Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Fall, 2023), B.Sc. in CSE (Day)

CPU Scheduling (A Console-Based Static Simulator) (Project Report)

Course Title: ***Operating System Lab***
Course Code: ***CSE - 310***
Section: ***PC - 213 D4***

Students Details

Name	ID
Md. Shahidul Islam Prodhan	**213902017**

Submission Date: 13th June, 2024

Course Teacher's Name: ***Md. Shoab Alam, Lecturer***

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Project Title

CPU Scheduling: (A Console-Based Static Simulator)

1.2 Overview

The CPU Scheduling (A Console Based Simulator) project aims to simulate various CPU scheduling algorithms, including First-Come-First-Serve (FCFS), Shortest-Job-First (SJF), Priority Scheduling, and Round Robin. The simulator allows users to input process data and visualize the scheduling process through a console-based interface.

1.3 Motivation

The motivation behind developing this project stems from the importance of CPU scheduling in operating systems. Efficient CPU scheduling algorithms play a crucial role in optimizing system performance and resource utilization. By simulating different scheduling algorithms, this project aims to enhance understanding and analysis of their impact on system behavior.

1.4 Problem Definition

1.4.1 Problem Statement / Problem Domain

On the CPU, there are many processes that are continuously processed and executed as long as the computer is on. This project addresses the need for a comprehensive CPU scheduling simulator to facilitate experimentation and analysis of various scheduling algorithms and how the CPU manages the processes between various programs in order to execute and function properly between these processes and executions conveniently.

1.4.2 Complex Engineering Problem

CPU scheduling involves complex engineering challenges, including managing process execution orders, minimizing waiting times, and optimizing resource utilization, making it an essential aspect of operating system design.

1.5 Design Goals / Objectives

My design goals and objectives are very basic and straight forward. That include:

- Develop a console-based simulator for CPU scheduling algorithms.
- Implement key scheduling algorithms such as FCFS, SJF, Priority Scheduling, and Round Robin.
- Provide a user-friendly interface for inputting process data and visualizing scheduling results.
- Facilitate analysis and comparison of different scheduling algorithms.

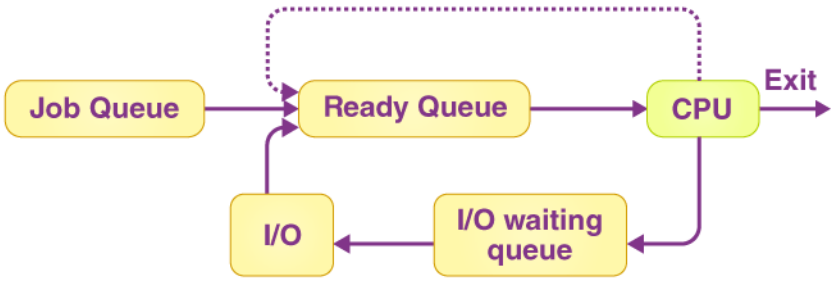


Figure 1.1: CPU Scheduling

1.6 Application

The CPU Scheduling Simulator has applications in:

- **Education:** Enhancing understanding of CPU scheduling algorithms for studying operating systems. Learning how the various processes are scheduled on the CPU.
- **Research:** Facilitating research on system performance and optimization strategies. Look deeply into the working principles of these algorithms and come up with new and more efficient ideas for the CPU.
- **System Design:** Informing decisions related to real-world system design and implementation.

1.7 Methodology

The project follows a structured approach involving:

- Understanding of CPU scheduling algorithms and their implementation.
- Designing a modular and extensible architecture for the simulator.
- Implementation of scheduling algorithms in the C programming language.
- Testing and validation of the simulator using various test cases.

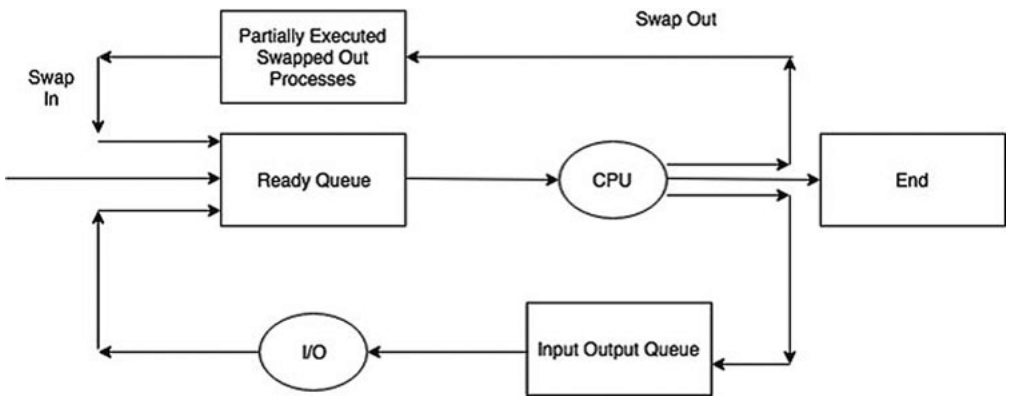


Figure 1.2: Schematic of Scheduling (Credit: ResearchGate)

1.8 Expected Outcome

The expected outcome of the project includes:

- Development of a functional CPU scheduling simulator in the console.
- Demonstration of key scheduling algorithms and their impact on system performance.
- Enhancement of understanding and analysis of CPU scheduling through experimentation.

The outcome of the CPU Scheduling (A console-based simulator) project is the development of a functional and user-friendly simulator for studying CPU scheduling algorithms. Through this project, users will gain a deeper understanding of scheduling strategies such as First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin. The simulator will enable experimentation and analysis of scheduling algorithms, providing insights into their impact on system performance metrics such as waiting time and turnaround time. Ultimately, the project aims to enhance knowledge and comprehension of CPU scheduling principles, facilitating education, research, and decision-making in the domain of operating systems.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This chapter provides an overview of the design and implementation process of the CPU scheduling simulator. **CPU Scheduling** is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution.

2.2 Project Details

CPU Scheduling is a process that allows one process to use the CPU while another process is delayed due to the unavailability of any resources such as I / O etc, thus making full use of the CPU. In short, CPU scheduling decides the order and priority of the processes to run and allocates the CPU time based on various parameters such as CPU usage, throughput, turnaround, waiting time, and response time. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.

2.3 Implementation

The project is implemented in the C programming language, leveraging data structures and algorithms to simulate CPU scheduling behavior. The implementation includes algorithms for sorting, process management, and scheduling logic.

2.3.1 Objectives of Process Scheduling Algorithm

- Utilization of CPU at maximum level. **Keep CPU as busy as possible**.
- **Allocation of CPU should be fair**.

- **Throughput should be maximum**, i.e. Number of processes that complete their execution per time unit should be maximized.
- **Minimum turnaround time**, i.e., the time taken by a process to finish execution should be the least.
- There should be a **minimum waiting time,** and the process should not starve in the ready queue.
- **Minimum response time.** It means that the time when a process produces the first response should be as less as possible.

