

Concurrency Management in OS

Concurrency in Operating System

In this article, you will learn the concurrency in the operating system with its principles, issues, advantages and disadvantages.

What is Concurrency?

It refers to the execution of multiple instruction sequences at the same time. It occurs in an operating system when multiple process threads are executing concurrently. These threads can interact with one another via shared memory or message passing. **Concurrency results in resource sharing, which causes issues like deadlocks and resource scarcity.** It aids with techniques such as process coordination, memory allocation, and execution schedule to maximize throughput.

Principles of Concurrency

Today's technology, like multi-core processors and parallel processing, allows multiple processes and threads to be executed simultaneously. Multiple processes and threads can access the same memory space, the same declared variable in code, or even read or write to the same file.

The amount of time it takes a process to execute cannot be simply estimated, and you cannot predict which process will complete first, enabling you to build techniques to deal with the problems that concurrency creates.

Interleaved and overlapping processes are two types of concurrent processes with the same problems. **It is impossible to predict the relative speed of execution, and the following factors determine it:**

1. The way operating system handles interrupts

2. Other processes' activities
3. The operating system's scheduling policies

Problems in Concurrency: Some of them are as follows:

1. Locating the programming errors

It's difficult to spot a programming error because reports are usually repeatable due to the varying states of shared components each time the code is executed.

2. Sharing Global Resources

Sharing global resources is difficult. If two processes utilize a global variable and both alter the variable's value, the order in which the many changes are executed is critical.

3. Locking the channel

It could be inefficient for the OS to lock the resource and prevent other processes from using it.

4. Optimal Allocation of Resources

It is challenging for the OS to handle resource allocation properly.

Advantages of Concurrency in OS

1. Better Performance

It improves the operating system's performance. When one application only utilizes the processor, and another only uses the disk drive, the time it takes to perform both apps simultaneously is less than the time it takes to run them sequentially.

2. Better Resource Utilization

It enables resources that are not being used by one application to be used by another.

3. Running Multiple Applications

It enables you to execute multiple applications simultaneously.

Disadvantages of Concurrency

1. It is necessary to protect multiple applications from each other.
2. It is necessary to use extra techniques to coordinate several applications.
3. Additional performance overheads and complexities in OS are needed for switching between applications.

Issues of Concurrency: Various issues of concurrency are as follows:

1. issues of Non-atomic operations

Operations that are non-atomic but interruptible by several processes may happen issues. A non-atomic operation depends on other processes, and an atomic operation runs independently of other processes.

2. Deadlock

In concurrent computing, it occurs when one group member waits for another member, including itself, to send a message and release a lock. Software and hardware locks are commonly used to arbitrate shared resources and implement process synchronization in parallel computing, distributed systems, and multiprocessing.

3. Blocking

A blocked process is waiting for some event, like the availability of a resource or completing an I/O operation. Processes may block waiting for resources, and a process may be blocked for a long time waiting for terminal input. If the process is needed to update some data periodically, it will be very undesirable.

4. Race Conditions

A race problem occurs when the output of a software application is determined by the timing or sequencing of other uncontrollable events. Race situations can also happen in multithreaded software, runs in a distributed environment, or is interdependent on shared resources.

5. Starvation

A problem in concurrent computing is where a process is continuously denied the resources it needs to complete its work. It could be caused by errors in scheduling or mutual exclusion algorithm, but resource leaks may also cause it.

Concurrent system design frequently requires developing dependable strategies for coordinating their execution, data interchange, memory allocation, and execution schedule to decrease response time and maximize throughput

CPU Scheduling in Operating Systems

Scheduling of processes/work is done to finish the work on time. **CPU Scheduling** is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.

Whenever the CPU becomes idle, the operating system must select one of the processes in the line ready for launch. The selection process is done by a temporary (CPU) scheduler. The Scheduler selects between memory processes ready to launch and assigns the CPU to one of them.

