Operating System Lab

Course Code: CSE-3632

Process Scheduler Simulator

Team Members

Jawadul Karim
ID: C221010

Abu Tanvir Hasan Tanmoy
ID: C221001

Md. Shariful Islam Junaed
ID: C221033

Submitted To
Zainal Abedin

Department of Computer Science and Engineering (CSE)
International Islamic University Chittagong, Kumira

1. Introduction

The Process Scheduler Simulator is a Python-based project that replicates CPU scheduling mechanisms in operating systems. It implements four widely-used scheduling algorithms:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Round Robin (RR)
- Priority Scheduling

The project calculates and compares key performance metrics such as **waiting time** and **turnaround time** while also visualizing process execution through Gantt charts. It serves as an educational tool for understanding how operating systems manage multiple processes.

2. Objectives

1. To implement various CPU scheduling algorithms.
2. To calculate and analyze performance metrics like waiting time and turnaround time.
3. To provide an interactive and visual representation of process scheduling for better understanding.

3. Tools and Technologies

- **Programming Language:** Python
- **Libraries Used:** matplotlib (for Gantt chart visualization)
- **Development Environment:** VS Code

4. Scheduling Algorithms Implemented

4.1 First Come First Serve (FCFS)

Description: Processes are executed in the order they arrive.
Advantages: Simple to implement.
Disadvantages: Causes convoy effect (long processes delay shorter ones).

4.2 Shortest Job First (SJF)

Description: Processes with the shortest burst time are executed first.
Advantages: Minimizes average waiting time.
Disadvantages: May lead to starvation of long processes.

4.3 Round Robin (RR)

Description: Each process gets a fixed time slice (quantum) to execute in a cyclic manner.
Advantages: Provides fairness among processes.
Disadvantages: Increased overhead due to frequent context switching.

4.4 Priority Scheduling

Description: Processes are executed based on their priority (lower number = higher priority).
Advantages: Critical processes are executed first.
Disadvantages: May cause starvation of low-priority processes.

5. Implementation Details

Input:

1. Number of processes.
2. For each process:
    - Process ID
    - Arrival time
    - Burst time
    - Priority (if using Priority Scheduling)
3. Time quantum (if using Round Robin).

Output:

- Individual Metrics: Waiting time and turnaround time for each process.
- Overall Metrics:
    - Average waiting time.
    - Average turnaround time.
- Gantt Chart: Visual representation of process execution order.

6. Results and Analysis

Sample Input:

- Number of Processes: 3
- Algorithm: FCFS
- Processes:
  - P1: Arrival = 0, Burst = 5
  - P2: Arrival = 1, Burst = 3
  - P3: Arrival = 2, Burst = 8

Sample Output:

```
Process  Arrival  Burst  Waiting  Turnaround
P1       0        5      0        5
P2       1        3      4        7
P3       2        8      6        14

Average Waiting Time: 3.33
Average Turnaround Time: 8.67
```

Analysis:

- **FCFS:** Executes processes in the order of arrival, causing longer waiting times for later processes.
- **SJF:** Minimizes waiting time but may cause starvation of long jobs.
- **RR:** Provides fairness but increases overhead due to context switching.
- **Priority Scheduling:** Ensures critical tasks are executed promptly but risks starvation of low-priority processes.

7. Gantt Chart Example

For FCFS (P1, P2, P3):

```
| P1 | P2 | P3 |
0    5    8    16
```

8. Conclusion

The Process Scheduler Simulator successfully demonstrates how different CPU scheduling algorithms impact system performance. It highlights the trade-offs between fairness, efficiency, and complexity, providing a deeper understanding of operating system process management.

9. Future Enhancements

1. Add multi-level queue scheduling.
2. Include a graphical user interface (GUI) for easier interaction.
3. Support I/O-bound and CPU-bound process differentiation.

10. References

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts.*
2. Python Official Documentation.
3. Matplotlib Library Documentation.