2.4 Criteria of CPU Scheduling

CPU Scheduling has several criteria. These are as follows-

1. CPU utilization

The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.

2. Throughput

A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending on the length or duration of the processes.

3. Turnaround Time

For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU, and waiting for I/O.

Turn Around Time = Completion Time – Arrival Time.

4. Waiting Time

A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.

Waiting Time = Turnaround Time – Burst Time.

5. Response Time

In an interactive system, turn-around time is not the best criterion. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criterion is the time taken from submission of the process of the request until the first response is produced. This measure is called response time.

Response Time = CPU Allocation Time(when the CPU was allocated for the first) – Arrival Time

6. Completion Time

The completion time is the time when the process stops executing, which means that the process has completed its burst time and is completely executed.

7. Priority

If the operating system assigns priorities to processes, the scheduling mechanism should favor the higher-priority processes.

8. Ready Queue

The Queue where all the processes are stored until the execution of the previous process. This ready queue is very important because there would be confusion in CPU when two same kinds of processes are being executed at the same time.

Then, in these kinds of conditions the ready queue comes into place and then, the its duty is fulfilled.

9. Gantt Chart

It is the place where all the already executed processes are stored. This is very useful for calculating Waiting Time, Completion Time, Turn Around Time.

Process	CPU Burst Time
P_1	30
P_2	6
P_3	8

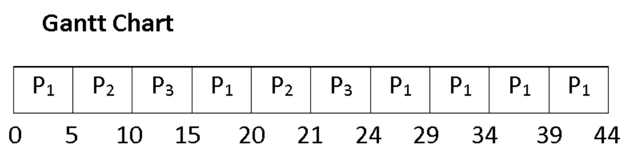


Figure 2.1: Gantt Chart while in Process

9. Process ID

The Process ID is the first Thing is to be written while solving the problem. The Process ID acts like the name of the process. It is usually represented with numbers or P letter with numbers.

2.5 Different Types of CPU Scheduling Algorithms

There are mainly two types of scheduling methods:

- **Preemptive Scheduling:** Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
- **Non-Preemptive Scheduling:** Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.

2.6 Scheduling Algorithms

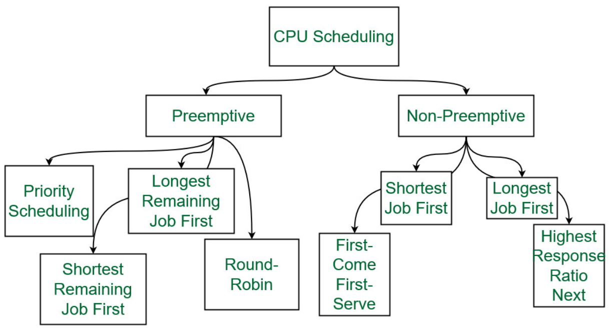


Figure 2.2: Various Scheduling Algorithms

2.6.1 FCFS

It states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

- Easy to implement
- First come, first serve method

2.6.2 SJF

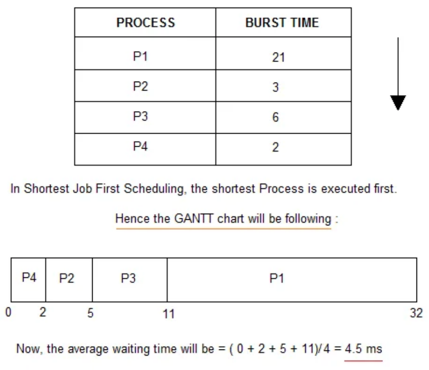
It is a scheduling process that **selects the waiting process with the smallest execution time to execute next**. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed.

Characteristics of SJF:

- Shortest Job First has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
- It is associated with each task as a unit of time to complete.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Advantages of SJF:

- As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
- It is generally used for long-term scheduling



PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

In Shortest Job First Scheduling, the shortest Process is executed first.

Hence the GANTT chart will be following :

Now, the average waiting time will be = (0 + 2 + 5 + 11)/4 = 4.5 ms

Figure 2.3: Working Principles of Shortest Job First

2.6.3 Priority Scheduling

It works **based on the priority** of a process. The most important process must be done first. In the case of any conflict, that is, where there is more than one process with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.

Characteristics of FCFS:

- Schedules tasks based on priority.
- When the higher priority work arrives and a task with less priority is executing, the higher priority proess will takes the place of the less priority proess and
- The later is suspended until the execution is complete.
- Lower is the number assigned, higher is the priority level of a process.

Advantages of FCFS:

- The average waiting time is less than FCFS
- Less complex

2.6.4 Round Robin

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of FCFS CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Characteristics of Round Robin:

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
- One of the most widely used methods in CPU scheduling as a core.
- It is considered preemptive as the processes are given to the CPU for a very limited time.

Advantages of Round Robin:

- Round robin seems to be fair as every process gets an equal share of CPU.
- The newly created process is added to the end of the ready queue.

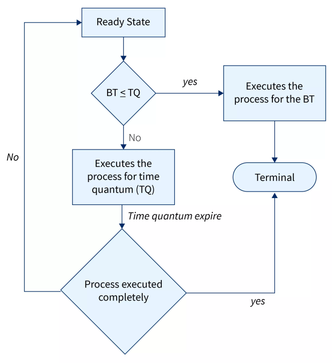

Ready State
BT ≤ TQ
yes
Executes the process for the BT
No
No
Executes the process for time quantum (TQ)
Terminal
Time quantum expire
Process executed completely
yes

Figure 2.4: Working Principles of Round Robin

Chapter 3

Performance Evaluation

3.1 Simulation Environment/Simulation Procedure

The simulator is tested in a controlled environment using various test cases representing different scenarios. It provides options for selecting scheduling algorithms and inputting process data. This project is written in C++, compiled and built in VS Code.

3.1.1 Factors Influencing CPU Scheduling Algorithms:

- The number of processes.
- The processing time required.
- The urgency of tasks.
- The system requirements.

3.2 Codes/Main Interface/Basic Working Process

The project's interface is very straight forward and user-friendly, allowing users to input code snippets for analysis. The basic working process involves the following steps:

- Accepting user input for process details.
- Executing the selected scheduling algorithm.
- Displaying the **scheduling results**, including **Gantt charts** and **performance metrics** along with **comparisons**.