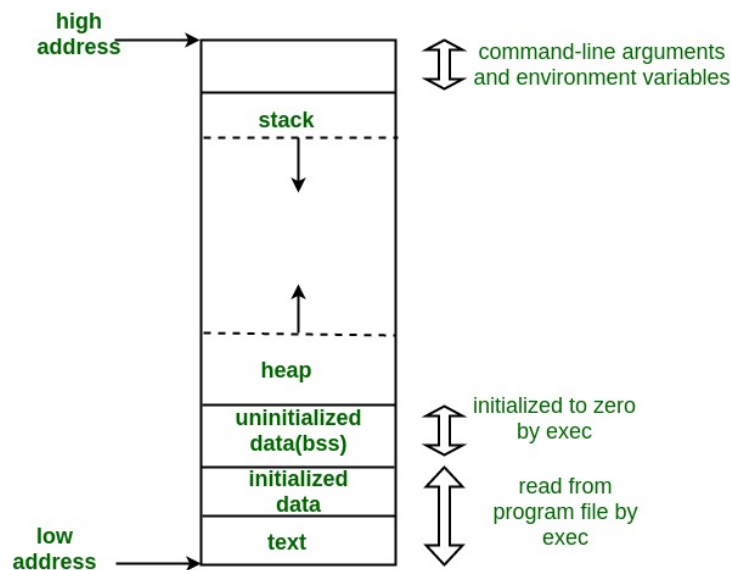
What is a process?

In computing, a process is the instance of a computer program that is being executed by one or many threads. It contains the program code and its activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently.

How is process memory used for efficient operation?

The process memory is divided into four sections for efficient operation:

- The **text category** is composed of integrated program code, which is read from fixed storage when the program is launched.
- The **data class** is made up of global and static variables, distributed and executed before the main action.
- Heap is used for flexible, or dynamic memory allocation and is managed by calls to new, delete, malloc, free, etc.
- The stack is used for local variables. The space in the stack is reserved for local variables when it is announced.



What is Process Scheduling?

Process Scheduling is the process of the process manager handling the removal of an active process from the CPU and selecting another process based on a specific strategy.

Process Scheduling is an integral part of Multi-programming applications. Such operating systems allow more than one process to be loaded into usable memory at a time and the loaded shared CPU process uses repetition time.

There are three types of process schedulers:

- Long term or Job Scheduler
- Short term or CPU Scheduler
- Medium-term Scheduler

Why do we need to schedule processes?

- **Scheduling** is important in many different computer environments. One of the most important areas is scheduling 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 Scheduling** allows the OS to allocate CPU time for each process. Another important reason to use a process scheduling system is that it keeps the CPU busy at all times. This allows you to get less response time for programs.
- Considering that there may be hundreds of programs that need to work, the OS must launch the program, stop it, switch to another program, etc.

The way the OS configures the system to run another in the CPU is called “context switching”. If the OS keeps context-switching programs in and out of the provided CPUs, it can give the user a tricky idea that he or she can run any programs he or she wants to run, all at once.

- So now that we know we can run 1 program at a given CPU, and we know we can change the operating system and remove another one using the context switch, how do we choose which programs we need. run, and with what program?
- That’s where **scheduling** comes in! First, you determine the metrics, saying something like “the amount of time until the end”. We will define this metric as “the time interval between which a function enters the system until it is completed”. Second, you decide on a metrics that reduces metrics. We want our tasks to end as soon as possible.

What is the need for CPU scheduling algorithm?

CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

In Multiprogramming, if the long-term scheduler selects multiple I / O binding processes then most of the time, the CPU remains an idle. The function of an effective program is to improve resource utilization.

If most operating systems change their status from performance to waiting then there may always be a chance of failure in the system. So in order to minimize this excess, the OS needs to schedule tasks in order to make full use of the CPU and avoid the possibility of deadlock.

