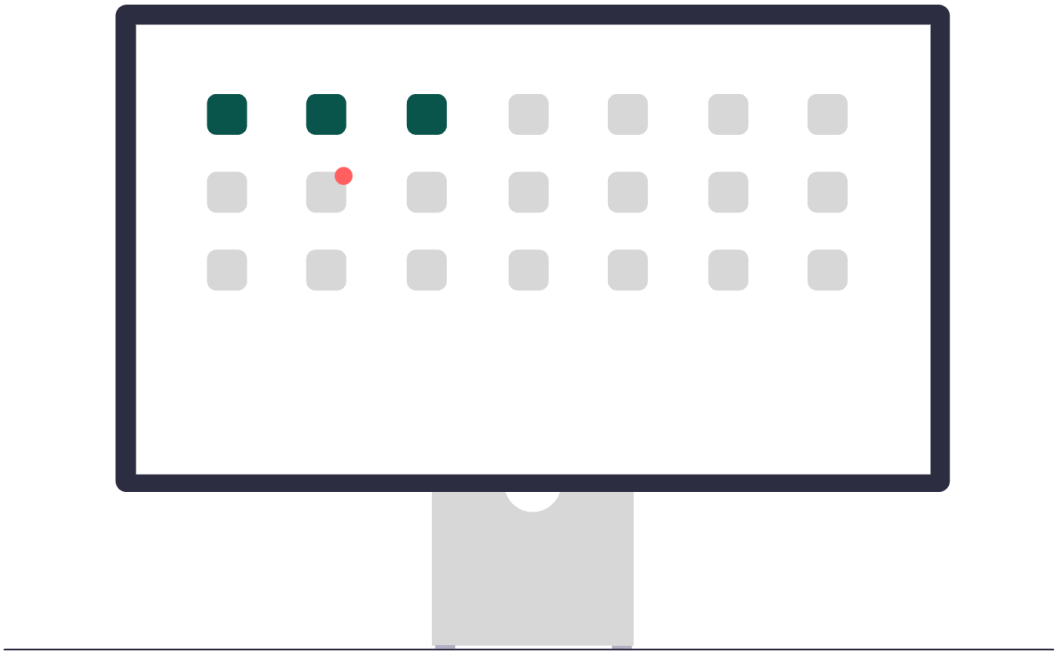
Report of CSE325 Project Report

SJF & FCFS CPU Scheduling



Submitted By:

Aklhak Hossain
2022-3-60-057
https://ahjim.com

Submitted To:

Khairum Islam
Lecturer
Department of CSE
East West University

INTRODUCTION

CPU scheduling plays a vital role in the functioning of an operating system by determining the order in which processes are executed by the CPU. Its primary goal is to make sure that the CPU resources are used efficiently while minimizing both waiting and turnaround times for processes. By optimizing how processes are scheduled, we can ensure that the system runs smoothly and that users experience minimal delays.

Among the numerous scheduling algorithms available, two of the most well-known and widely used are **First-Come-First-Served (FCFS)** and **Shortest Job First (SJF)**. Both are favored for their simplicity and effectiveness, each serving a specific need within different system scenarios.

- **First-Come-First-Served (FCFS)**: This is a non-preemptive scheduling algorithm where tasks are processed in the exact order they arrive. In this approach, the first process to arrive is the first to be executed, followed by the next, and so on. While it's simple to implement, it can lead to inefficient use of CPU resources, particularly when a long task arrives first, causing delays for shorter tasks behind it.

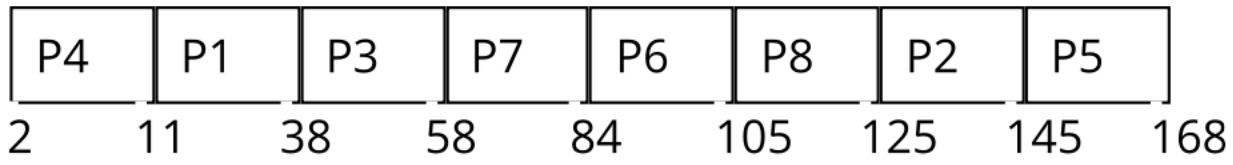
- **Shortest Job First (SJF)**: Another non-preemptive scheduling algorithm, SJF focuses on selecting the process with the shortest CPU burst time from the ready queue. This algorithm minimizes waiting times by giving priority to shorter tasks, ensuring they are completed first. However, it relies on knowing the burst times in advance, and if short processes keep arriving, longer ones can experience delays or "starvation."

For this simulation project, I'll be considering how these two algorithms function, comparing their performance, and analyzing their impact on system efficiency and fairness.

