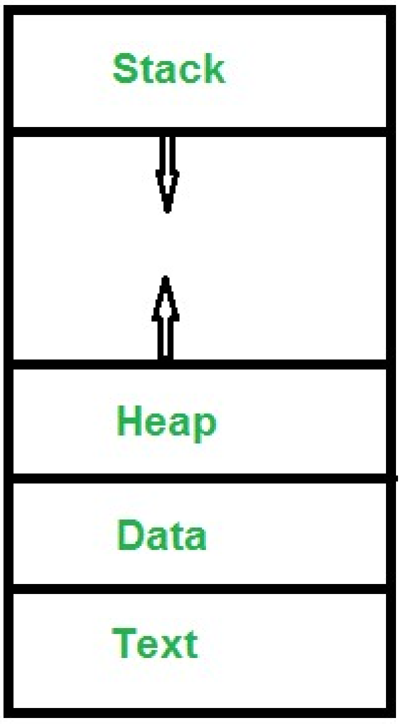
Process Management

Process management is a key part of an operating system. It controls how processes are carried out, and controls how your computer runs by handling the active processes. This includes stopping processes, setting which processes should get more attention, and many more. You can manage processes on your own computer too.

The OS is responsible for managing the start, stop, and scheduling of processes, which are programs running on the system. The operating system uses a number of methods to prevent deadlocks, facilitate inter-process communication, and synchronize processes. Efficient resource allocation, conflict-free process execution, and optimal system performance are all guaranteed by competent process management. This essential component of an operating system enables the execution of numerous applications at once, enhancing system utilization and responsiveness.

How Does a Process Look Like in Memory?

A process in memory is divided into several distinct sections, each serving a different purpose. Here's how a process typically looks in memory:

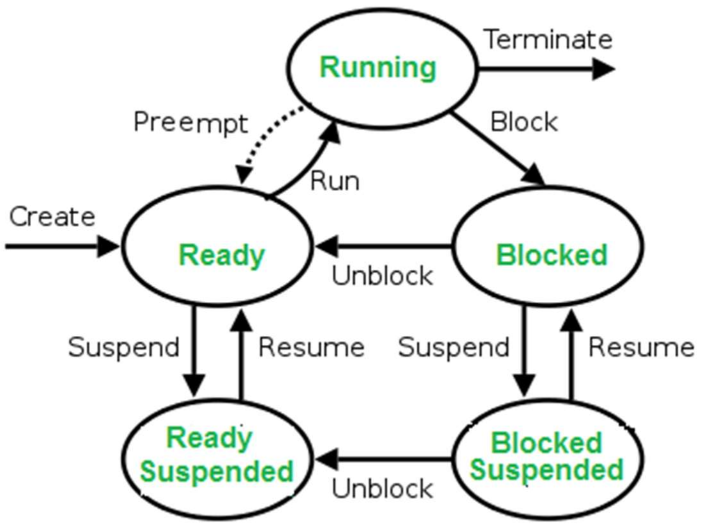


Text Section: A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the Program Counter.
Stack: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
Data Section: Contains the global variable.
Heap Section: Dynamically memory allocated to process during its run time.

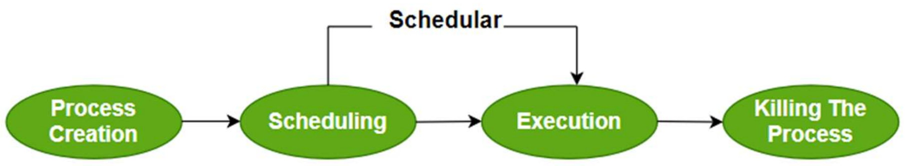
Characteristics of a Process
A process has the following attributes.
1. **Process Id:** A unique identifier assigned by the operating system.
2. **Process State:** Can be ready, running, etc.
3. **CPU Registers:** Like the Program Counter (CPU registers must be saved and restored when a process is swapped in and out of the CPU)
4. **Accounts Information:** Amount of CPU used for process execution, time limits, execution ID, etc
5. **I/O Status Information:** For example, devices allocated to the process, open files, etc
6. **CPU Scheduling Information:** For example, Priority (Different processes may have different priorities, for example, a shorter process assigned high priority in the shortest job first scheduling)

All of the above attributes of a process are also known as the context of the process. Every process has its own process control block(PCB), i.e. each process will have a unique PCB. All of the above attributes are part of the PCB.