Objectives of Process Scheduling Algorithm:

- Utilization of CPU at maximum level. **Keep CPU as busy as possible.**
- **Allocation of CPU should be fair.**
- **Throughput should be Maximum.** i.e. Number of processes that complete their execution per time unit should be maximized.
- **Minimum turnaround time**, i.e. time taken by a process to finish execution should be the least.
- There should be a **minimum waiting time** and the process should not starve in the ready queue.
- **Minimum response time.** It means that the time when a process produces the first response should be as less as possible.

Different terminologies to take care of in CPU Scheduling algorithm?

- **Arrival Time:** Time at which the process arrives in the ready queue.
- **Completion Time:** Time at which process completes its execution.
- **Burst Time:** Time required by a process for CPU execution.
- **Turn Around Time:** Time Difference between completion time and arrival time. $\text{Turn Around Time} = \text{Completion Time} - \text{Arrival Time}$
- **Waiting Time (W.T):** Time Difference between turnaround time and burst time. $\text{Waiting Time} = \text{Turn Around Time} - \text{Burst Time}$

CPU Scheduling Criteria

Different **CPU Scheduling algorithms** have different structures and the choice of a particular algorithm depends on a variety of factors. Many criteria have been suggested for comparing CPU scheduling algorithms. The criteria include the following:

1. **CPU utilization:** The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.
2. **Throughput:** A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending on the length or duration of the processes.
3. **Turnaround time:** For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU, and waiting for I/O. The formula to calculate Turn Around Time

$$\text{Turn Around Time} = \text{Completion Time} - \text{Arrival Time.}$$

4. **Waiting time:** A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue. The formula for calculating Waiting Time

$$\text{Waiting Time} = \text{Turnaround Time} - \text{Burst Time.}$$

5. **Response time:** In an interactive system, turn-around time is not the best criterion. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus, another criterion is the time taken from submission of the process of the request until the first response is produced. This measure is called response time. The formula to calculate Response Time

$$\text{Response Time} = \text{CPU Allocation Time (when the CPU was allocated for the first)} - \text{Arrival Time}$$

6. **Completion time:** The completion time is the time when the process stops executing, which means that the process has completed its burst time and is completely executed.
7. **Priority:** If the operating system assigns priorities to processes, the scheduling mechanism should favor the higher-priority processes.
8. **Predictability:** A given process always should run in about the same amount of time under a similar system load.

What are the different types of CPU Scheduling Algorithms?

Preemptive Scheduling

Preemptive scheduling is used when a process switches from the running state to the ready state or from the waiting state to the ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute.