Calculation & ANALYSIS

FCFS(Non-Preemptive):

Gantt Chart:



Calculation Table:

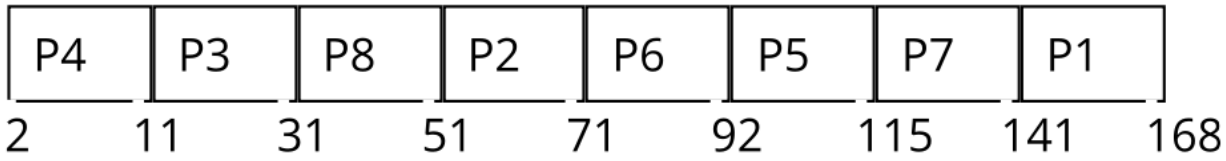
PID	AT	BT	WT	CT	TAT	RT
P1	5	27	6	38	33	6
P2	10	20	115	145	135	115
P3	5	20	33	58	53	33
P4	2	9	0	11	9	0
P5	10	23	135	168	158	135
P6	6	21	78	105	99	78
P7	5	26	53	84	79	53
P8	9	20	96	125	116	96
Avg.			64.50	91.75	85.25	64.50

First-Come-First-Served (FCFS) Explained:

The processes are sorted in the **Ready Queue** by their **Arrival Time** and then each process starts with execution as soon as the CPU becomes free and the process has come First . Completion time, turnaround time, and waiting time are calculated sequentially.
The Gantt chart visually represents the execution order of processes, while metrics like average turnaround time and waiting time quantify the algorithm's efficiency.

SJF(Non-Preemptive):

Gantt Chart:



Calculation Table:

PID	AT	BT	WT	CT	TAT	RT
P1	5	27	136	168	163	136
P2	10	20	41	71	61	41
P3	5	20	6	31	26	6
P4	2	9	0	11	9	0
P5	10	23	82	115	105	82
P6	6	21	65	92	86	65
P7	5	26	110	141	136	110
P8	9	20	22	51	42	22
Avg.			57.75	85.00	78.50	57.75

Shortest Job First (SJF) Explained:

The processes are sorted in the **Ready Queue** by their **Burst Time** and then each process starts with execution as soon as the CPU becomes free and the process's **Burst Time** is smaller than others. Completion time, turnaround time, and waiting time are calculated sequentially.
The Gantt chart visually represents the execution order of processes, while metrics like average turnaround time and waiting time quantify the algorithm's efficiency.

Analysis:

From the calculation table of the two algorithms we see the most efficient algorithm for this particular Job Sequence is the **Shortest Job First Algorithm** because of its overall reduction in the average **Wait Time**, average **Completion Time** and average **Turnaround Time**.

Comparison between the two algorithm:

FCFS:

FCFS (First-Come, First-Served) is easy to implement, but it can lead to long waiting times for processes with shorter burst times, especially when they're scheduled after a process with a much longer burst time.

Best Case: If all processes have similar arrival and burst times, delays are kept to a minimum.

Worst Case: When a process with a very long burst time arrives first, it can cause significant delays for all the processes that follow.

SJF:

SJF (Shortest Job First) is efficient in minimizing average waiting and turnaround times, but it depends on having accurate knowledge of burst times. It can also lead to the problem of starvation if short processes keep arriving, as longer processes may get delayed indefinitely.

Best Case: When burst times are accurately predicted, and there's a balanced mix of short and long processes.

Worst Case: When the queue is dominated by short processes, causing excessive delays for longer ones.

While SJF generally outperforms FCFS in reducing waiting and turnaround times, FCFS is fairer and more predictable, making it a better choice for real-time or interactive systems.

Conclusion:

This project explores how the FCFS (First-Come, First-Served) and SJF (Shortest Job First) scheduling algorithms work, comparing their performance based on key metrics like waiting times and turnaround times. The goal is to understand the strengths and weaknesses of each algorithm and how they impact the efficiency and fairness of task processing.

From the analysis, it's clear that SJF tends to be more efficient overall. By prioritizing shorter tasks, SJF reduces the amount of waiting time, which leads to faster processing and a better average turnaround time for most tasks. However, SJF comes with its own set of challenges. One of the main issues is **starvation**—if short tasks keep arriving, longer tasks may be perpetually delayed, which can create fairness concerns, especially in systems where all tasks are important.

In contrast, FCFS is much simpler and more predictable. It processes tasks strictly in the order they arrive, which means that every task is treated equally without bias. This can make FCFS feel fairer in some contexts, as there's no prioritization based on task length. However, the downside is that it may lead to significant delays, particularly when a long task is scheduled first. If a lengthy task arrives before shorter ones, all the subsequent tasks must wait, increasing their waiting and turnaround times, which makes the overall process less efficient.

In summary, while SJF is generally better for environments where efficiency is crucial and minimizing delays is a top priority, FCFS remains valuable in situations where fairness and predictability are more important. Each algorithm has its ideal use case, and understanding their strengths and weaknesses can help in choosing the best one for a given scenario.