****** N.B.: The codes are attached inside a ZIP file along with test cases expected input/output.***

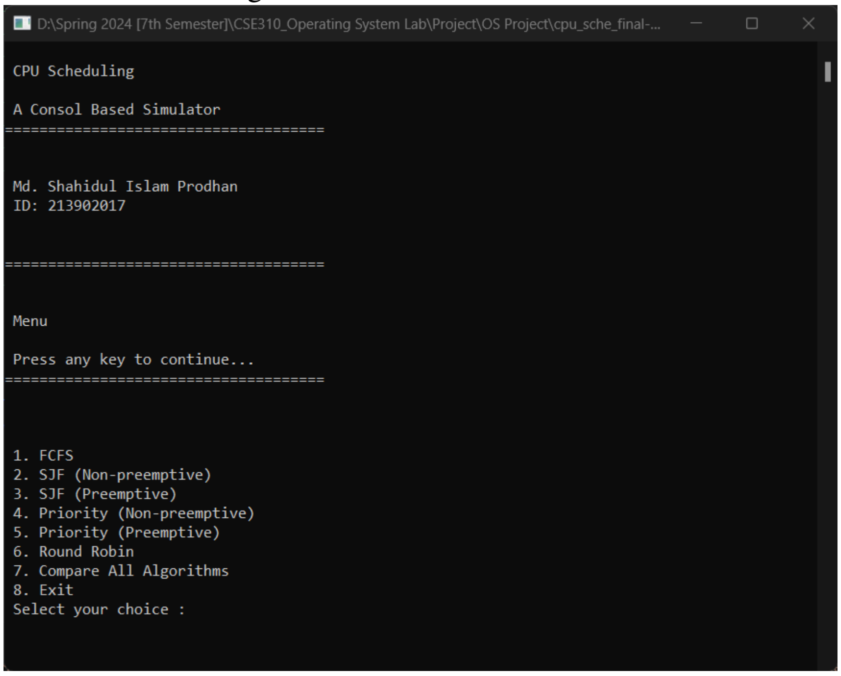
Chapter 4

Result Analysis/Testing

4.1 Analysis, Testing and Test Cases

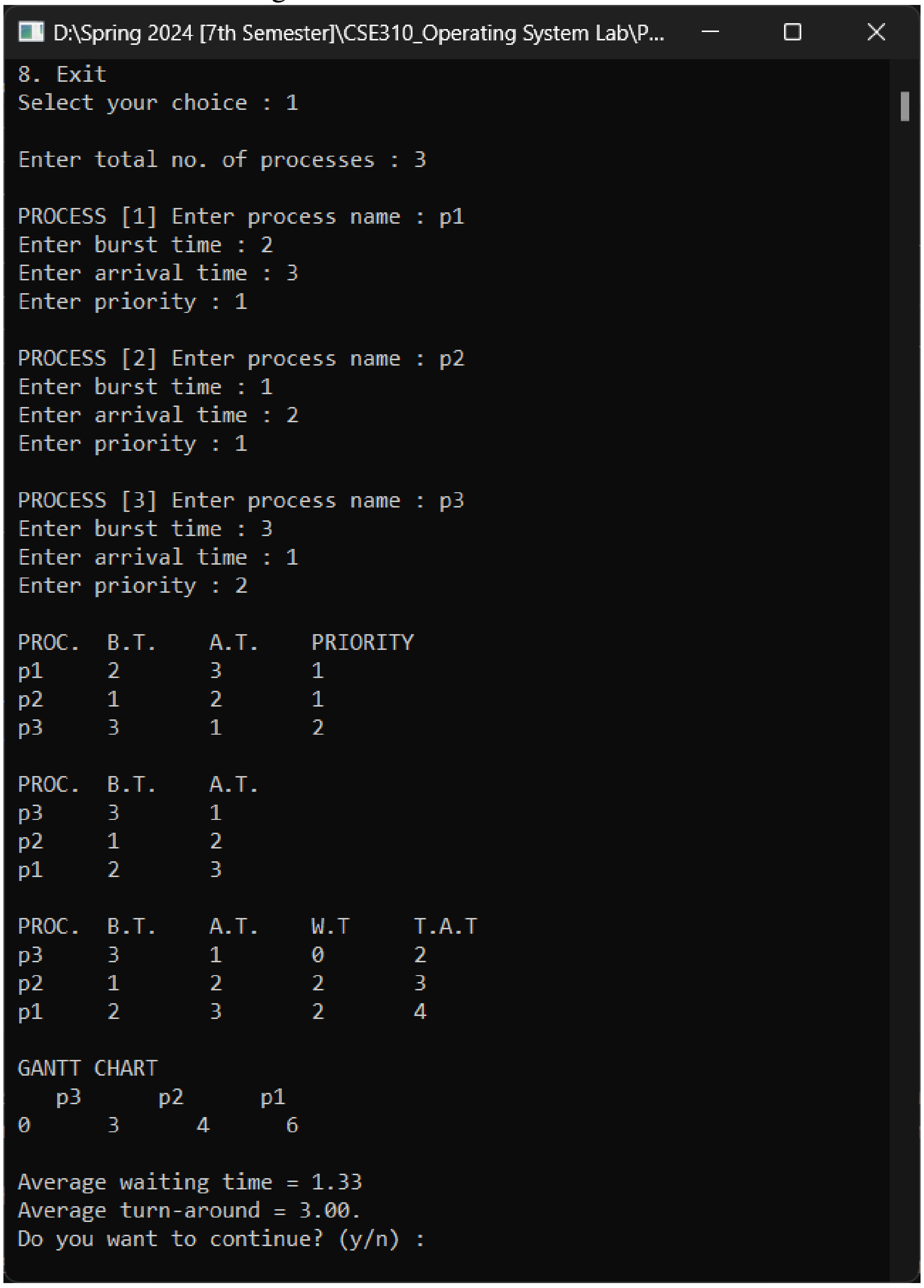
The simulator is tested using various test cases representing different scheduling scenarios. Test cases include: (i) basic scenarios with a few processes and varying burst times, (ii) complex scenarios with multiple processes and different arrival times.

Figure 4.1: Interface and Menu



```
D:\Spring 2024 [7th Semester]\CSE310_Operating System Lab\Project\OS Project\cpu_sche_final-...

CPU Scheduling

A Consol Based Simulator
=====================================


Md. Shahidul Islam Prodhan
ID: 213902017


=====================================


Menu

Press any key to continue...
=====================================



1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
4. Priority (Non-preemptive)
5. Priority (Preemptive)
6. Round Robin
7. Compare All Algorithms
8. Exit
Select your choice :
```

