**Unit-2**

**Introduction of Process Management**

Process Management for a single tasking or batch processing system is easy as only one [process](https://www.geeksforgeeks.org/process-in-operating-system/) is active at a time. With multiple processes (multiprogramming or multitasking) being active, the process management becomes complex as a CPU needs to be efficiently utilized by multiple processes. Multiple active processes can may share resources like memory and may communicate with each other. This further makes things complex as an Operating System has to do process synchronization.

Please remember the main advantages of having [multiprogramming](https://www.geeksforgeeks.org/multiprogramming-in-operating-system/) are system responsiveness and better CPU utilization. We can run multiple processes in interleaved manner on a single CPU. For example, when the current process is getting busy with IO, we assign CPU to some other process.

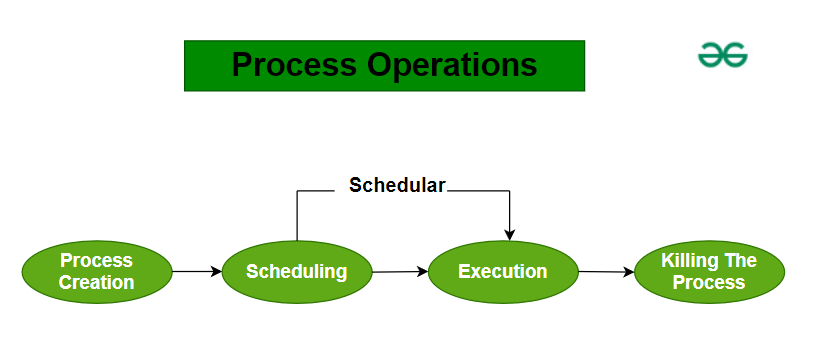
**Process Management Tasks**

Process management is a key part in operating systems with multi-programming or multitasking.

* [Process Creation and Termination](https://www.geeksforgeeks.org/process-creation-and-deletions-in-operating-systems/) : Process creation involves creating a Process ID, setting up Process Control Block, etc. A process can be terminated either by the operating system or by the parent process. Process termination involves clearing all resources allocated to it.
* [CPU Scheduling](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/) : In a multiprogramming system, multiple processes need to get the CPU. It is the job of Operating System to ensure smooth and efficient execution of multiple processes.
* [Deadlock Handling](https://www.geeksforgeeks.org/handling-deadlocks/) : Making sure that system does not reach a state where two or processes cannot proceed due to a cycling dependency on each other.
* [Inter-Process Communication](https://www.geeksforgeeks.org/inter-process-communication-ipc/) : Operating System provides facilities such as shared memory and message passing for cooperating processes to communicate.
* [Process Synchronization](https://www.geeksforgeeks.org/introduction-of-process-synchronization/) : Process Synchronization is the coordination of execution of multiple processes in a multiprogramming system to ensure that they access shared resources (like memory) in a controlled and predictable manner.

**Process Operations**

Please remember a [process goes through different states](https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/)before termination and these state changes require different operations on processes by an operating system. These operations include process creation, process scheduling, execution and killing the process. Here are the key process operations:



*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).

**Advantages of Process Management**

* **Running Multiple Programs:** Process management lets you run multiple applications at the same time, for example, listen to music while browsing the web.
* **Process Isolation:** It ensures that different programs don’t interfere with each other, so a problem in one program won’t crash another.
* **Fair Resource Use:** It makes sure resources like CPU time and memory are shared fairly among programs, so even lower-priority programs get a chance to run.
* **Smooth Switching:** It efficiently handles switching between programs, saving and loading their states quickly to keep the system responsive and minimize delays.

**Disadvantages of Process Management**

* **Overhead:** Process management uses system resources because the OS needs to keep track of various data structures and scheduling queues. This requires CPU time and memory, which can affect the system’s performance.
* **Complexity:** Designing and maintaining an OS is complicated due to the need for complex scheduling algorithms and resource allocation methods.
* **Deadlocks:** To keep processes running smoothly together, the OS uses mechanisms like semaphores and mutex locks. However, these can lead to deadlocks, where processes get stuck waiting for each other indefinitely.
* **Increased Context Switching:** In [multitasking](https://www.geeksforgeeks.org/difference-between-multitasking-multithreading-and-multiprocessing/) systems, the OS frequently switches between processes. Storing and loading the state of each process (context switching) takes time and computing power, which can slow down the system.