States of Process
A process is in one of the following states:
1. **New:** Newly Created Process (or) being-created process.
2. **Ready:** After the creation process moves to the Ready state, i.e. the process is ready for execution.
3. **Running:** Currently running process in CPU (only one process at a time can be under execution in a single processor).
4. **Wait (or Block):** When a process requests I/O access.
5. **Complete (or Terminated):** The process completed its execution.
6. **Suspended Ready:** When the ready queue becomes full, some processes are moved to a suspended ready state
7. **Suspended Block:** When the waiting queue becomes full.



Process Operations
Process operations in an operating system refer to the various activities the OS performs to manage processes. These operations include process creation, process scheduling, execution and killing the process. Here are the key process operations:



Process Creation: Process creation in an operating system (OS) is the act of generating a new process. This new process is an instance of a program that can execute independently.

Scheduling: Once a process is ready to run, it enters the “ready queue.” The scheduler’s job is to pick a process from this queue and start its execution.

Execution: Execution means the CPU starts working on the process. During this time, the process might:

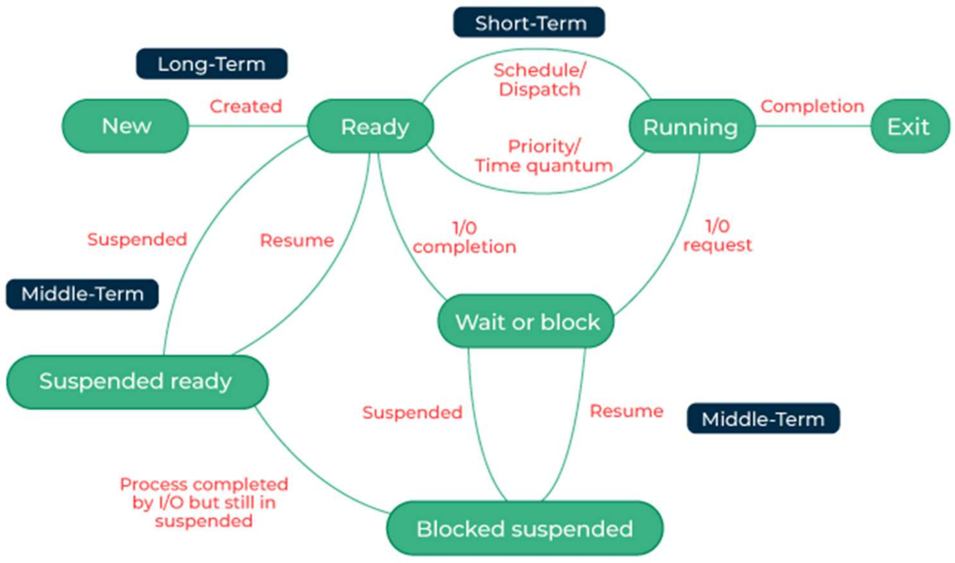
- Move to a waiting queue if it needs to perform an I/O operation.
- Get blocked if a higher-priority process needs the CPU.

Killing the Process: After the process finishes its tasks, the operating system ends it and removes its Process Control Block (PCB).

Process Schedulers in Operating System
In computing, a process is the instance of a computer program that is being executed by one or many threads. Scheduling is important in many different computer environments. One of the most important areas of scheduling is which programs will work on the CPU. This task is handled by the Operating System (OS) of the computer and there are many different ways in which we can choose to configure programs.

Process schedulers are fundamental components of operating systems responsible for deciding the order in which processes are executed by the CPU. In simpler terms, they manage how the CPU allocates its time among multiple tasks or processes that are competing for its attention.

Process Scheduling
Process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process based on a particular strategy.
Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.



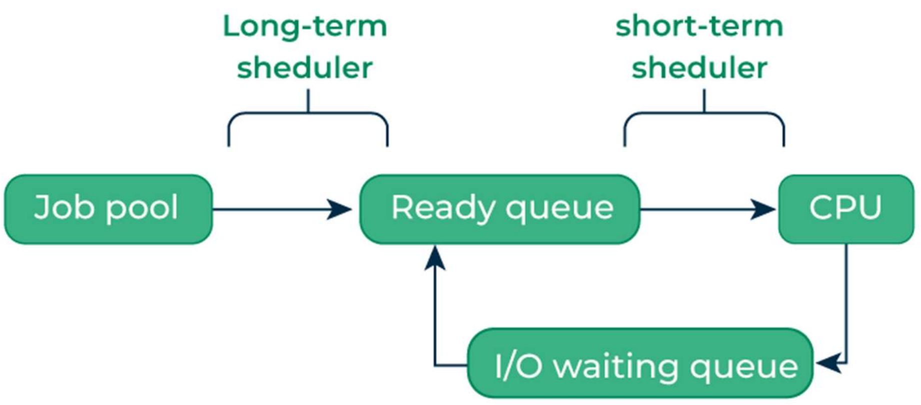
Scheduling falls into one of two categories:

