Round Robin (RR) CPU Scheduler Simulator Report

Project Title: Round Robin (RR) CPU Scheduling Simulator
Language Used: Python
Author: IBRAHIM
Date: DEC, 31, 2025

1. Introduction

The Round Robin (RR) scheduling algorithm is widely used in time-sharing operating systems. Each process is assigned a fixed time quantum, and processes are executed in a cyclic order. This project develops a console-based RR CPU Scheduler Simulator in Python. The simulator reads process data from CSV files, calculates key scheduling metrics, generates a Gantt chart, and supports multiple test cases. Users can observe how RR handles process execution, waiting, and turnaround times under different scenarios.

2. Objectives

Implement RR scheduling in Python.

Read process data from CSV files.

Calculate Waiting Time (WT) and Turnaround Time (TAT).

Compute average TAT and average WT.

Generate a text-based Gantt chart to visualize execution order.

Support multiple test cases for analysis of different scenarios.

3**. Key Concepts and Definitions**

Process: A program in execution.

CPU Burst Time: Time required by a process to execute on the CPU.

Arrival Time: Time at which a process enters the ready queue.

Remaining Time: Burst time left for the process to complete.

Waiting Time (WT): Total time a process spends waiting in the ready queue. WT = TAT - Burst Time.

Turnaround Time (TAT): Total time from arrival to completion. TAT = Completion Time - Arrival Time.

Time Quantum: Fixed CPU time assigned to each process per cycle.

Gantt Chart: Text-based visual showing process execution order and time intervals.

CSV File: Comma-separated file containing process information: ProcessID, ArrivalTime, BurstTime.

4. Program Structure

4.1 Dynamic CSV Selection

The simulator allows users to select a CSV file from multiple test files (rr_input_1.csv to rr_input_4.csv). This enables multiple runs without restarting the program.

4.2 **Python Code Flow**

User Input: User selects a CSV file (1–4) and enters time quantum.

CSV Reading: Processes are read using csv.DictReader, and arrival/burst times are converted to integers.

Process Execution:

Processes enter a ready queue based on arrival time.

Each process executes for the time quantum or until completion.

Remaining processes are re-queued if unfinished.

Calculation of Metrics:

Turnaround Time (TAT) = Completion Time - Arrival Time

Waiting Time (WT) = TAT - Burst Time

Gantt Chart Generation: Execution order and time intervals are displayed.

Results Display: Detailed table of processes, WT, TAT, and averages are printed.

5. Test Cases

Sample CSV (rr_input_1.csv):

ProcessID,ArrivalTime,BurstTime

P1,0,5

P2,1,3

P3,2,8

P4,3,6

Other CSV files include varied arrival times and burst lengths to simulate idle CPU times, simultaneous arrivals, and mixed scenarios.

6. **Scheduling Results Format**

Example output (simplified):

Process Arrival Burst Waiting Turnaround

P1	0	5	0	5
P2	1	3	4	7
P3	2	8	9	17
P4	3	6	10	16

Average TAT: 11.25

Average WT: 5.75

Gantt Chart:

| P1(0-4) | P2(4-7) | P3(7-11) | P4(11-15) | ...

7. Analysis of Test Cases

CSV File	Scenario Description	Observations
rr_input_1.csv	Processes arrive sequentially	Minimal waiting; CPU almost always busy
rr_input_2.csv	Some processes arrive later	Idle CPU gaps; waiting time increases
rr_input_3.csv	Processes arrive simultaneously	High contention; later processes wait longer
rr_input_4.csv	Mixed arrivals and bursts	Demonstrates realistic variation in WT and TAT

Key Insight: RR ensures fairness by giving each process a time slice, preventing starvation of longer processes, unlike FCFS.

8. **How to Run the Program**

Ensure Python 3.x is installed.

Place all CSV files in csv_test_files/RR_INPUTS/.

Open terminal or VS Code and navigate to the project folder.

Run: python round_robin_scheduler.py

Select a CSV file (1–4) and enter a time quantum.

Results, averages, and Gantt charts will display in the terminal.

You can repeat with another CSV file or exit the program.

9. Conclusion

The RR CPU Scheduler Simulator successfully demonstrates process scheduling with time-sharing. It calculates waiting and turnaround times, generates Gantt charts, and handles multiple scenarios. The simulator highlights fairness and efficiency in RR scheduling and can serve as a foundation for comparing with other algorithms like FCFS and SJF.