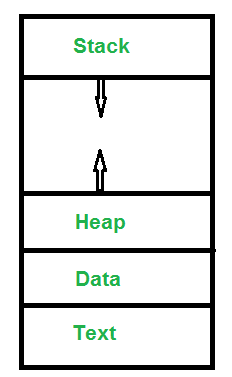
**Operations on Processes**

Process operations refer to the actions or activities performed on processes in an operating system. These operations include creating, terminating, suspending, resuming, and communicating between processes. Operations on processes are crucial for managing and controlling the execution of programs in an operating system.

Operations on processes are fundamental to the functioning of operating systems, enabling effective flow of program execution and resource allocation. The lifecycle of a process includes several critical operations: creation, scheduling, blocking, preemption, and termination. Each operation plays a vital role In ensuring that processes are efficiently managed, allowing for multitasking and optimal resource utilization. In this article, we will discuss various operations on Process.

**What is a Process**

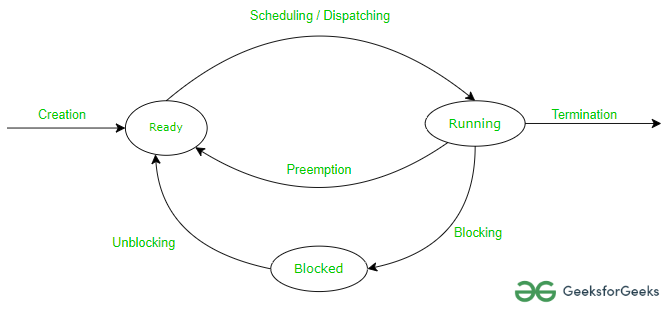
A process is an activity of executing a program. It is a program under execution. Every process needs certain resources to complete its task. Processes are the programs that are dispatched from the ready state and are scheduled in the CPU for execution. [PCB (Process Control Block)](https://www.geeksforgeeks.org/process-control-block-in-os/)holds the context of the process. A process can create other processes which are known as Child Processes. The process takes more time to terminate, and it is isolated means it does not share the memory with any other process. The process can have the following [states](https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/) new, ready, running, waiting, terminated, and suspended.



* **Text**: A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the [Program Counter](https://www.geeksforgeeks.org/what-is-program-counter) .
* **Stack**: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
* **Data**: Contains the global variable.
* **Heap**: [Dynamically memory allocated](https://www.geeksforgeeks.org/what-is-dynamic-memory-allocation) to process during its run time.

**Operation on a Process**

The execution of a process is a complex activity. It involves various operations. Following are the operations that are performed while execution of a process:



**1. Creation**

This is the initial step of the process execution activity. Process creation means the construction of a new process for execution. This might be performed by the system, the user, or the old process itself. There are several events that lead to the process creation. Some of the such events are the following:

* When we start the computer, the system creates several background processes.
* A user may request to create a new process.
* A process can create a new process itself while executing.
* The batch system takes initiation of a batch job.

**2. Scheduling/Dispatching**

The event or activity in which the state of the process is changed from ready to run. It means the operating system puts the process from the ready state into the running state. Dispatching is done by the operating system when the resources are free or the process has higher priority than the ongoing process. There are various other cases in which the process in the running state is preempted and the process in the ready state is dispatched by the [operating system.](https://www.geeksforgeeks.org/what-is-an-operating-system/)

**3. Blocking**

When a process invokes an input-output system call that blocks the process, and operating system is put in block mode. Block mode is basically a mode where the process waits for input-output. Hence on the demand of the process itself, the operating system blocks the process and dispatches another process to the processor. Hence, in process-blocking operations, the operating system puts the process in a ‘waiting’ state.

**4. Preemption**

When a timeout occurs that means the process hadn’t been terminated in the allotted time interval and the next process is ready to execute, then the operating system preempts the process. This operation is only valid where [CPU scheduling](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/)supports preemption. Basically, this happens in priority scheduling where on the incoming of high priority process the ongoing process is preempted. Hence, in process preemption operation, the operating system puts the process in a ‘ready’ state.

**5. Process Termination**

Process termination is the activity of ending the process. In other words, process termination is the relaxation of computer resources taken by the process for the execution. Like creation, in termination also there may be several events that may lead to the process of termination. Some of them are:

* The process completes its execution fully and it indicates to the OS that it has finished.
* The operating system itself terminates the process due to service errors.
* There may be a problem in hardware that terminates the process.

