CPU SCHEDULING ALGORITHMS SIMULATOR

OS LAB

PROJECT REPORT

Submitted To

Engr.Ibrahim Shahid

MEMBERS:- RIZWAN Aqib Raza Rehan

DECLARATION

I hereby declare that this Task work has been prepared us during the year 2026 under the guidance of **Engr. Ibrahim Shahid**, Department of Computer Engineering, Bahauddin Zakariya University, Multan, Pakistan, in the partial fulfillment of BSc. Computer Engineering degree prescribed by the university.

I also declare that this Task is the outcome of my own effort, that it has not been submitted to any other university for the award of any degree.

SR. NO.	ROLL NUMBER	NAME OF THE STUDENT
01	2023-CPE-16	Aqib Hussain
02	2023-CPE-06	Muhammad Rizwan
03	2023-CPE-02	Muhammad Raza
04	2023-CPE-19	Muhammad Rehan

CERTIFICATE

It is to certify that Group Leader, **Aqib Hussain (**2023-CPE-**16)** along with his group members **Muhammad Rizwan (**2023-CPE-**06)**, **Muhammad Raza (**2023-CPE-**02)** and **Muhammad Rehan(**2023-CPE-**19)** have successfully completed the Project Based Learning \ open-ended laboratory task of Course title: **Operating Systems** at Department of Computer Engineering under my supervision and guidance in the fulfilment of requirements of Fifth semester, Bachelor of Science in Computer Engineering of Bahauddin Zakariya University, Multan.

Engr. Ibrahim Shahid
(Laboratory Engineer)
Dept. of Computer Engineering,
BZU, Multan

Dr. Shahid Iqbal
(Course Instructor)
Dept. of Computer Engineering,
BZU, Multan

Dr. Imran Malik.
Chairman,
Dept. of Computer Engineering,
BZU, Multan

ACKNOWLEDGEMENT

We wish to express our deepest gratitude to Laboratory Engineer **Engr. Ibrahim Shahid** who guided us on every step of making this Task. It is whole-heartedly appreciated that your great advice for our Task proved monumental towards the success and cooperation of this Task.

We would like to express our gratitude towards our parents & friends for their kind cooperation and encouragement which help us in completion of this Task.

Our thanks and appreciations also go to our colleagues in developing the Task and people who have willingly helped us out with their abilities.

ABSTRACT

This project implements a Process Scheduler Simulator that demonstrates how an operating system schedules processes using different CPU scheduling algorithms. The simulator supports First Come First Serve (FCFS), Shortest Job First (SJF), and Round Robin (RR) scheduling. It calculates key performance metrics such as Waiting Time and Turnaround Time. The project is implemented in C language with Bash scripts to automate multiple test cases using file-based input and output.

Table of Contents

List of Figures

CHAPTER-01: INTRODUCTION

3.1 Background

In a multiprogramming operating system, the CPU must be shared among multiple processes. CPU scheduling decides which process gets the CPU and for how long. Efficient scheduling improves CPU utilization, throughput, and system responsiveness.

3.2 Problem Statement

Manual understanding of CPU scheduling algorithms is difficult without practical visualization. There is a need for a simulator that can execute different scheduling algorithms, compare their performance, and show measurable results.

3.3 Objectives

To simulate FCFS, SJF, and Round Robin scheduling algorithms

- To calculate waiting time and turnaround time for each process
- To compare scheduling algorithms using performance metrics
- To automate testing using Bash scripts

3.4 Scope

- The project focuses on CPU scheduling only.
- It does not include real process execution, memory management, or I/O scheduling.

CHAPTER-02: SYSTEM DESIGN

4.1 Overall Architecture

The simulator reads process data from input files, applies a selected scheduling algorithm, and writes results to output files.

Main Components:

- Input File Handler
- Scheduling Algorithms
- Performance Metric Calculator
- Output Generator

4.2 Process Control Block (PCB)

Each process is represented using a structure containing:

- Process ID (PID)
- Arrival Time
- Burst Time
- Remaining Time (for Round Robin)
- Waiting Time
- Turnaround Time

4.3 Scheduling Algorithms

FCFS (First Come First Serve)

- Processes are executed in order of arrival
- Simple but may cause long waiting times

SJF (Shortest Job First)

- Process with shortest burst time executes first
- Minimizes average waiting time

Round Robin

- Each process gets CPU for a fixed time quantum
- Improves fairness and responsiveness

4.4 Context Switching

Context switching is simulated by moving CPU control from one process to another according to the scheduling algorithm.

CHAPTER-03: IMPLEMENTATION

Programming Language: C

- Used to implement the process scheduling algorithms (FCFS, SJF, Round Robin).
- Chosen for its speed and fine control over memory and arrays.

Scripting Language: Bash

- Used to automate running multiple test cases for all scheduling algorithms.
- Bash script compiles the C code and executes it for all input files.

Development Environment:

- Linux (or any Linux-based system).
- GCC compiler for compiling C code.
- Text editor for writing code (nano, vim, or VS Code).

Supporting Tools:

- Folder structure to manage input and output files systematically.
- Chmod command to make Bash scripts executable**.**

3.2 File Structure

The project uses the following folder structure:

```
Process_scheduler/
├── scedular.c       ← C program implementing all scheduling algorithms
├── run_tests.sh     ← Bash script for automation
│
├── testcases/       ← Folder containing input text files
│   ├── input1.txt
│   └── input2.txt
│
└── results/         ← Folder where output files are       generated
```

Explanation:

- **Scedular.c :**contains all the code for FCFS, SJF, and Round Robin algorithms.
- **Run_tests.sh** :automates compilation and execution for multiple input files.
- **Testcases/** :contains input files for testing.
- **Results/** :stores the output of each algorithm for each test case.

