



Scheduling algorithms


Scheduling algorithms are used in operating systems to manage the execution of processes or threads on a computer's CPU. These algorithms determine the order in which processes are executed and how the CPU time is allocated to different tasks. Different scheduling algorithms are designed to achieve specific goals or priorities.


1. First-Come, First-Served (FCFS):

 **Description:** This is a non-preemptive scheduling algorithm where processes are executed in the order they arrive in the ready queue. Once a process starts executing, it continues until it completes or blocks.


 **Use Case:** FCFS is simple and fair but not suitable for time-sensitive tasks. It is appropriate when there is no urgency for short response times, and all processes have similar execution times.


2. Shortest Job Next (SJN) / Shortest Job First (SJF):

 **Description:** SJN is a preemptive scheduling algorithm that selects the process with the shortest remaining burst time to execute next. SJF is non-preemptive and selects the process with the shortest total execution time.


 **Use Case:** SJN/SJF aims to minimize average waiting time and response time. It is suitable when precise knowledge of process burst times is available, but it can lead to starvation for long processes in non-preemptive mode.


3. Round Robin (RR):

 Description: RR is a preemptive scheduling algorithm where each process is assigned a fixed time slice (quantum) to execute. If the process doesn't complete within its time slice, it's moved to the end of the queue.


 Use Case: RR provides fair allocation of CPU time among processes and is suitable for interactive systems where responsiveness is important. However, it can result in high overhead due to context switches.


4. Priority Scheduling:

 Description: Each process is assigned a priority value, and the scheduler selects the process with the highest priority to execute. Preemptive and non-preemptive versions are available.


 Use Case: Priority scheduling is useful when certain processes have higher priority than others. However, it can lead to starvation for low-priority processes if not managed properly.


5. Multilevel Queue Scheduling:

 Description: Processes are divided into multiple priority levels, and each level has its own scheduling algorithm. Processes within the same level are scheduled based on the chosen algorithm.


 Use Case: This approach balances between tasks with different priorities. Interactive tasks can be assigned higher priority for responsiveness, while longer batch processes have lower priority.


6. Multilevel Feedback Queue Scheduling:

 Description: Similar to multilevel queue scheduling, but processes can move between different queues based on their behavior. Feedback is used to adjust process priorities dynamically.


 Use Case: This algorithm provides a good balance between response time for interactive processes and throughput for batch processes.


7. Lottery Scheduling:

 Description: Processes are assigned "lottery tickets" and a lottery is held to determine which process runs next. The more tickets a process has, the higher its chances of being selected.

 Use Case: Lottery scheduling provides a way to allocate CPU time proportionally to each process's tickets, allowing flexible prioritization. It can be used when fairness is important.

8. Earliest Deadline First (EDF):

 Description: A real-time scheduling algorithm where each process has a deadline by which it must complete. The process with the earliest deadline is selected for execution.

 Use Case: EDF is suitable for real-time systems where meeting deadlines is critical. It requires knowledge of process execution times and deadlines.

When to Use Each Algorithm:

- 🔑 Use **FCFS** for simplicity and fairness, when response times are not critical.
 - 🔑 Use **SJN/SJF** when burst times are known and you want to minimize waiting and response times.
 - 🔑 Use **RR** for interactive systems that require responsiveness and fairness.
 - 🔑 Use **priority scheduling** when you need to prioritize certain processes.
 - 🔑 Use **multilevel queue/feedback queue scheduling** for mixed workload environments.
 - 🔑 Use **lottery scheduling** for flexible and proportional allocation.
 - 🔑 Use **EDF** for real-time systems with stringent deadline requirements.
- 🔑 The choice of scheduling algorithm depends on the characteristics of the processes, system requirements, and performance goals. Different algorithms offer different trade-offs in terms of fairness, responsiveness, throughput, and complexity.