**Programming Assignment 1**

**Project Report**

|  |  |
| --- | --- |
| **Section#:52586**  **Group#: 4** | |
| **Name** | **ID** |
| **Ghaida Alhussain (Leader)** | **444201213** |
| **Fajer Alamro** | **444200800** |
| **Sara Alhowaimel** | **444200462** |
| **Deem Aljarba** | **444200523** |

**Supervised by: Dr.Amani Alahmadi**

**Task distribution:**

|  |  |
| --- | --- |
| **Name** | **Task** |
| **Ghaida Alhussain** | Implemented the **Process class** and debugging, contributed to**report writing (Implementation section)**. |
| **Fajer Alamro** | Implemented part of the**Process execution and performance calculations**, contributed to**report writing (**Conclusion section) |
| **Sara Alhowaimel** | Implemented part of the **Scheduling class** (SRTF & FCFS logic), contributed to**report writing (**Introduction section). |
| **Deem Aljarba** | Implemented part of the **user input handling and Gantt chart generation,** contributed to**report writing (**Result and Analysis section) |

Table of Contents

[Introduction: 4](#_Toc191689648)

[Implementation: 4](#_Toc191689649)

[Result and Analysis: 5](#_Toc191689650)

[Full Program Execution Output: 7](#_Toc191689651)

# **Introduction:**

Efficient process scheduling is a fundamental aspect of modern operating systems, ensuring optimal CPU utilization and minimizing process waiting time. This project focuses on implementing the**Shortest Remaining Time First (SRTF) scheduling algorithm**, incorporating **First-Come, First-Served (FCFS) scheduling** for processes with equal CPU burst times.

The goal is to analyze the **performance** of this scheduling approach by evaluating key metrics such as **CPU utilization, average turnaround time, and average waiting time**. Additionally, the project illustrates how **context switching** affects scheduling efficiency and demonstrates the effectiveness of combining **SRTF and FCFS** to handle processes with varying execution requirements.