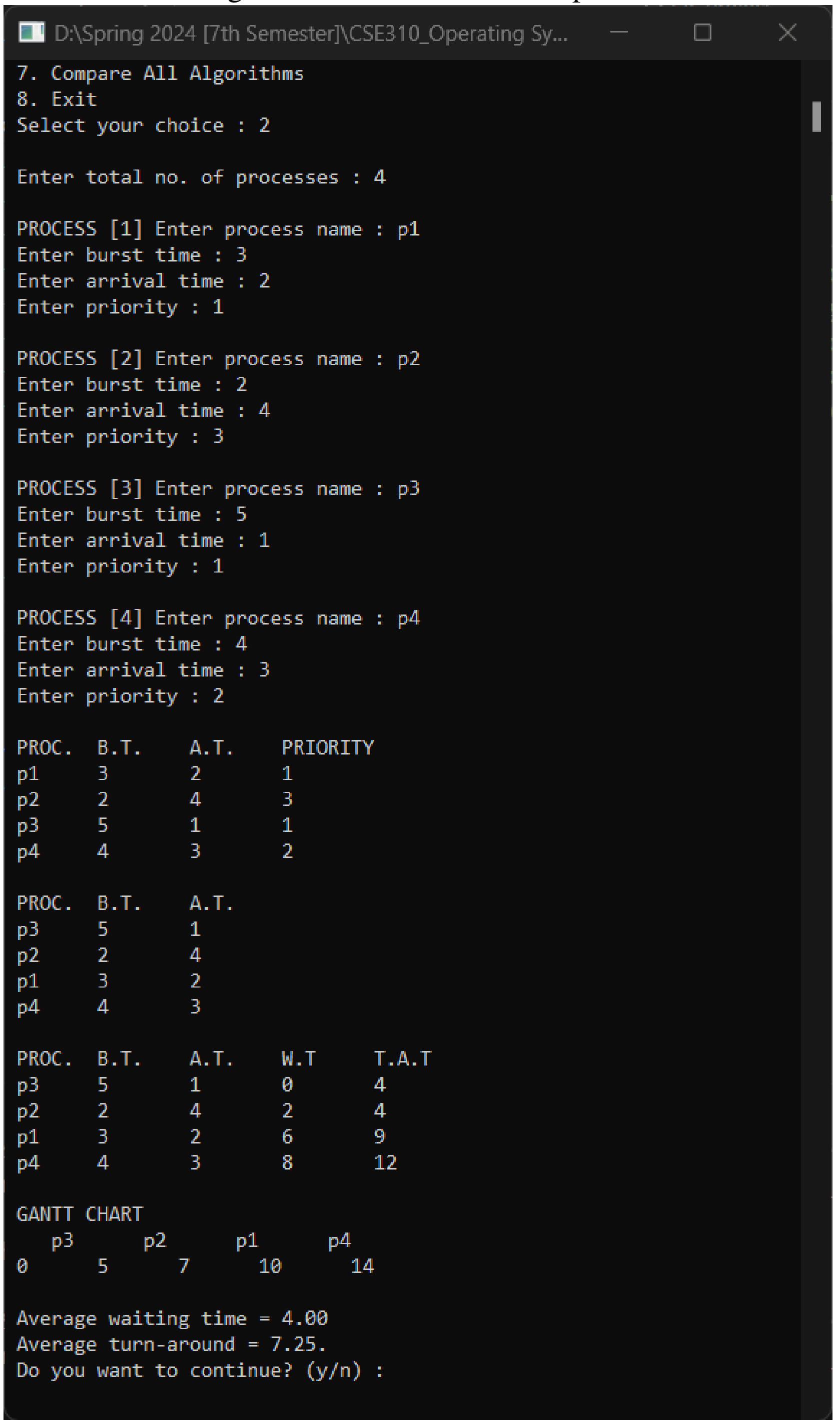
Figure 4.2: FCFS with Gantt chart



D:\Spring 2024 [7th Semester]\CSE310_Operating System Lab\P...

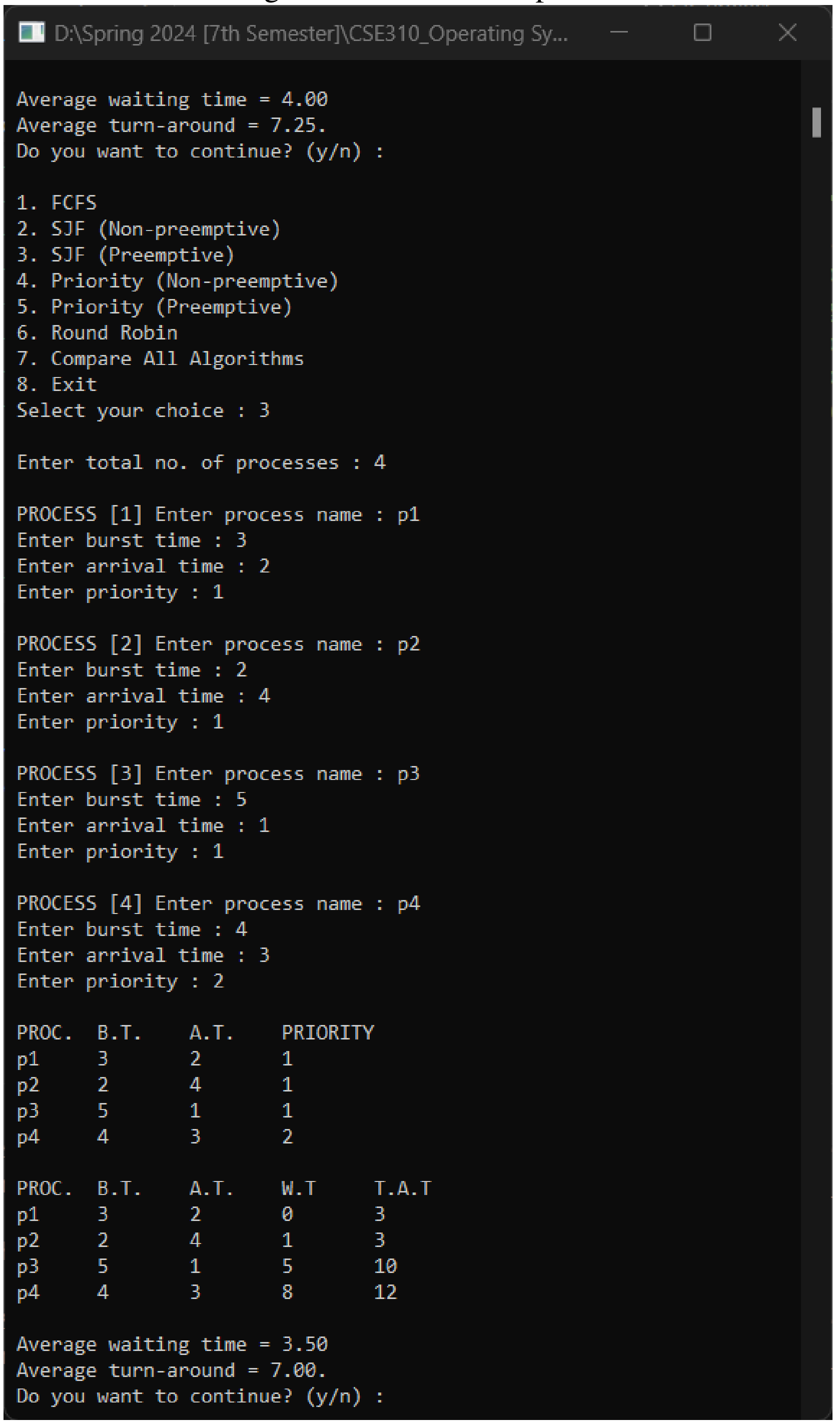
```
8. Exit
Select your choice : 1

Enter total no. of processes : 3

PROCESS [1] Enter process name : p1
Enter burst time : 2
Enter arrival time : 3
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 1
Enter arrival time : 2
Enter priority : 1

PROCESS [3] Enter process name : p3
Enter burst time : 3
Enter arrival time : 1
Enter priority : 2

PROC.  B.T.    A.T.    PRIORITY
p1     2       3       1
p2     1       2       1
p3     3       1       2

PROC.  B.T.    A.T.
p3     3       1
p2     1       2
p1     2       3

PROC.  B.T.    A.T.    W.T     T.A.T
p3     3       1       0       2
p2     1       2       2       3
p1     2       3       2       4

GANTT CHART
   p3      p2      p1
0      3      4      6

Average waiting time = 1.33
Average turn-around = 3.00.
Do you want to continue? (y/n) :
```