**Inter Process Communication (IPC)**

Processes need to communicate with each other in many situations, for example, to count occurrences of a word in text file, output of grep command needs to be given to wc command, something like grep -o -i <word> <file> | wc -l. **Inter-Process Communication or IPC** is a mechanism that allows processes to communicate. It helps processes synchronize their activities, share information, and avoid conflicts while accessing shared resources.

**Types of Process**

* **Independent process:**An independent process is not affected by the execution of other processes. Independent processes are processes that do not share any data or resources with other processes. No inte-process communication required here.
* **Co-operating process:**Interact with each other and share data or resources. A co-operating process can be affected by other executing processes.Inter-process communication (IPC) is a mechanism that allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of cooperation between them.

**Client Server Communication in Operating System**

In an Operating System, Client Server Communication refers to the exchange of data and Services among multiple machines or processes. In Client client-server communication System one process or machine acts as a client requesting a service or data, and Another machine or process acts like a server for providing those Services or Data to the client machine. This Communication model is widely used for exchanging data among various computing environments like **Distributed Systems, Internet Applications, and Networking Application communication.**The communication between Server and Client takes place with different Protocols and mechanisms.

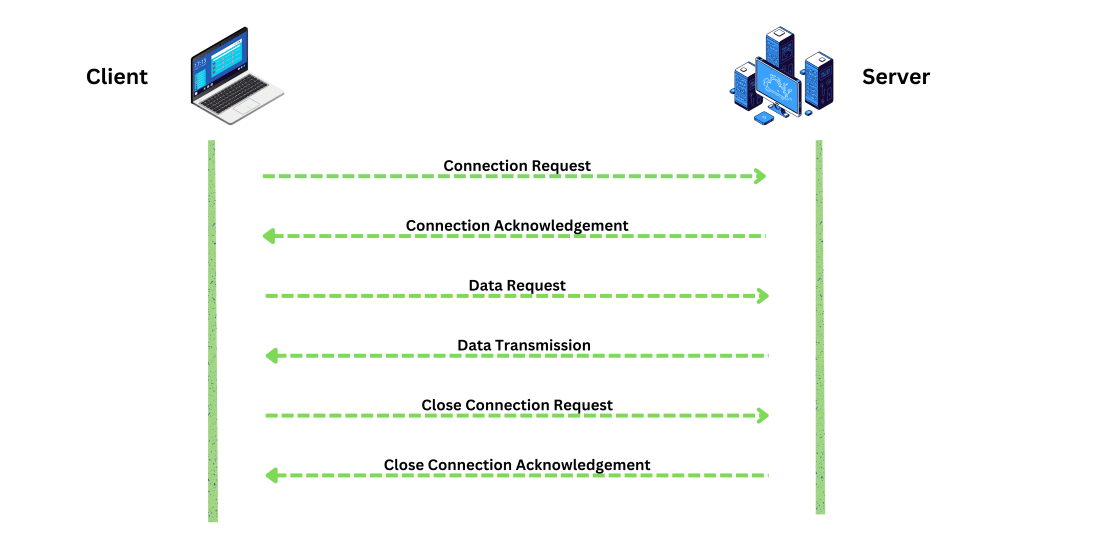
**Different Ways of Client-Server Communication**

**In Client Server Communication we can use different ways.**

1. Sockets Mechanism
2. Remote Procedure Call
3. Message Passing
4. Inter-process Communication
5. Distributed File Systems