3.3 Input Format

The input file contains process details:

PID	Arrival time	Burst time
1	0	5
2	1	3
3	2	8
4	3	6

3.4 Output Format

The program generates output files in the results/ folder.
Each output file contains:

PID AT BT CT TAT WT

Where:

- PID → Process ID
- AT → Arrival Time
- BT → Burst Time
- CT → Completion Time
- TAT → Turnaround Time (CT - AT)
- WT → Waiting Time (TAT - BT)

3.5 Bash Automation

Bash Script (run_tests.sh) automates compilation and execution:

```
Bash
#!/bin/bash

# Compile the C program
Gcc scedular.c -o scheduler

# Loop through all input files
For file in testcases/*.txt
Do
Name=$(basename $file .txt)
./scheduler fcfs $file results/${name}_fcfs.txt
./scheduler sjf  $file results/${name}_sjf.txt
./scheduler rr   $file results/${name}_rr.txt 2
Done

Echo “All tests completed. Check the results folder.”
```

Explanation:

- Gcc scedular.c -o scheduler → Compiles the C program into an executable named scheduler.
- For file in testcases/*.txt → Loops through all input files.
- Basename $file .txt → Extracts the input file name without extension.
- ./scheduler fcfs $file results/${name}_fcfs.txt → Runs FCFS algorithm and stores output.
- Similarly runs SJF and Round Robin (rr) with a time quantum of 2.
- Outputs are saved automatically in the results/ folder.

Advantages of Bash Automation:

- Runs all test cases without manual execution.
- Ensures consistent output files for all algorithms.
- Saves time and reduces human error.

CHAPTER-04: TESTING AND VALIDATION

4.1 Sample Test Case:

- We use the input for demonstration (input1.txt)
- 4processes with different arrival and burst times.
- Time quantum for Round Robin is 2.

4.2 FCFS Result:

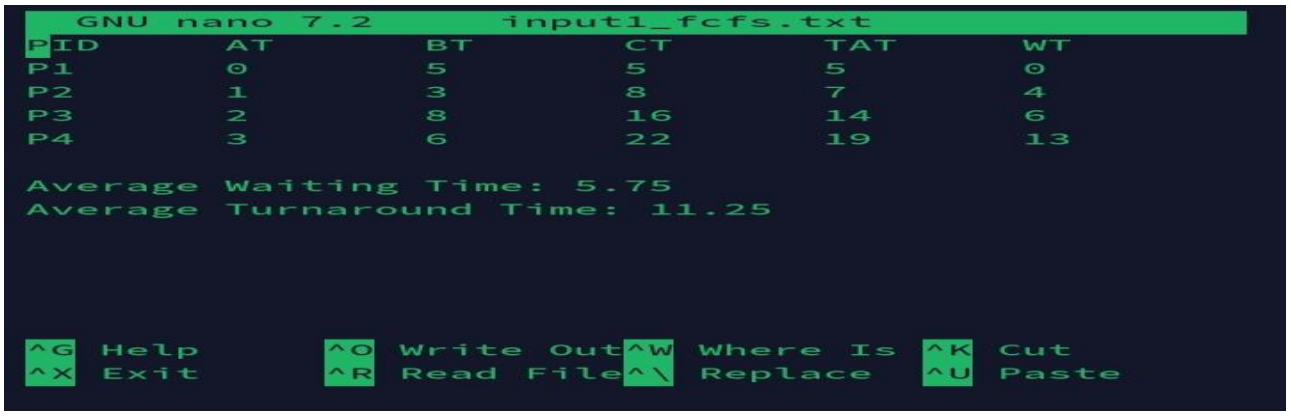

Figure 1 FCFS Result

4.3 SJF (Non-preemptive) Result:

Figure 2 SJF (Non-preemptive) Result

4.4 Round Robin (RR) Result:

Time Quantum = 2

Figure 3 Round Robin (RR) Result

CHAPTER-06: CONCLUSION

This project successfully demonstrates a Process Scheduler Simulator that models core operating system CPU scheduling behavior using FCFS, SJF, and Round Robin algorithms. By simulating process execution with file-based inputs and automated Bash scripts, the system provides a clear, repeatable way to analyze and compare scheduling strategies. The computation of key performance metrics such as Waiting Time and Turnaround Time enables objective evaluation of each algorithm's efficiency and trade-offs. Implemented in C for performance and transparency, the simulator reinforces fundamental OS concepts and serves as an effective educational and experimental tool for understanding process scheduling dynamics.

CHAPTER-07: FUTURE SCOPE

Future enhancements may include adding advanced scheduling algorithms such as Priority Scheduling, Multilevel Queue, and Multilevel Feedback Queue to broaden comparative analysis. The simulator could be extended to support preemptive variants, context-switch overhead, I/O bursts, and dynamic process arrival times for greater realism. A graphical user interface (GUI) or web-based visualization could improve usability and learning outcomes, while statistical reporting and plotting would aid deeper performance analysis. Additionally, modularizing the codebase and integrating configuration-driven test generation would enhance scalability, maintainability, and research applicability.

REFERENCES

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ, USA: Wiley, 2018.
[2] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*, 4th ed. Boston, MA, USA: Pearson, 2015.
[3] W. Stallings, *Operating Systems: Internals and Design Principles*, 9th ed. Boston, MA, USA: Pearson, 2018.
[4] R. Love, *Linux Kernel Development*, 3rd ed. Indianapolis, IN, USA: Addison-Wesley, 2010.