Figure 4.3: SJF - Non-Preemptive



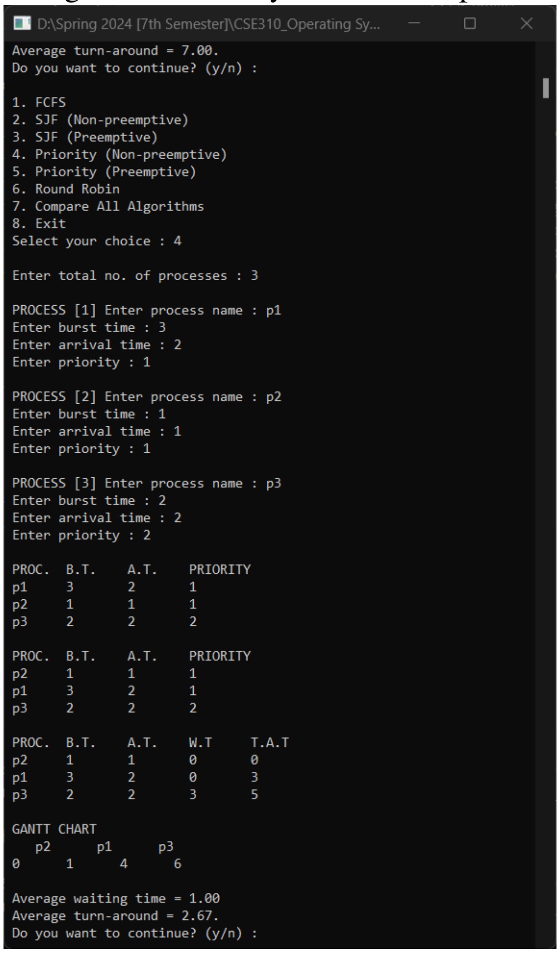
```
7. Compare All Algorithms
8. Exit
Select your choice : 2

Enter total no. of processes : 4

PROCESS [1] Enter process name : p1
Enter burst time : 3
Enter arrival time : 2
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 2
Enter arrival time : 4
Enter priority : 3

PROCESS [3] Enter process name : p3
Enter burst time : 5
Enter arrival time : 1
Enter priority : 1

PROCESS [4] Enter process name : p4
Enter burst time : 4
Enter arrival time : 3
Enter priority : 2

PROC.  B.T.    A.T.    PRIORITY
p1     3       2       1
p2     2       4       3
p3     5       1       1
p4     4       3       2

PROC.  B.T.    A.T.
p3     5       1
p2     2       4
p1     3       2
p4     4       3

PROC.  B.T.    A.T.    W.T     T.A.T
p3     5       1       0       4
p2     2       4       2       4
p1     3       2       6       9
p4     4       3       8       12

GANTT CHART
   p3      p2      p1      p4
0      5      7      10      14

Average waiting time = 4.00
Average turn-around = 7.25.
Do you want to continue? (y/n) :
```

Figure 4.4: SJF -Preemptive



```
Average waiting time = 4.00
Average turn-around = 7.25.
Do you want to continue? (y/n) :

1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
4. Priority (Non-preemptive)
5. Priority (Preemptive)
6. Round Robin
7. Compare All Algorithms
8. Exit
Select your choice : 3

Enter total no. of processes : 4

PROCESS [1] Enter process name : p1
Enter burst time : 3
Enter arrival time : 2
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 2
Enter arrival time : 4
Enter priority : 1

PROCESS [3] Enter process name : p3
Enter burst time : 5
Enter arrival time : 1
Enter priority : 1

PROCESS [4] Enter process name : p4
Enter burst time : 4
Enter arrival time : 3
Enter priority : 2

PROC.  B.T.    A.T.    PRIORITY
p1     3       2       1
p2     2       4       1
p3     5       1       1
p4     4       3       2

PROC.  B.T.    A.T.    W.T     T.A.T
p1     3       2       0       3
p2     2       4       1       3
p3     5       1       5       10
p4     4       3       8       12

Average waiting time = 3.50
Average turn-around = 7.00.
Do you want to continue? (y/n) :
```

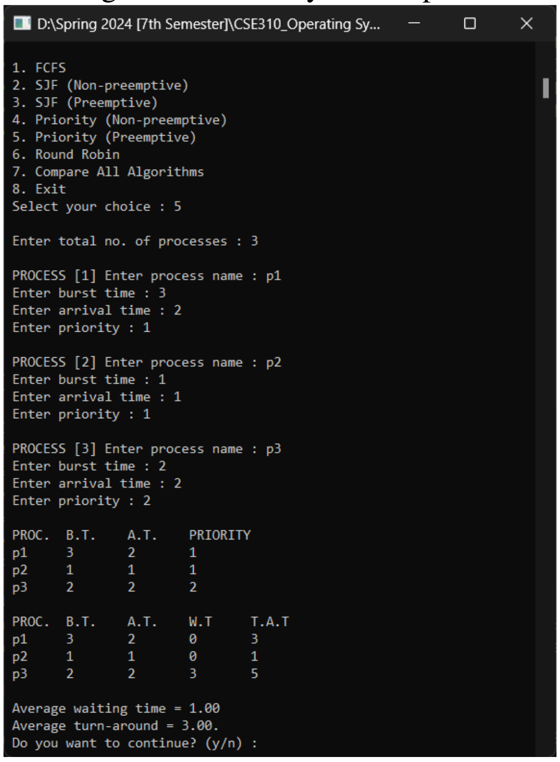
Figure 4.5: Priority - Non-Preemptive



D:\Spring 2024 [7th Semester]\CSE310_Operating Sy...

```
Average turn-around = 7.00.
Do you want to continue? (y/n) :

1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
4. Priority (Non-preemptive)
5. Priority (Preemptive)
6. Round Robin
7. Compare All Algorithms
8. Exit
Select your choice : 4

Enter total no. of processes : 3

PROCESS [1] Enter process name : p1
Enter burst time : 3
Enter arrival time : 2
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 1
Enter arrival time : 1
Enter priority : 1

PROCESS [3] Enter process name : p3
Enter burst time : 2
Enter arrival time : 2
Enter priority : 2

PROC.  B.T.    A.T.    PRIORITY
p1     3       2       1
p2     1       1       1
p3     2       2       2

PROC.  B.T.    A.T.    PRIORITY
p2     1       1       1
p1     3       2       1
p3     2       2       2

PROC.  B.T.    A.T.    W.T     T.A.T
p2     1       1       0       0
p1     3       2       0       3
p3     2       2       3       5

GANTT CHART
   p2      p1      p3
0      1      4      6

Average waiting time = 1.00
Average turn-around = 2.67.
Do you want to continue? (y/n) :
```