Non-Preemptive Scheduling

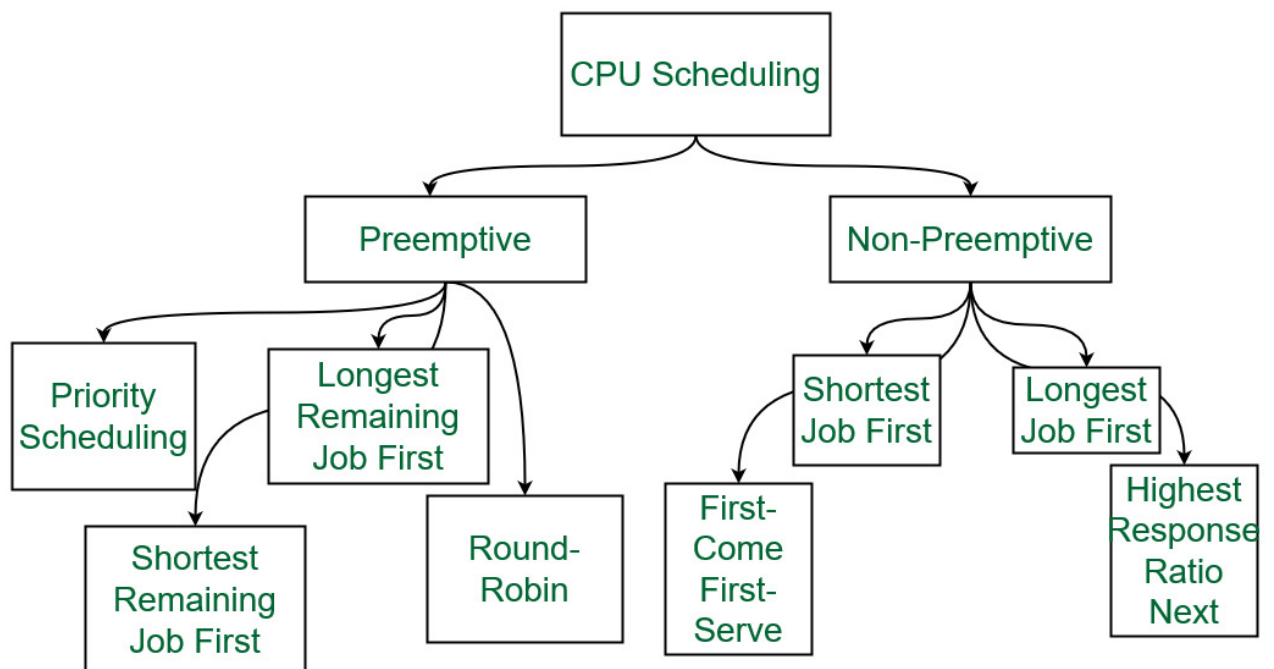
Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process.

Key Differences Between Preemptive and Non-Preemptive Scheduling

1. In preemptive scheduling, the CPU is allocated to the processes for a limited time whereas, in Non-preemptive scheduling, the CPU is allocated to the process till it terminates or switches to the waiting state.
2. The executing process in preemptive scheduling is interrupted in the middle of execution when a higher priority one comes whereas, the executing process in non-preemptive scheduling is not interrupted in the middle of execution and waits till its execution.
3. In Preemptive Scheduling, there is the overhead of switching the process from the ready state to the running state, vice-verse, and maintaining the ready queue. Whereas in the case of non-preemptive scheduling has no overhead of switching the process from running state to ready state.
4. In preemptive scheduling, if a high-prior The process The process non-preemptive low-priority process frequently arrives in the ready queue then the process with low priority has to wait for a long, and it may have to starve. , in non-preemptive scheduling, if CPU is allocated to the process having a larger burst time then the processes with a small burst time may have to starve.
5. Preemptive scheduling attains flexibility by allowing the critical processes to access the CPU as they arrive in the ready queue, no matter what process is executing currently. Non-preemptive scheduling is called rigid as even if a critical process enters the ready queue the process running CPU is not disturbed.
6. Preemptive Scheduling has to maintain the integrity of shared data that's why it is cost associative which is not the case with Non-Preemptive Scheduling.

Parameter	PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
Basic	In this resources(CPU Cycle) are allocated to a process for a limited time.	Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted until it terminates itself or its time is up.
Starvation	If a process having high priority frequently arrives in the ready queue, a low priority process may starve.	If a process with a long burst time is running CPU, then later coming process with less CPU burst time may starve.
Overhead	It has overheads of scheduling the processes.	It does not have overheads.
Flexibility	flexible	rigid
Cost	cost associated	no cost associated
CPU Utilization	In preemptive scheduling, CPU utilization is high.	It is low in non preemptive scheduling.
Waiting Time	Preemptive scheduling waiting time is less.	Non-preemptive scheduling waiting time is high.
Response Time	Preemptive scheduling response time is less.	Non-preemptive scheduling response time is high.

Parameter	PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
Decision making	Decisions are made by the scheduler and are based on priority and time slice allocation	Decisions are made by the process itself and the OS just follows the process's instructions
Process control	The OS has greater control over the scheduling of processes	The OS has less control over the scheduling of processes
Overhead	Higher overhead due to frequent context switching	Lower overhead since context switching is less frequent
Examples	Examples of preemptive scheduling are Round Robin and Shortest Remaining Time First.	Examples of non-preemptive scheduling are First Come First Serve and Shortest Job First.



