Scheduling Algorithms

1. First Come First Serve(FCFS): As the name implies that the jobs are executed on a first come first serve basis. It is a simple algorithm based on FIFO that's first in first out. The process that comes first in the ready queue can access the CPU first. If the arrival time is less, then the process will get the CPU soon. It can suffer from the convoy effect if the burst time of the first process is the longest among all the jobs.

2. Shortest Job First (SJF): Also known as the shortest job first or shortest job next is a non-preemptive type algorithm that is easy to implement in batch systems and is best in minimizing the waiting time. It follows the strategies where the process that is having the lowest execution time is chosen for the next execution.

3. Longest Job First Scheduling(LJF): The longest job first (LJF) is the non-preemptive version. This algorithm is also based upon the burst time of the processes. The processes are put into the ready queue according to their burst times and the process with the largest burst time is processed first.

4. Longest Remaining Time First Scheduling (LRTF): The preemptive version of LJF is LRTF. Here the process with the maximum remaining CPU time will be considered first and then processed. And after some time interval, it will check if another process having more Burst Time arrived up to that time or not. If any other process has more remaining burst time, so the running process will get pre-empted by that process.

5. Shortest Remaining Time First(SRTF): This algorithm is based on SJF and this is the preemptive version of SJF. In this scheduling algorithm, the process with the smallest remaining burst time is executed first and it may be preempted with a new job that has arrived with a shorter execution time.

6. Round Robin(RR): It is a preemptive scheduling algorithm in which each process is given a fixed time called quantum to execute. At this time one process is allowed to execute for a quantum and then preempts and then another process is executed. In this way, there is context switching between processes to save states of these preempted processes.

7. Priority Scheduling: It is a non-preemptive algorithm that works in batch systems and, each process is assigned with a priority and the process with the highest priority is executed first. This can lead to starvation for other processes.

8. Multiple Levels Queues Scheduling: In this scheduling, multiple queues have their scheduling Algorithms and are maintained with the processes that possess the same characteristics. For this, priorities are assigned to each queue for the jobs to be executed.

9. Multilevel-Feedback-Queue Scheduler: It defines several queues and the scheduling algorithms for each queue. This algorithm is used to determine when to upgrade a process when to demote a process, and also determine the queue in which a process will enter and when that process needs service.
Note: The SJF scheduling algorithm is hypothetical and un-implementable, as it is impossible to determine the burst time of any process without running it. SJF is a benchmarking algorithm as it provides minimum waiting time than any other scheduling algorithm

Usage of Scheduling Algorithms in Different Situations

Every scheduling algorithm has a type of a situation where it is the best choice. Let's look at different such situations:

Situation 1:
The incoming processes are short and there is no need for the processes to execute in a specific order.

In this case, FCFS works best when compared to SJF and RR because the processes are short which means that no process will wait for a longer time. When each process is executed one by one, every process will be executed eventually.

Situation 2:
The processes are a mix of long and short processes and the task will only be completed if all the processes are executed successfully in a given time.

Round Robin scheduling works efficiently here because it does not cause starvation and also gives equal time quantum for each process.

Situation 3:
The processes are a mix of user based and kernel based processes.

Priority based scheduling works efficiently in this case because generally kernel based processes have higher priority when compared to user based processes.

For example, the scheduler itself is a kernel based process, it should run first so that it can schedule other processes.