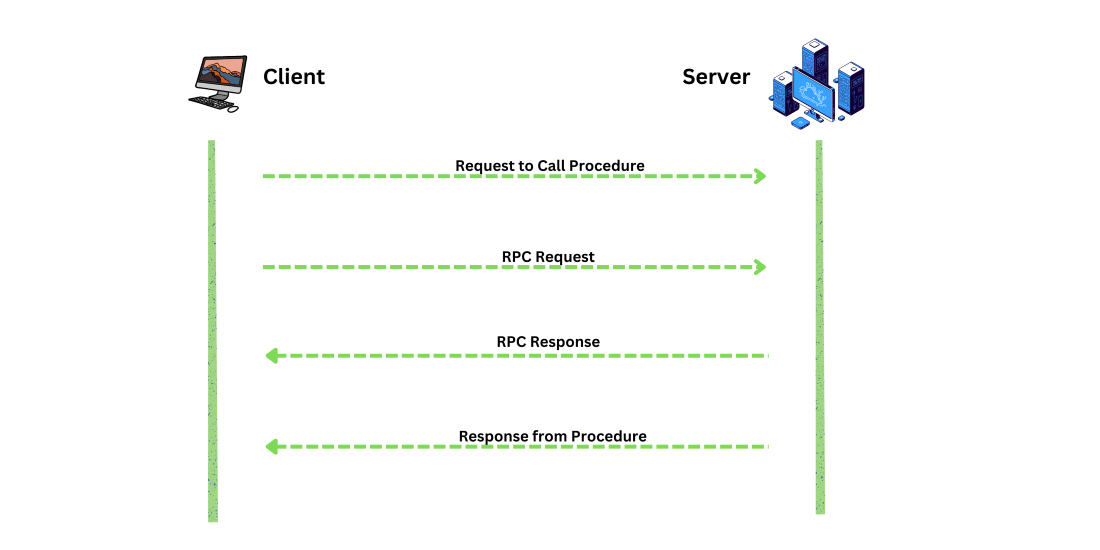
**Sockets Mechanism**

The Sockets are the End Points of Communication between two machines. They provide a way for processes to communicate with each other, either on the same on machine or over through Internet also possible. The Sockets enable the communication connection between Serthe er and the client to transfer data in a bidirectional way.

Client Server Communication using Sockets

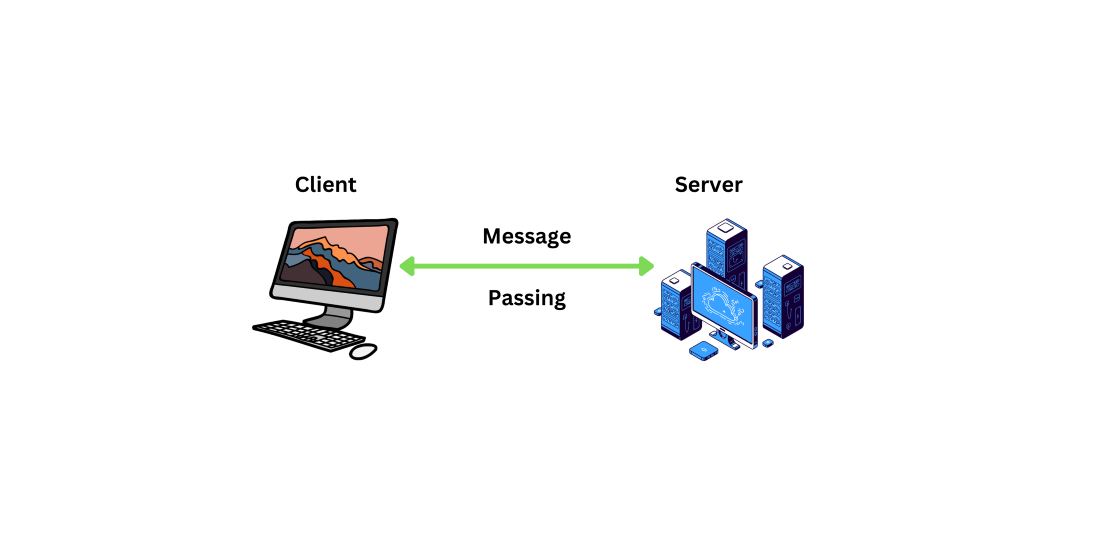
**Remote Procedure Call (PRC)**

Remote Procedure Call is a Protocol. A Protocol is set of Instructions. It allows a client to execute a procedure call on remote server, as if it is local procedure call. PRC is commonly used in Client Server communication Architecture. PRC Provide high level of abstraction to the programmer. In This The client Program issues a procedure call , which is translated into message that is sent over the network to the Server, The Server execute the call and send back to the Client Machine.