Different types of CPU Scheduling Algorithms

Let us now learn about these CPU scheduling algorithms in operating systems one by one:

1. First Come First Serve:

FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

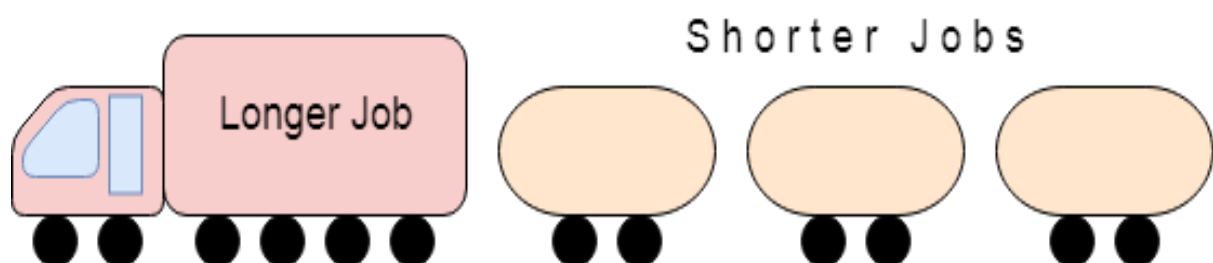
- Easy to implement
- First come, first serve method

Disadvantages of FCFS:

- FCFS suffers from **Convoy effect**.
- The average waiting time is much higher than the other algorithms.
- FCFS is very simple and easy to implement and hence not much efficient.

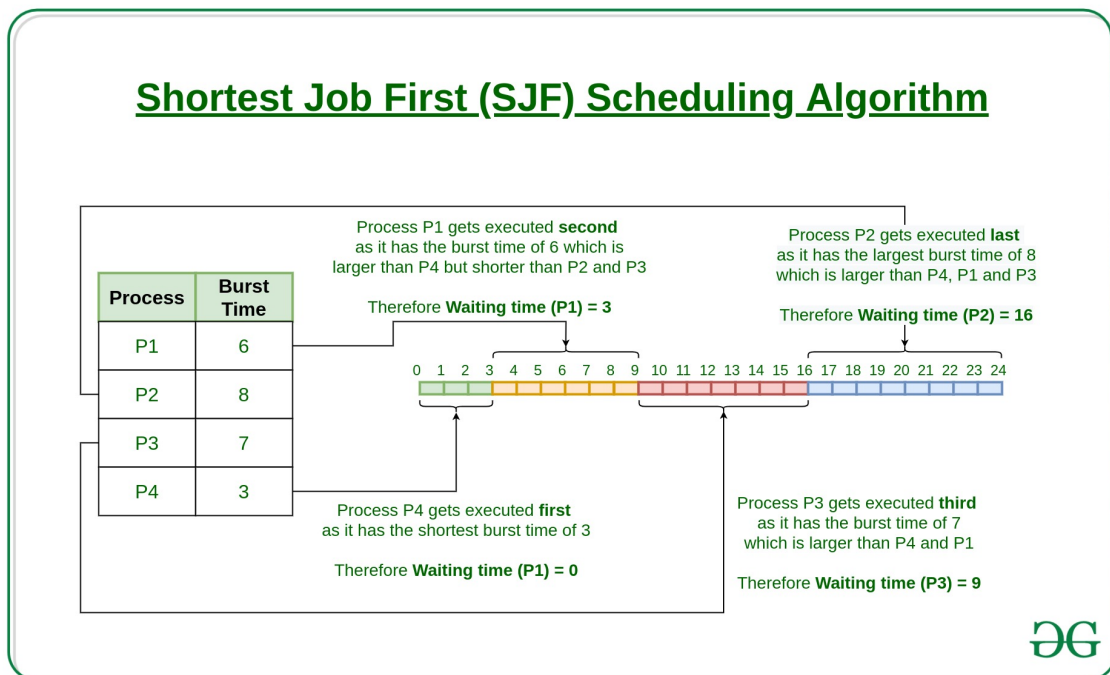
To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on [First come, First serve Scheduling](#).

