King Saud University
College of Computer and Information Sciences
Information Technology Department

كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

CSC227: Operating System

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

King Saud University
College of Computer and Information Sciences
Information Technology Department

كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

CSC227: Operating System

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)

King Saud University
College of Computer and Information Sciences
Information Technology Department

كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

CSC227: Operating System

Table of Contents

King Saud University
College of Computer and Information Sciences
Information Technology Department

كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

CSC227: Operating System

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.

```
run:
Enter number of processes: 2
Enter Arrival Time and Burst Time for Process 1:
2 4
Enter Arrival Time and Burst Time for Process 2:
3 1
```

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.

2. 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**.**

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

```
Time    Process/CS
0-2     IDLE
2-3     P1
3-4     CS
4-5     P2
5-6     CS
6-9     P1
```

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

```
Performance Metrics
Average Turnaround Time: 4.5
Average Waiting Time: 2.0
CPU Utilization: 55.56%BUILD
```

- 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): The 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:

```
run:
Enter number of processes: 2
Enter Arrival Time and Burst Time for Process 1:
2 4
Enter Arrival Time and Burst Time for Process 2:
3 1
Number of processes= 2
Arrival times and burst times as follows:
P1: Arrival time = 2, Burst time = 4 ms
P2: Arrival time = 3, Burst time = 1 ms
Scheduling Algorithm: Shortest Remaining Time First
Context Switch Time: 1 ms
Time    Process/CS
0-2     IDLE
2-3     P1
3-4     CS
4-5     P2
5-6     CS
6-9     P1
Performance Metrics
Average Turnaround Time: 4.5
Average Waiting Time: 2.0
CPU Utilization: 55.56%BUILD SUCCESSFUL (total time: 15 seconds)
```

Figure1: Program execution with two processes using SRTF scheduling.

```
run:
Enter number of processes: 3
Enter Arrival Time and Burst Time for Process 1:
1 4
Enter Arrival Time and Burst Time for Process 2:
2 3
Enter Arrival Time and Burst Time for Process 3:
1 6
Number of processes= 3
Arrival times and burst times as follows:
P1: Arrival time = 1, Burst time = 4 ms
P2: Arrival time = 2, Burst time = 3 ms
P3: Arrival time = 1, Burst time = 6 ms
Scheduling Algorithm: Shortest Remaining Time First
Context Switch Time: 1 ms
Time    Process/CS
0-1     IDLE
1-5     P1
5-6     CS
6-9     P2
9-10    CS
10-16   P3
Performance Metrics
Average Turnaround Time: 8.666666666666666
Average Waiting Time: 4.333333333333333
CPU Utilization: 81.25%BUILD SUCCESSFUL (total time: 18 seconds)
```

*Figer2: Example of program execution with three processes demonstrating **SRTF** scheduling with **FCFS** for equal arrival times. The Gantt Chart shows preemptions and context switching with a 1ms delay.*

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. Additionally, incorporating FCFS for processes with equal burst times ensures fair execution, preventing starvation and maintaining process order. This highlights the importance of balancing efficiency, fairness, and CPU performance in process scheduling.