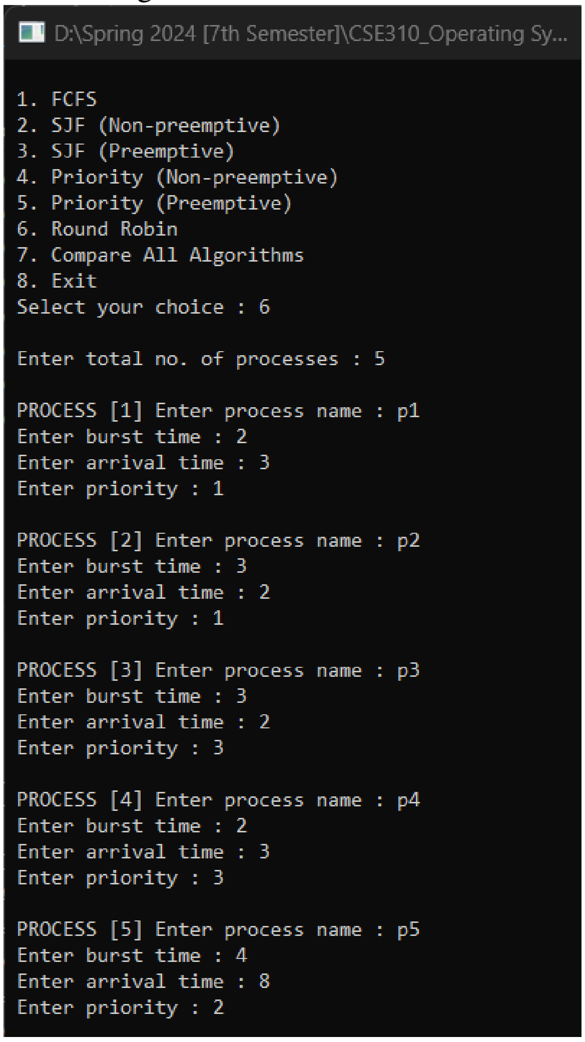
Figure 4.6: Priority -Preemptive



D:\Spring 2024 [7th Semester]\CSE310_Operating Sy...

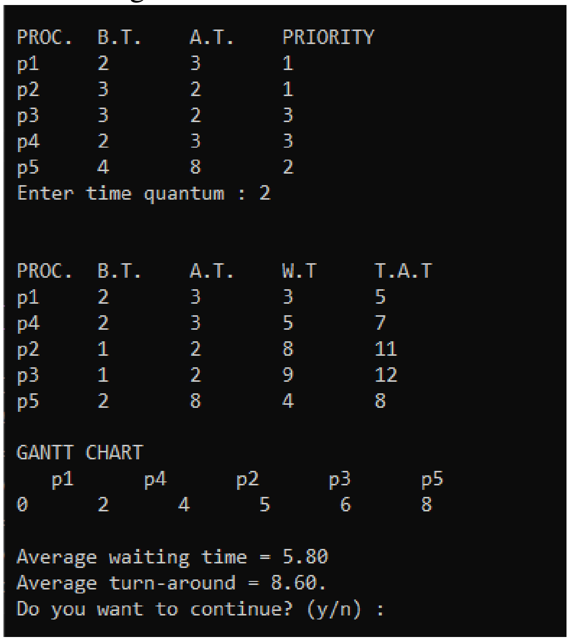
```
1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
4. Priority (Non-preemptive)
5. Priority (Preemptive)
6. Round Robin
7. Compare All Algorithms
8. Exit
Select your choice : 5

Enter total no. of processes : 3

PROCESS [1] Enter process name : p1
Enter burst time : 3
Enter arrival time : 2
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 1
Enter arrival time : 1
Enter priority : 1

PROCESS [3] Enter process name : p3
Enter burst time : 2
Enter arrival time : 2
Enter priority : 2

PROC.  B.T.    A.T.    PRIORITY
p1     3       2       1
p2     1       1       1
p3     2       2       2

PROC.  B.T.    A.T.    W.T     T.A.T
p1     3       2       0       3
p2     1       1       0       1
p3     2       2       3       5

Average waiting time = 1.00
Average turn-around = 3.00.
Do you want to continue? (y/n) :
```

Figure 4.7: Round Robin - 1



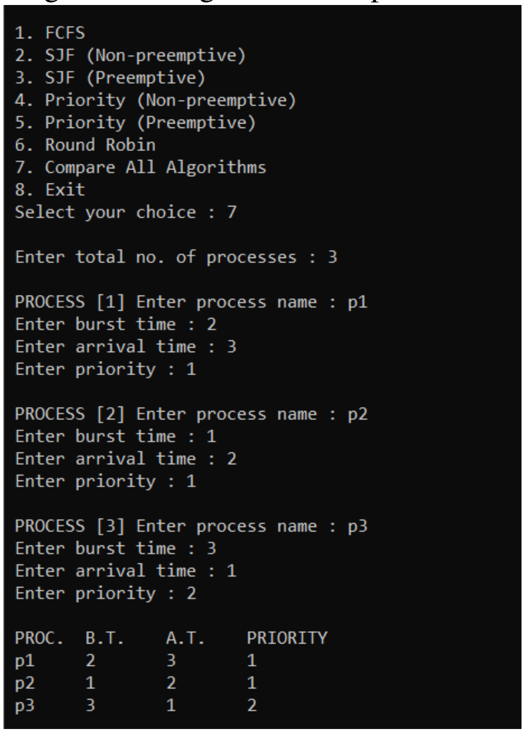
```
D:\Spring 2024 [7th Semester]\CSE310_Operating Sy...

1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
4. Priority (Non-preemptive)
5. Priority (Preemptive)
6. Round Robin
7. Compare All Algorithms
8. Exit
Select your choice : 6

Enter total no. of processes : 5

PROCESS [1] Enter process name : p1
Enter burst time : 2
Enter arrival time : 3
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 3
Enter arrival time : 2
Enter priority : 1

PROCESS [3] Enter process name : p3
Enter burst time : 3
Enter arrival time : 2
Enter priority : 3

PROCESS [4] Enter process name : p4
Enter burst time : 2
Enter arrival time : 3
Enter priority : 3

PROCESS [5] Enter process name : p5
Enter burst time : 4
Enter arrival time : 8
Enter priority : 2
```