Remote Procedure Call Process

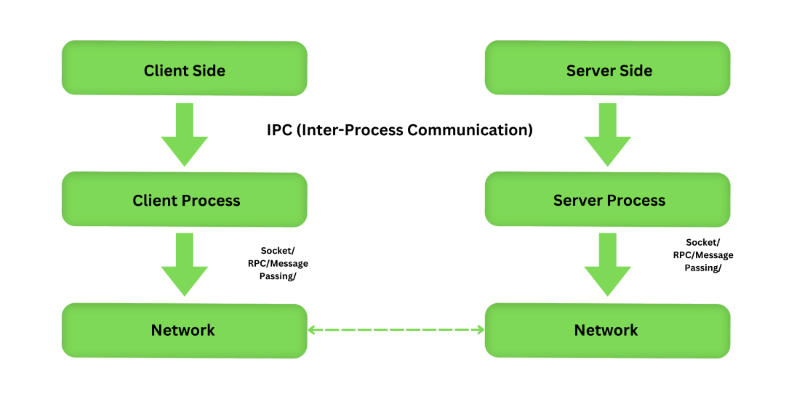
**Message Passing**

Message Passing is a communication Method. Where the machines communicated with each one by send and receiving the messages. This approach is commonly used in Parallel and Distributed Systems, This approach enables data exchange among the System.

Message Passing Process

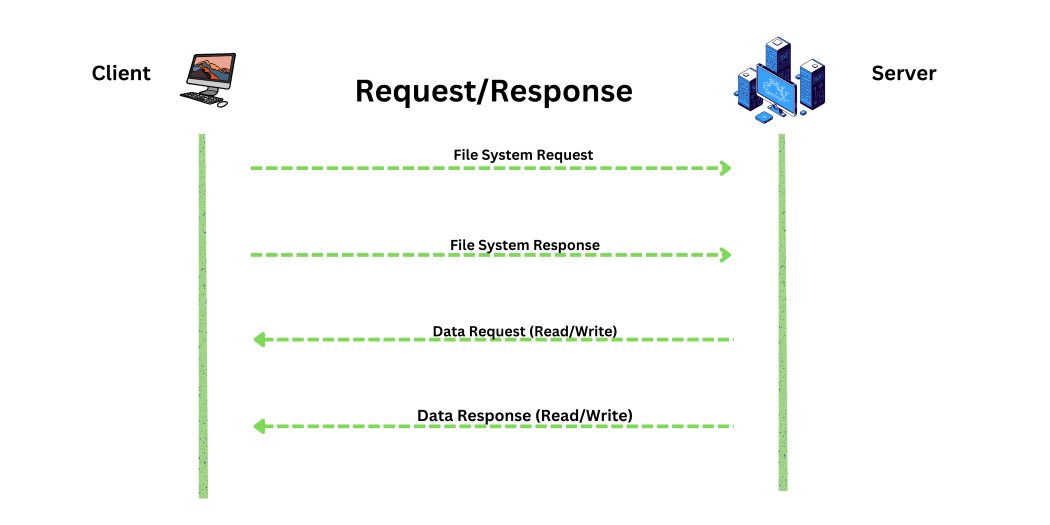
**Inter process Communication**

The Inter Process Communication also called **IPC.**It allows communication between processes within the same Machine. The IPC can enable data sharing and Synchronous between different processes running concurrently on an operating system. And it includes Sharing Memory, message queues, semaphores and pipes among others.

Inter process Communication Process

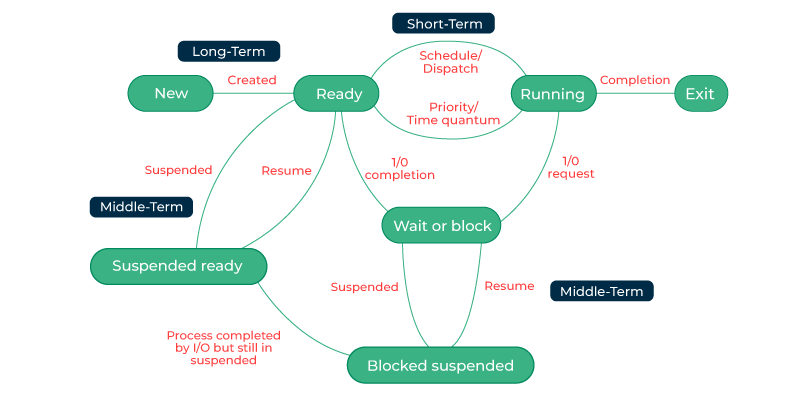
**Distributed File Systems**

Distributed File Systems provide access to files from multiple machines in network. Client can access and manipulate files stored on Remote Server, Through Standard Interface Example Network File System and Server Message Block.



**Process Schedulers in Operating System**

A process is the instance of a computer program in execution.