1. **Non-Preemptive:** In this case, a process's resource cannot be taken before the process has finished running. When a running process finishes and transitions to a waiting state, resources are switched.
2. **Preemptive:** In this case, the OS assigns resources to a process for a predetermined period. The process switches from running state to ready state or from waiting state to ready state during resource allocation. This switching happens because the CPU may give other processes priority and substitute the currently active process for the higher priority process.

Types of Process Schedulers
There are three types of process schedulers:
1. Long Term or Job Scheduler
It brings the new process to the 'Ready State'. It controls the Degree of Multi-programming, i.e., the number of processes present in a ready state at any point in time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O-bound tasks are which use much of their time in input and output operations while CPU-bound processes are which spend their time on the CPU. The job scheduler increases efficiency by

maintaining a balance between the two. They operate at a high level and are typically used in batch-processing systems.

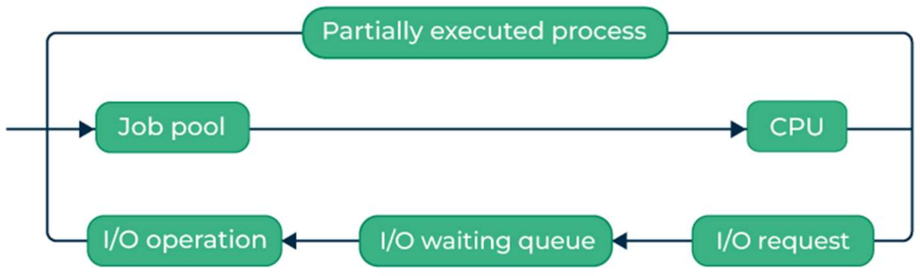
2. Short-Term or CPU Scheduler
It is responsible for selecting one process from the ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring no starvation due to high burst time processes.



The dispatcher is responsible for loading the process selected by the Short-term scheduler on the CPU (Ready to Running State) Context switching is done by the dispatcher only. A dispatcher does the following:

- Switching context.
- Switching to user mode.
- Jumping to the proper location in the newly loaded program.

3. Medium-Term Scheduler
It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.



Some Other Schedulers

I/O Schedulers: I/O schedulers are in charge of managing the execution of I/O operations such as reading and writing to discs or networks. They can use various algorithms to determine the order in which I/O operations are executed, such as FCFS (First-Come, First-Served) or RR (Round Robin).

Real-Time Schedulers: In real-time systems, real-time schedulers ensure that critical tasks are completed within a specified time frame. They can prioritize and schedule tasks using various algorithms such as EDF (Earliest Deadline First) or RM (Rate Monotonic).

Comparison Among Scheduler

Long Term Scheduler	Short Term Schedular	Medium Term Scheduler
It is a job scheduler	It is a CPU scheduler	It is a process-swapping scheduler.
Generally, Speed is lesser than short term scheduler	Speed is the fastest among all of them.	Speed lies in between both short and long-term schedulers.
It controls the degree of multiprogramming	It gives less control over how much multiprogramming is done.	It reduces the degree of multiprogramming.
It is barely present or nonexistent in the time-sharing system.	It is a minimal time-sharing system.	It is a component of systems for time sharing.