Figure 4.8: Round Robin - 2



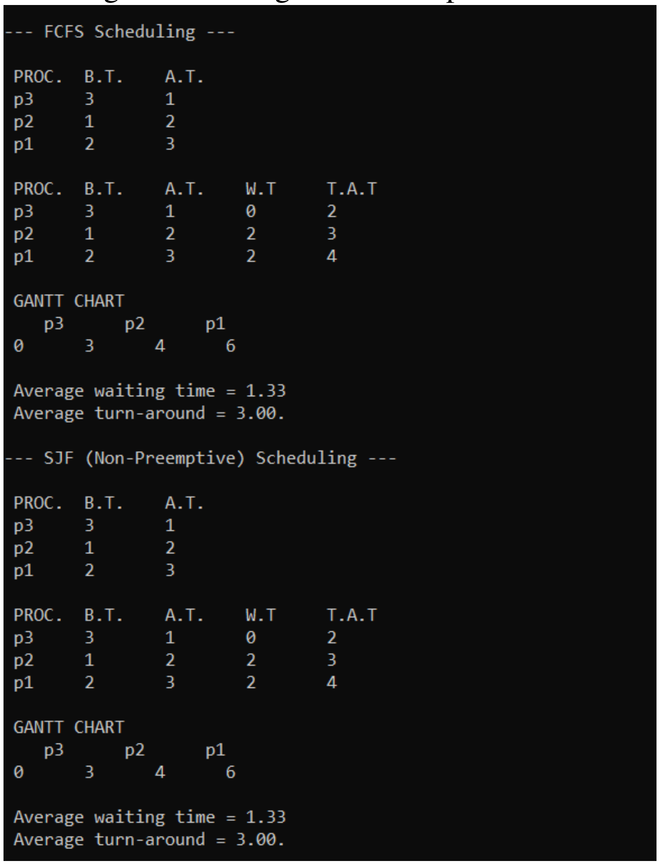
```
PROC.  B.T.    A.T.    PRIORITY
p1     2       3       1
p2     3       2       1
p3     3       2       3
p4     2       3       3
p5     4       8       2
Enter time quantum : 2


PROC.  B.T.    A.T.    W.T     T.A.T
p1     2       3       3       5
p4     2       3       5       7
p2     1       2       8       11
p3     1       2       9       12
p5     2       8       4       8

GANTT CHART
   p1      p4      p2      p3      p5
0      2      4      5      6      8

Average waiting time = 5.80
Average turn-around = 8.60.
Do you want to continue? (y/n) :
```

Figure 4.9: Algorithm Comparison - 1



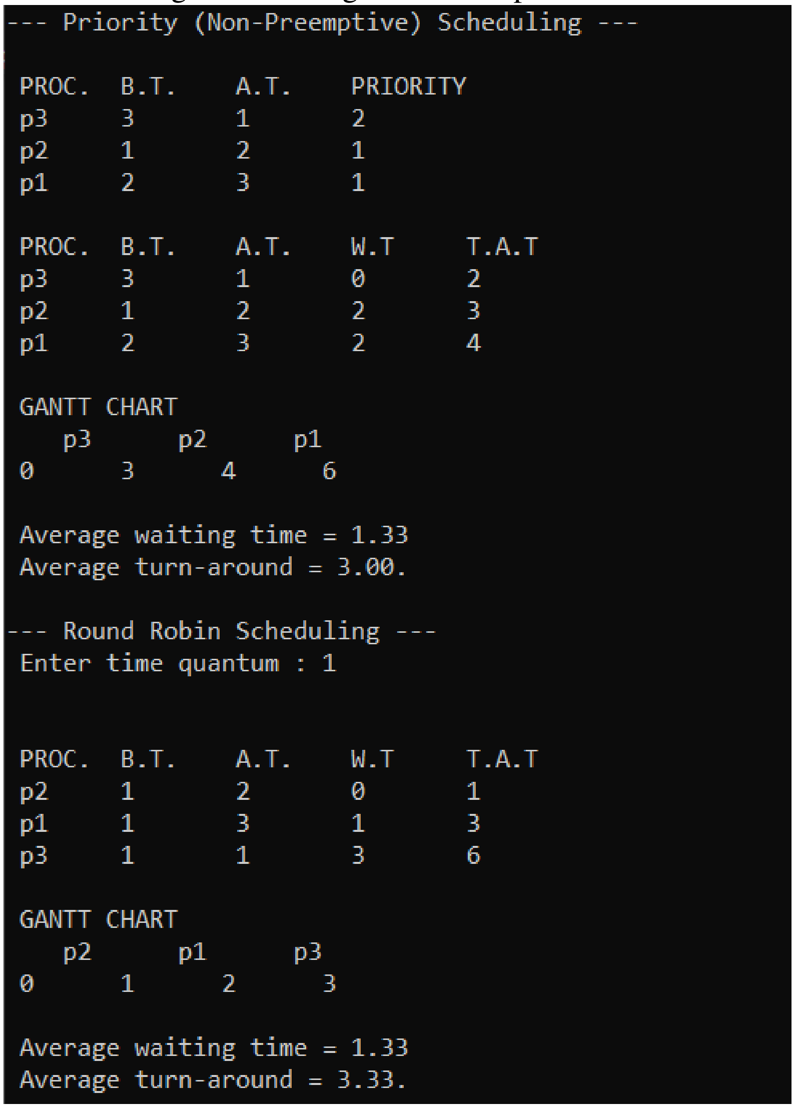
```
1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
4. Priority (Non-preemptive)
5. Priority (Preemptive)
6. Round Robin
7. Compare All Algorithms
8. Exit
Select your choice : 7

Enter total no. of processes : 3

PROCESS [1] Enter process name : p1
Enter burst time : 2
Enter arrival time : 3
Enter priority : 1

PROCESS [2] Enter process name : p2
Enter burst time : 1
Enter arrival time : 2
Enter priority : 1

PROCESS [3] Enter process name : p3
Enter burst time : 3
Enter arrival time : 1
Enter priority : 2

PROC.  B.T.    A.T.    PRIORITY
p1     2       3       1
p2     1       2       1
p3     3       1       2
```