* Scheduling is important in operating systems with multiprogramming as multiple processes might be eligible for running at a time.
* One of the key responsibilities of an Operating System (OS) is to decide which programs will execute on the CPU.
* 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 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. Throughout its lifetime, a process moves between various [**scheduling queues**](https://www.geeksforgeeks.org/types-of-scheduling-queues/), such as the ready queue, waiting queue, or devices queue.
* 

**Categories of Scheduling**

Scheduling falls into one of two categories:

* **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.
* **Preemptive:**In this case, the OS can switch a process from running state to ready state. 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**

Long Term Scheduler loads a process from disk to main memory for execution. The new process to the ‘Ready State’.

* It mainly moves processes from[Job Queue](https://www.geeksforgeeks.org/types-of-scheduling-queues/)to[Ready Queue](https://www.geeksforgeeks.org/types-of-scheduling-queues/).
* It controls the Degree of [Multi-programming](https://www.geeksforgeeks.org/multiprogramming-in-operating-system/), i.e., the number of processes present in a ready state or in main memory 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.
* In some systems, the long-term scheduler might not even exist. For example, in time-sharing systems like Microsoft Windows, there is usually no long-term scheduler. Instead, every new process is directly added to memory for the short-term scheduler to handle.
* Slowest among the three (that is why called long term).

**2. Short-Term or CPU Scheduler**

CPU Scheduler is responsible for selecting one process from the ready state for running (or assigning CPU to it).

* STS (Short Term Scheduler) must select a new process for the CPU frequently to avoid starvation.
* The CPU scheduler uses different [scheduling algorithms](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/) to balance the allocation of CPU time.
* It picks a process from ready queue.
* Its main objective is to make the best use of CPU.
* It mainly calls [dispatcher](https://www.geeksforgeeks.org/difference-between-dispatcher-and-scheduler/).
* Fastest among the three (that is why called Short Term).

The [dispatcher](https://www.geeksforgeeks.org/difference-between-dispatcher-and-scheduler/)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 work:

* Saving context (process control block) of previously running process if not finished.
* Switching system mode to user mode.
* Jumping to the proper location in the newly loaded program.

**3. Medium-Term Scheduler**

Medium Term Scheduler (MTS) is responsible for moving a process from memory to disk (or swapping).

* It reduces the degree of multiprogramming (Number of processes present in main memory).
* A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix (of CPU bound and IO bound)
* When needed, it brings process back into memory and pick up right where it left off.
* It is faster than long term and slower than short term.

**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. |
| The slowest 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. |
| 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. |

**Scheduling Algorithms In OS**

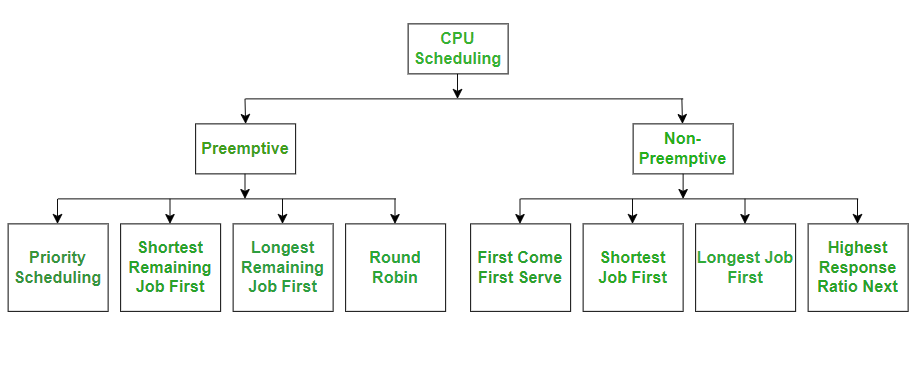
Different **CPU Scheduling algorithms**have different structures and the choice of a particular algorithm depends on a variety of factors.

* **CPU Utilization:**The main purpose of any CPU algorithm is to keep the CPU as busy as possible. Theoretically, CPU usage can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the system load.
* **Throughput:**The average CPU performance is the number of processes performed and completed during each unit. This is called throughput. The output may vary depending on the length or duration of the processes.
* **Turn Round Time:**For a particular process, the important conditions are how long it takes to perform that process. The time elapsed from the time of process delivery to the time of completion is known as the conversion time. Conversion time is the amount of time spent waiting for memory access, waiting in line, using CPU, and waiting for I / O.
* **Waiting Time:**The Scheduling algorithm does not affect the time required to complete the process once it has started performing. It only affects the waiting time of the process i.e. the time spent in the waiting process in the ready queue.
* **Response Time:**In a collaborative system, turn around time is not the best option. The process may produce something early and continue to computing the new results while the previous results are released to the user. Therefore another method is the time taken in the submission of the application process until the first response is issued. This measure is called response time.