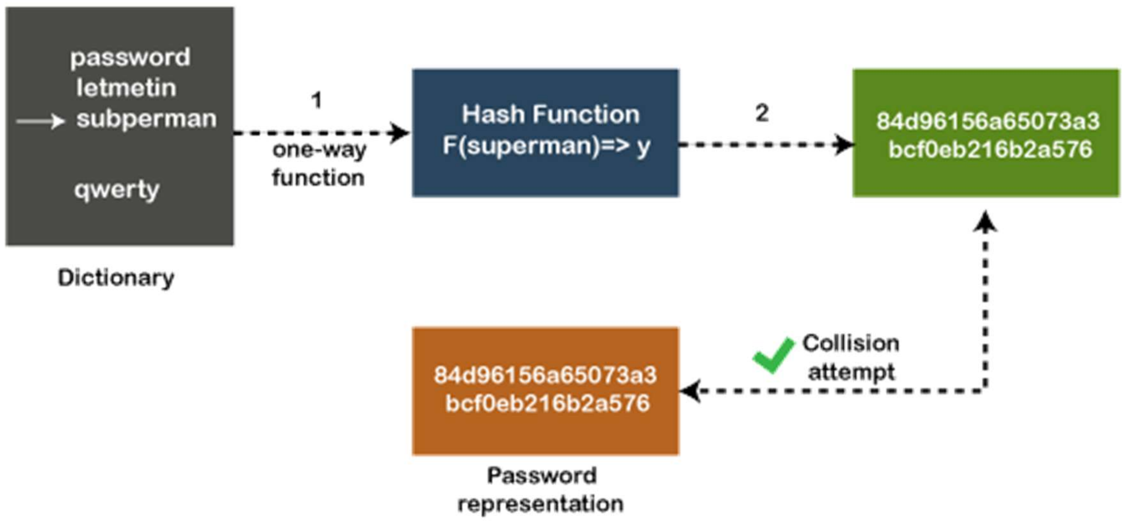
Long Term Scheduler	Short Term Schedular	Medium Term Scheduler
It can re-enter the process into memory, allowing for the continuation of execution.	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Inter Process Communication: In general, Inter Process Communication is a type of mechanism usually provided by the operating system (or OS). The main aim or goal of this mechanism is to provide communications in between several processes. In short, the intercommunication allows a process letting another process know that some event has occurred.

Let us now look at the general definition of inter-process communication, which will explain the same thing that we have discussed above.

Definition: "Inter-process communication is used for exchanging useful information between numerous threads in one or more processes (or programs)."

To understand inter process communication, you can consider the following given diagram that illustrates the importance of inter-process communication:



Role of Synchronization in Inter Process Communication

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.

These are the following methods that used to provide the synchronization:

- Mutual Exclusion
- Semaphore
- Barrier
- Spinlock

1. **Mutual Exclusion:-** It is generally required that only one process thread can enter the critical section at a time. This also helps in synchronization and creates a stable state to avoid the race condition.
2. **Semaphore:-** Semaphore is a type of variable that usually controls the access to the shared resources by several processes. Semaphore is further divided into two types which are as follows:

- Binary Semaphore
- Counting Semaphore

3. **Barrier:-**

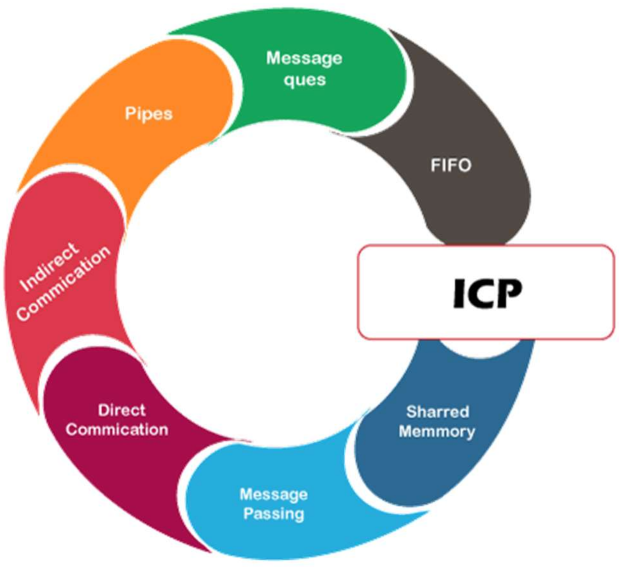
A barrier typically not allows an individual process to proceed unless all the processes does not reach it. It is used by many parallel languages, and collective routines impose barriers.

4. **Spinlock:-**

Spinlock is a type of lock as its name implies. The processes are trying to acquire the spinlock waits or stays in a loop while checking that the lock is available or not. It is known as busy waiting because even though the process active, the process does not perform any functional operation (or task).