Figure 4.10: Algorithm Comparison - 2



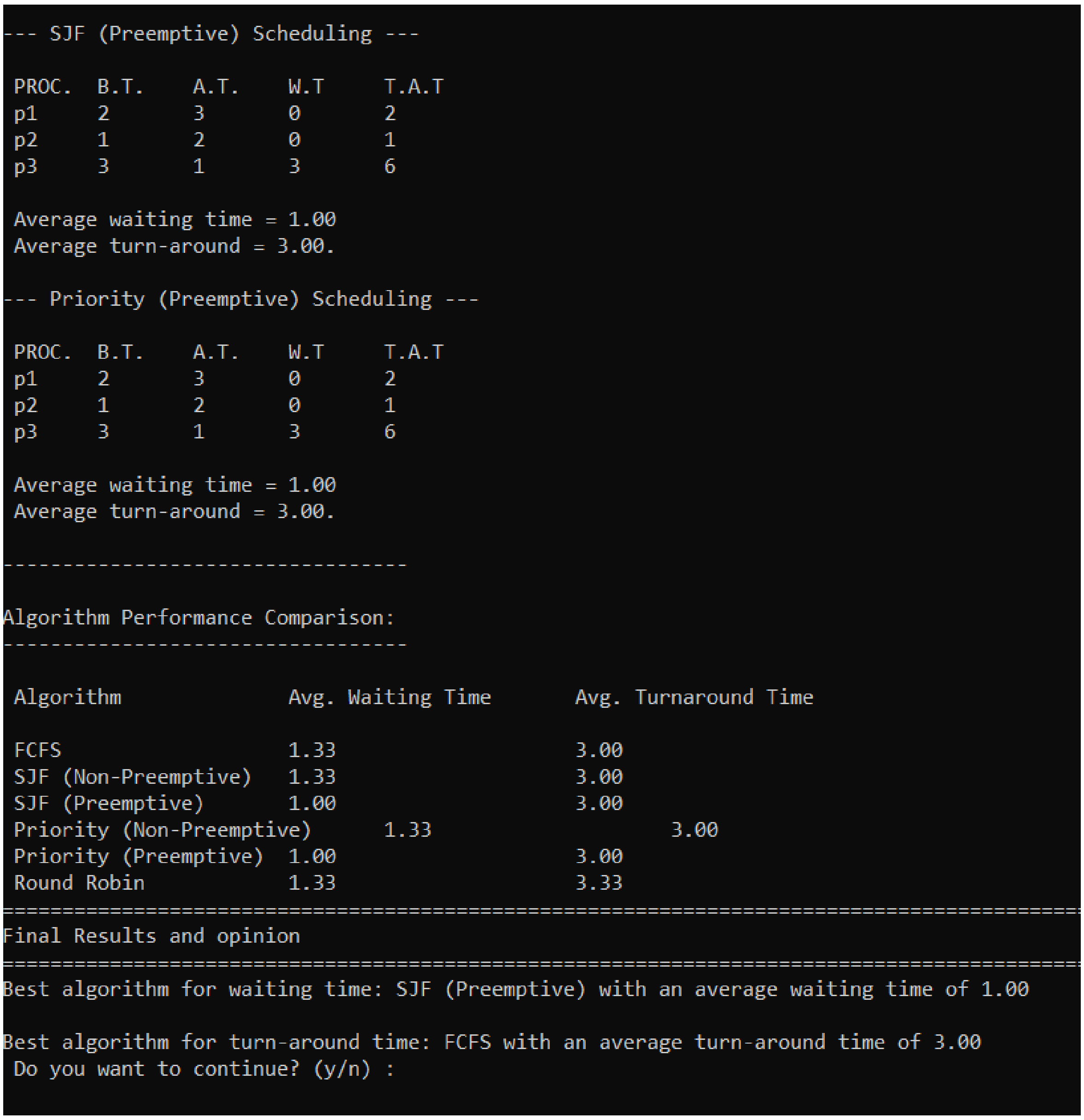
```
--- FCFS Scheduling ---

 PROC.  B.T.    A.T.
 p3     3       1
 p2     1       2
 p1     2       3

 PROC.  B.T.    A.T.    W.T     T.A.T
 p3     3       1       0       2
 p2     1       2       2       3
 p1     2       3       2       4

 GANTT CHART
    p3      p2      p1
 0      3      4      6

 Average waiting time = 1.33
 Average turn-around = 3.00.

--- SJF (Non-Preemptive) Scheduling ---

 PROC.  B.T.    A.T.
 p3     3       1
 p2     1       2
 p1     2       3

 PROC.  B.T.    A.T.    W.T     T.A.T
 p3     3       1       0       2
 p2     1       2       2       3
 p1     2       3       2       4

 GANTT CHART
    p3      p2      p1
 0      3      4      6

 Average waiting time = 1.33
 Average turn-around = 3.00.
```

Figure 4.11: Algorithm Comparison - 3



```
--- Priority (Non-Preemptive) Scheduling ---

 PROC.  B.T.    A.T.    PRIORITY
 p3     3       1       2
 p2     1       2       1
 p1     2       3       1

 PROC.  B.T.    A.T.    W.T     T.A.T
 p3     3       1       0       2
 p2     1       2       2       3
 p1     2       3       2       4

 GANTT CHART
    p3      p2      p1
 0      3      4      6

 Average waiting time = 1.33
 Average turn-around = 3.00.

--- Round Robin Scheduling ---
 Enter time quantum : 1


 PROC.  B.T.    A.T.    W.T     T.A.T
 p2     1       2       0       1
 p1     1       3       1       3
 p3     1       1       3       6

 GANTT CHART
    p2      p1      p3
 0      1      2      3

 Average waiting time = 1.33
 Average turn-around = 3.33.
```

Figure 4.12: Algorithm Comparison - 4



```
--- SJF (Preemptive) Scheduling ---

 PROC.  B.T.    A.T.    W.T     T.A.T
 p1     2       3       0       2
 p2     1       2       0       1
 p3     3       1       3       6

 Average waiting time = 1.00
 Average turn-around = 3.00.

--- Priority (Preemptive) Scheduling ---

 PROC.  B.T.    A.T.    W.T     T.A.T
 p1     2       3       0       2
 p2     1       2       0       1
 p3     3       1       3       6

 Average waiting time = 1.00
 Average turn-around = 3.00.

----------------------------------

Algorithm Performance Comparison:
----------------------------------

 Algorithm              Avg. Waiting Time       Avg. Turnaround Time

 FCFS                   1.33                    3.00
 SJF (Non-Preemptive)   1.33                    3.00
 SJF (Preemptive)       1.00                    3.00
 Priority (Non-Preemptive)      1.33                    3.00
 Priority (Preemptive)  1.00                    3.00
 Round Robin            1.33                    3.33
==========================================================================================
Final Results and opinion
==========================================================================================
Best algorithm for waiting time: SJF (Preemptive) with an average waiting time of 1.00

Best algorithm for turn-around time: FCFS with an average turn-around time of 3.00
 Do you want to continue? (y/n) :
```

Chapter 5

Conclusion

5.1 Discussion

The project offers insights into CPU scheduling algorithms, highlighting trade-offs in system performance metrics. Understanding algorithmic behavior is aided by the variation of process scheduling that is revealed through experimentation.

5.2 Limitations

The **absence of a graphical interface (GUI)** restricts visualization options and user interaction. **Real-time scheduling algorithms are not supported** (it is a static simulation) which limits their applicability to time sensitive scenarios. Addressing these gaps could enhance usability and broaden the simulator's scope.

5.3 Scope of Future Work

1. Development of a graphical user interface (GUI)
2. Integrating support for real-time scheduling algorithms and additional scheduling policies.
3. Incorporating advanced visualization techniques and performance analysis tools.

5.4 Conclusion / Remarks

In conclusion, the CPU Scheduling (A Console Based Simulator) project has contributed to the understanding and exploration of CPU scheduling algorithms. Although the current version of the simulator serves as a valuable educational and research tool, there is room for growth and refinement. By addressing the identified limitations and pursuing future enhancements, we can continue to advance our understanding of CPU scheduling and its role in operating system design and optimization.