**Different Types of CPU Scheduling Algorithms**

There are mainly two types of scheduling methods:

* **Preemptive Scheduling:**Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
* **Non-Preemptive Scheduling:**Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.



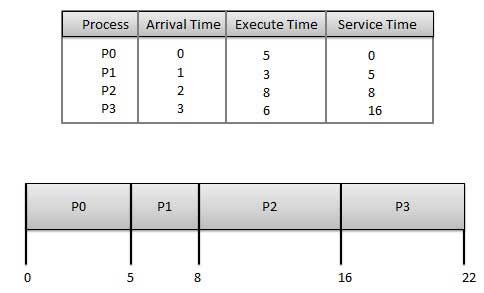
A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

* First-Come, First-Served (FCFS) Scheduling
* Shortest-Job-Next (SJN) Scheduling
* Priority Scheduling
* Shortest Remaining Time
* Round Robin(RR) Scheduling
* Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

* Jobs are executed on first come, first serve basis.
* It is a non-preemptive, pre-emptive scheduling algorithm.
* Easy to understand and implement.
* Its implementation is based on FIFO queue.
* Poor in performance as average wait time is high.



**Wait time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time - Arrival Time** |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 8 - 2 = 6 |
| P3 | 16 - 3 = 13 |

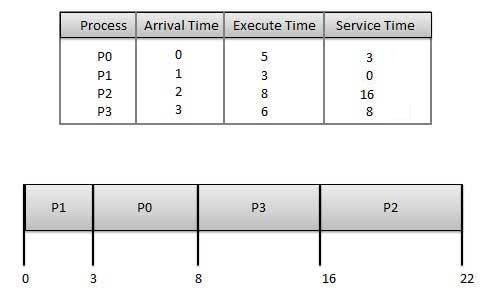
Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

* This is also known as **shortest job first**, or SJF
* This is a non-preemptive, pre-emptive scheduling algorithm.
* Best approach to minimize waiting time.
* Easy to implement in Batch systems where required CPU time is known in advance.
* Impossible to implement in interactive systems where required CPU time is not known.
* The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Execution Time** | **Service Time** |
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P3 | 3 | 6 | 8 |



**Waiting time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Waiting Time** |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 14 - 2 = 12 |
| P3 | 8 - 3 = 5 |

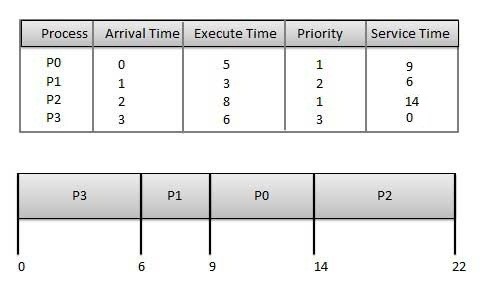
Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Execution Time** | **Priority** | **Service Time** |
| P0 | 0 | 5 | 1 | 0 |
| P1 | 1 | 3 | 2 | 11 |
| P2 | 2 | 8 | 1 | 14 |
| P3 | 3 | 6 | 3 | 5 |



**Waiting time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Waiting Time** |
| P0 | 0 - 0 = 0 |
| P1 | 11 - 1 = 10 |
| P2 | 14 - 2 = 12 |
| P3 | 5 - 3 = 2 |

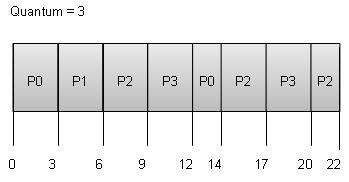
Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Shortest Remaining Time

* Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
* The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
* Impossible to implement in interactive systems where required CPU time is not known.
* It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

* Round Robin is the preemptive process scheduling algorithm.
* Each process is provided a fix time to execute, it is called a **quantum**.
* Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
* Context switching is used to save states of preempted processes.



**Wait time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time - Arrival Time** |
| P0 | (0 - 0) + (12 - 3) = 9 |
| P1 | (3 - 1) = 2 |
| P2 | (6 - 2) + (14 - 9) + (20 - 17) = 12 |
| P3 | (9 - 3) + (17 - 12) = 11 |

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

* Multiple queues are maintained for processes with common characteristics.
* Each queue can have its own scheduling algorithms.
* Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.