Approaches to Interprocess Communication

We will now discuss some different approaches to inter-process communication which are as follows:



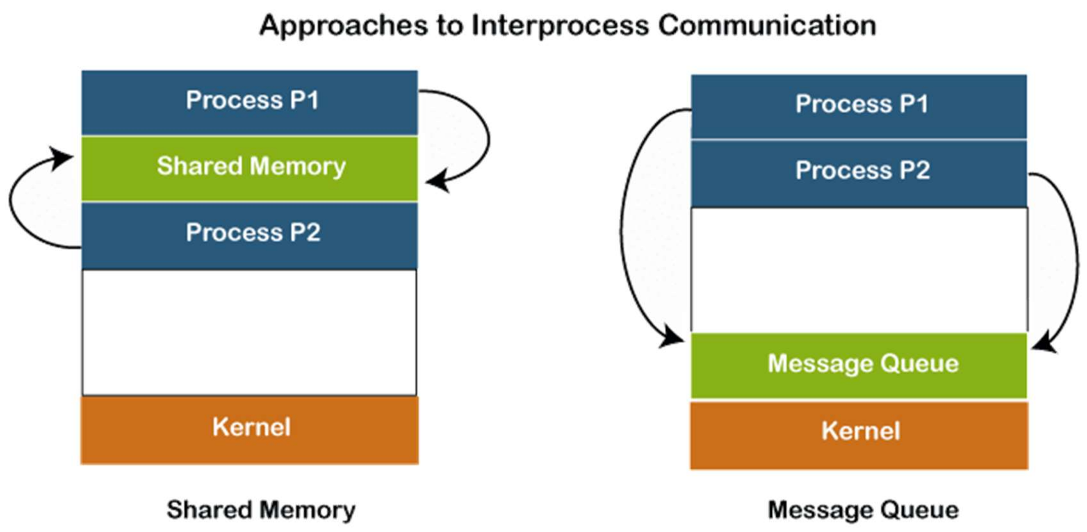
These are a few different approaches for Inter- Process Communication:

- Pipes
- Shared Memory
- Message Queue
- Direct Communication
- Indirect communication
- Message Passing
- FIFO

Pipe:- The pipe is a type of data channel that is unidirectional in nature. It means that the data in this type of data channel can be moved in only a single direction at a time. Still, one can use two-channel of this type, so that he can able to send and receive data in two processes. Typically, it uses the standard methods for input and output. These pipes are used in all types of POSIX systems and in different versions of window operating systems as well.

Shared Memory:- It can be referred to as a type of memory that can be used or accessed by multiple processes simultaneously. It is primarily used so that the processes can communicate with each other. Therefore the shared memory is used by almost all POSIX and Windows operating systems as well.

Message Queue:- In general, several different messages are allowed to read and write the data to the message queue. In the message queue, the messages are stored or stay in the queue unless their recipients retrieve them. In short, we can also say that the message queue is very helpful in inter-process communication and used by all operating systems.



Message Passing:- It is a type of mechanism that allows processes to synchronize and communicate with each other. However, by using the message passing, the processes can communicate with each other without restoring the hared variables.

Usually, the inter-process communication mechanism provides two operations that are as follows:

- send (message)
- received (message)

Direct Communication:- In this type of communication process, usually, a link is created or established between two communicating processes. However, in every pair of communicating processes, only one link can exist.

Indirect Communication:- Indirect communication can only exist or be established when processes share a common mailbox, and each pair of these processes shares multiple communication links. These shared links can be unidirectional or bi-directional.

FIFO:- It is a type of general communication between two unrelated processes. It can also be considered as full-duplex, which means that one process can communicate with another process and vice versa.

Some other different approaches

Socket:- It acts as a type of endpoint for receiving or sending the data in a network. It is correct for data sent between processes on the same computer or data sent between different computers on the same network. Hence, it used by several types of operating systems.

File:- A file is a type of data record or a document stored on the disk and can be acquired on demand by the file server. Another most important thing is that several processes can access that file as required or needed.

Signal:- As its name implies, they are a type of signal used in inter process communication in a minimal way. Typically, they are the massages of systems that are sent by one process to another. Therefore, they are not used for sending data but for remote commands between multiple processes.

Usually, they are not used to send the data but to remote commands in between several processes.

Need of interprocess communication

There are numerous reasons to use inter-process communication for sharing the data. Here are some of the most important reasons that are given below:

- It helps to speedup modularity
- Computational
- Privilege separation
- Convenience
- Helps operating system to communicate with each other and synchronize their actions as well.