Operating Systems Project Report

Implementation of FCFS Scheduling Algorithm using Java

Sanush Kannamkulangara Sanoj
Student ID: GH1030311

June 30, 2025

Project Links

- **GitHub Repository:** `https://github.com/Sann7x/OS_FCFS_Project`
- **Video Demonstration:** `https://youtu.be/ZkrnkzX6aK0`

1 Introduction

This project shows how the **First-Come, First-Served (FCFS)** CPU scheduling algorithm works using Java, It is a basic but an important concept in Operating systems. The main goal is to show how the processes are scheduled based on when they arrive and how to calculate the key processes like turnaround time, waiting time and so on.

2 System Design

The project is composed of the following components:

- **Process Class:** This holds important details about each process like process ID, how long it needs the CPU, arrival time, memory requirement and how long it took for the CPU to run the processes.
- **Scheduler Class:** This handles the FCFS scheduling by organizing the processes in an order, replicating how time passes as each runs, and calculating important stats like waiting and turnaround times.
- **User Interface:** It is a straightforward command-line interface where you enter process details and then see the results displayed.

The design focuses on making the code easy to read and organize, while closely matching how operating systems actually work.

3 Implementation

The implementation follows structured Java programming principles. Below shows the key functions:

- **Input Collection:** Uses `Scanner` to collect process attributes from the user.
- **Sorting:** The processes are sorted using Java's built-in `Comparator` based on time of arrival
- **Scheduling:** For each process, the current simulated system time is updated and metrics like waiting time and turnaround time are computed.
- **Output:** The results are printed in tabular form along with average performance metrics.

4 Problems faced and Solutions

- **Timing Simulation:** Handling varying arrival times was challenging; and was only resolved through careful time simulation repeatedly.
- **Sorting by Arrival:** Java's `Collections.sort()` with custom comparators made sure the processes were in correct order.
- **Code Clarity:** The code was organized into clear sections to make it easier to read and to add new features later on.

5 Results

For a sample input of 3 processes, the following output was generated:

PID	Arrival	Burst	Memory	Waiting	Turnaround
2	1	2	3	0	2
1	3	1	5	0	1
3	4	4	2	0	4

Average Waiting Time: 0.00
Average Turnaround Time: 2.33

This confirmed that the FCFS algorithm was implemented correctly.

6 Conclusion and Future Work

I successfully simulated the FCFS scheduling algorithm using Java. This project helped me better understand key operating system ideas like managing processes, keeping track of time, and measuring how well the system performs.

Future Enhancements

- Implement additional scheduling strategies like SJF and Round Robin.
- Integrate memory management techniques such as First-Fit or Best-Fit.
- creating a GUI-based visualizer for better user interaction.