The Convoy Effect, Visualized Starvation



2. Shortest Job First(SJF):

Shortest job first (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.



Characteristics of SJF:

- Shortest Job first has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
- It is associated with each task as a unit of time to complete.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Advantages of Shortest Job first:

- As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
- SJF is generally used for long term scheduling

Disadvantages of SJF:

- One of the demerit SJF has is starvation.
- Many times, it becomes complicated to predict the length of the upcoming CPU request

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on Shortest Job First.

3. Longest Job First(LJF):

Longest Job First(LJF) scheduling process is just opposite of shortest job first (SJF), as the name suggests this algorithm is based upon the fact that the process with the largest burst time is processed first. Longest Job First is non-pre-emptive in nature.

Characteristics of LJF:

- Among all the processes waiting in a waiting queue, CPU is always assigned to the process having largest burst time.
- If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first.
- LJF CPU Scheduling can be of both preemptive and non-preemptive types.

Advantages of LJF:

- No other task can schedule until the longest job or process executes completely.
- All the jobs or processes finish at the same time approximately.

Disadvantages of LJF:

- Generally, the LJF algorithm gives a very high average waiting time and average turn-around time for a given set of processes.
- This may lead to convoy effect.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the [Longest job first scheduling](#).

4. Priority Scheduling:

Pre-emptive Priority CPU Scheduling Algorithm is a pre-emptive method of CPU scheduling algorithm that works **based on the priority** of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first. In the case of any conflict, that is, where there is more than one processor with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.

Characteristics of Priority Scheduling:

- Schedules tasks based on priority.
- When the higher priority work arrives while a task with less priority is executed, the higher priority work takes the place of the less priority one and
- The latter is suspended until the execution is complete.

- Lower is the number assigned, higher is the priority level of a process.

Advantages of Priority Scheduling:

- The average waiting time is less than FCFS
- Less complex

Disadvantages of Priority Scheduling:

- One of the most common demerits of the Preemptive priority CPU scheduling algorithm is the Starvation Problem. This is the problem in which a process has to wait for a longer amount of time to get scheduled into the CPU. This condition is called the starvation problem.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on Priority Pre-emptive Scheduling algorithm.

5. Round robin:

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Characteristics of Round robin:

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
- One of the most widely used methods in CPU scheduling as a core.
- It is considered preemptive as the processes are given to the CPU for a very limited time.

Advantages of Round robin:

- Round robin seems to be fair as every process gets an equal share of CPU.
- The newly created process is added to the end of the ready queue.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the Round robin Scheduling algorithm.

6. Shortest Remaining Time First:

Shortest remaining time first is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to the job closest to completion. In SRTF the process with the smallest amount of time remaining until completion is selected to execute.

Characteristics of Shortest remaining time first:

- SRTF algorithm makes the processing of the jobs faster than SJF algorithm, given its overhead charges are not counted.
- The context switch is done a lot more times in SRTF than in SJF and consumes the CPU's valuable time for processing. This adds up to its processing time and diminishes its advantage of fast processing.

Advantages of SRTF:

- In SRTF the short processes are handled very fast.
- The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.

Disadvantages of SRTF:

- Like the shortest job first, it also has the potential for process starvation.
- Long processes may be held off indefinitely if short processes are continually added.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the [shortest remaining time first](#).

7. Longest Remaining Time First:

The longest remaining time first is a preemptive version of the longest job first scheduling algorithm. This scheduling algorithm is used by the operating system to program incoming processes for use in a systematic way. This algorithm schedules those processes first which have the longest processing time remaining for completion.

