CPU Scheduling

CPU Scheduling

- **CPU scheduling** is the task performed by the **CPU** that decides **the order processes should be executed**.
- To **schedule processes** a **queue** is used.

Types of Processes

- There are **two types** of **processes**:
  1. **I/O Bound** processes are processes that have small bursts of CPU activity, and then wait for I/O. These type of processes directly affect the user interaction, so they should have a higher priority.
  2. **CPU Bound** processes are processes that have little to no I/O, they mostly perform computations. These types of processes are able to function with a lower priority.

Scheduling Criteria

- **Maximize CPU utilization** — We want to keep the CPU as busy as possible.
- **Maximize throughput** — We want to maximize the amount of processes that complete their execution per time unit.
- **Minimize turnaround time** — We want to minimize the amount of time it takes to execute a process.
- **Minimize waiting time** — We want to minimize the amount of time a processes has to wait before getting executed.
- **Minimize response time** — We want to minimize the amount of time it takes from when a request was submitted, to the first response.
- **Fairness** — We want to give each process a fair share of the CPU.

First Come First Server (FCFS) Scheduling

- This algorithm works by **fully executing processes** in the order they are **placed into the queue**.
- This algorithm is simple, and fair. However, it wait time depends on the arrival time, and short processes will be stuck waiting for long processes to complete.
- This algorithm could also cause the system to half if a single process enters an infinite loop.

Shortest-Job-First (SJF) Scheduling

- This algorithm works by **associating** each **processes** by the length of it's **next CPU burst**. These lengths are then used to schedule the processes with the shortest time first.
- Implementing this algorithm with **no preemption**, the process continues to execute until it's CPU burst is complete.
- Implementing this algorithm with **preemption**, the process continues to execute but may be paused to switch to a shorter process that enteres the queue.
- This algorithm has a more optimal minimum average time, but is not pratical. It is too difficult to predict burt times, and may lead to starvation of really long jobs.

Round Robin Scheduling

- This algorithm works by assigning **each process** a **small unit of CPU time** (time quantum q). This time is ussually 10-100 milliseconds. **After this time has elapsed**, the **process** is **preempted** and added to the **end of the ready queue**.

Priority Scheduling

- A **priority** is an **integer** associated with **each process**.
- In this algorithm, the **CPU** is allocated to the **process with the highest priority** (smallest integer).
  - There is a **preemtive**, and **non-preemtive** version of the algorithm.
- **SFJ** is **priority scheduling** where priority is the inverse of predicted next CPU burst time.
- The main problem with this algorithm is that **low priority processes** may **never execute**. The solotuon to this is **aging**, as a **processes ages, it's priority is increased**.

Multi-Level Queue Scheduling

- **Multi-Level queue scheduling** works by creating a ready queue, and **partitioning it into seperate queues** (eg foreground, background).
- **Processes** permanently stay in **a single partition**.
- Each **queue** has it's own **scheduling algorithm** (foreground uses RR, background uses FCFS).
- **Scheduling** must be done **between the queues**. To do this, **time slicing** is used. Each queue gets a **certain amount of CPU time**, which it can schedule amongst **it's processees**.

Multi-Level Feedback Queue Scheduling

- The **Multi-Level queue** algorithm does not allow process to **change queues**, which is a problem, as the functionality of **processes can change over time**. (eg from foreground to background).
- The **Multi-Level feedback queue** algorithm allows processes to **change queues** based on the **characteristics of their CPU bursts**.