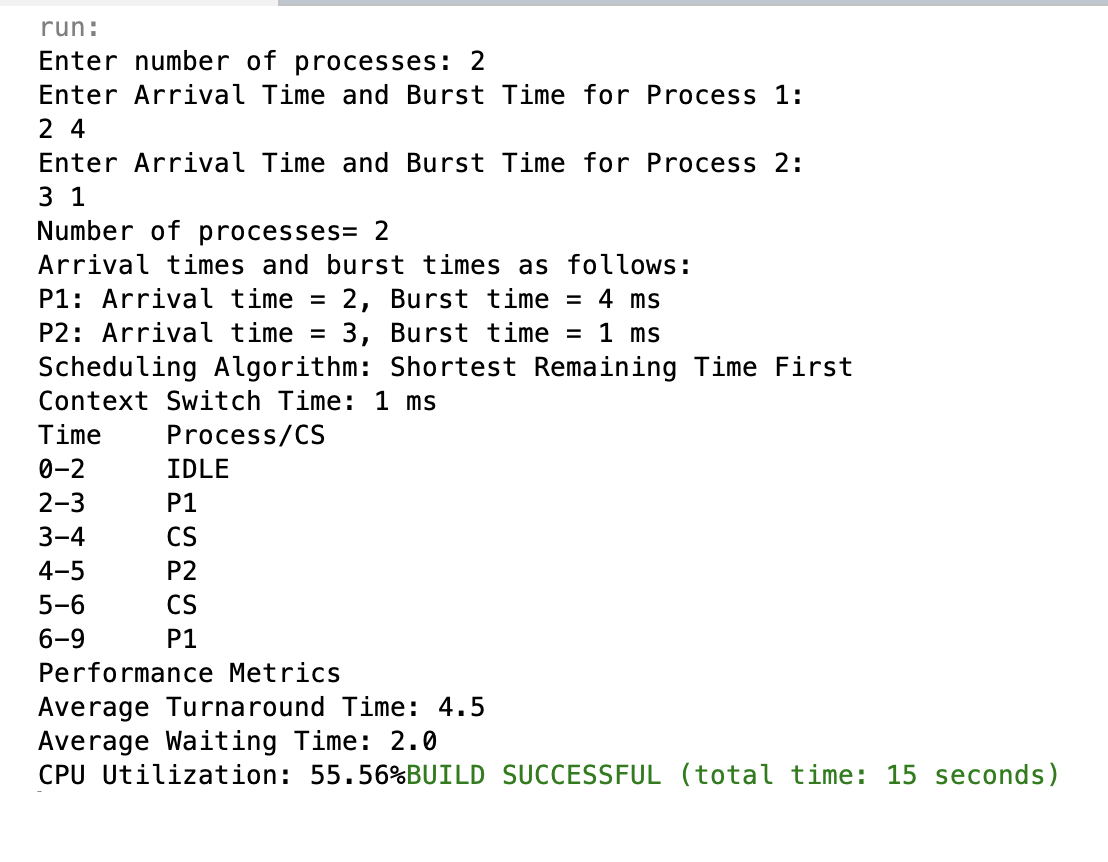
# **Implementation:**

**The program consists of three main classes:**

* **OSProject Class:** Handles user input and initializes processes.
* **Process Class:** Represents a Process with the following attributes:
* **ID**: A unique identifier for each process.
* **ArrivalTime**
* **BurstTime**
* **remainingTime**
* **completionTime**
* **waitingTime:** calculated as: Waiting Time=Turnaround Time−Burst Time
* **turnaroundTime:** calculated as: Turnaround Time=Completion Time−Arrival Time
* **Scheduling Class:** Implements the SRTF scheduling algorithm with FCFS for processes with equal remaining time, executes processes, and calculates performance metrics. It’s also included attributes such that:
* **processes:** An array that holds the list of processes to be scheduled.
* **ContexSwitch:** A constant representing the **context switch time**, which is the time required to switch from one process to another (set to 1 millisecond).

# **Result and Analysis:**

* **Execution Process and Analysis:**

1. The program prompts the user to enter the number of processes, followed by the **arrival time** and **burst time** for each process. These inputs are then stored in an array for scheduling.

**Process 1 (P1):** Arrival Time = 2 ms, Burst Time = 4 ms

**Process 2 (P2):** Arrival Time = 3 ms, Burst Time = 1 ms

The program then applies **Shortest Remaining Time First (SRTF)** scheduling, using **First-Come, First-Served (FCFS)**when two processes have the same remaining time.

1. After scheduling the processes, the program generates the following **Gantt Chart**, which represents the execution timeline:

|  |  |
| --- | --- |
| Time | Process/CS/Idle |
| **0-2 ms** | The CPU remains **IDLE** because no process has arrived yet. |
| **2-3 ms** | **P1 starts execution**, but since**P2 arrives at 3ms** and has a shorter burst time, the CPU **preempts P1** and switches to P2. |
| **3-4 ms** | **A context switch (CS) occurs** to transfer control to P2. |
| **4-5 ms** | **P2 executes and completes**, since it only required **1ms.** |
| **5-6 ms** | Another **context switch** occurs before P1 resumes execution. |
| **6-9 ms** | **P1 resumes execution and completes at 9ms.** |

**A screen shot of a computer

Description automatically generated**This demonstrates how **SRTF prioritizes the process with the shortest remaining time**, leading to **preemption** when a shorter job arrives.

1. After execution, the program calculates key performance metrics:

**A screen shot of a computer

Description automatically generated**

* **Average Turnaround Time (TAT):  On average, each process spends 4.5ms from arrival to completion.** This is relatively low due to SRTF's ability to minimize wait times.
* **Average Waiting Time (WT): T**he average waiting time is 2.0ms, meaning that processes experience moderate delays before execution starts.
* **CPU Utilization:**The CPU was actively executing processes **55.56%** of the time, meaning that a significant portion was spent on context switching and idle time (nearly 44%).

# **Full Program Execution Output:**

**A screen shot of a computer

Description automatically generated**

Finally, we realized that while the**SRTF scheduling algorithm** effectively reduces waiting and turnaround times by prioritizing shorter processes, it also increases **context switching overhead**, which impacts CPU efficiency. If fewer preemptions occurred, CPU utilization could be improved, especially as the number of processes grows. This highlights the importance of balancing **efficiency and fairness**in process scheduling.