Characteristics of longest remaining time first:

- Among all the processes waiting in a waiting queue, the CPU is always assigned to the process having the largest burst time.
- If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first.
- LRF CPU Scheduling can be of both preemptive and non-preemptive types.

Advantages of LRTF:

- No other process can execute until the longest task executes completely.
- All the jobs or processes finish at the same time approximately.

Disadvantages of LRTF:

- This algorithm gives a very high average waiting time and average turn-around time for a given set of processes.
- This may lead to a convoy effect.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the [longest remaining time first](#).

8. Highest Response Ratio Next:

Highest Response Ratio Next is a non-preemptive CPU Scheduling algorithm and it is considered as one of the most optimal scheduling algorithms. The name itself states that we need to find the response ratio of all available processes and select the one with the highest Response Ratio. A process once selected will run till completion.

Characteristics of Highest Response Ratio Next:

- The **criteria** for HRRN is **Response Ratio**, and the **mode** is **Non-Preemptive**.
- HRRN is considered as the modification of Shortest Job First to reduce the problem of starvation.
- In comparison with SJF, during the HRRN scheduling algorithm, the CPU is allotted to the next process which has the **highest response ratio** and not to the process having less burst time.

$$\text{Response Ratio} = (W + S)/S$$

Here, **W** is the waiting time of the process so far and **S** is the Burst time of the process.

Advantages of HRRN:

- HRRN Scheduling algorithm generally gives better performance than the shortest job first Scheduling.
- There is a reduction in waiting time for longer jobs and also it encourages shorter jobs.
-

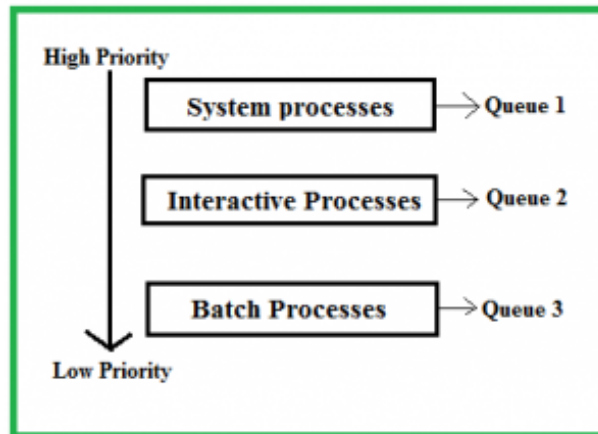
Disadvantages of HRRN:

- The implementation of HRRN scheduling is not possible as it is not possible to know the burst time of every job in advance.
- In this scheduling, there may occur an overload on the CPU.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on Highest Response Ratio Next.

9. Multiple Queue Scheduling:

Processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and a **background (batch)** process. These two classes have different scheduling needs. For this kind of situation **Multilevel Queue Scheduling** is used.



The description of the processes in the above diagram is as follows:

- **System Processes:** The CPU itself has its process to run, generally termed as System Process.
- **Interactive Processes:** An Interactive Process is a type of process in which there should be the same type of interaction.
- **Batch Processes:** Batch processing is generally a technique in the Operating system that collects the programs and data together in the form of a **batch** before the **processing** starts.

Advantages of multilevel queue scheduling:

- The main merit of the multilevel queue is that it has a low scheduling overhead.

Disadvantages of multilevel queue scheduling:

- Starvation problem
- It is inflexible in nature

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on [Multilevel Queue Scheduling](#).

10. Multilevel Feedback Queue Scheduling:

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like Multilevel **Queue Scheduling** but in this process, can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system, and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand, disadvantage of being inflexible.

Advantages of Multilevel feedback queue scheduling:

- It is more flexible
- It allows different processes to move between different queues

Disadvantages of Multilevel feedback queue scheduling:

- It also produces CPU overheads
- It is the most complex algorithm.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on [Multilevel Feedback Queue Scheduling](#).