NMIMS University

SVKM'S NMIMS Deemed to be UNIVERSITY

Project Title:

Shortest Remaining Time First (SRTF)

Course: **Operating system**

Submitted by:

Team Members:

1. **Prasad Kannawar (70572200034)**
2. **Ruthvik Akula (70572200028)**
3. **M Vaishnavi (70572200012)**
4. **Lokeshwar (70572200029)**
5. **Sainath (70572200047)**

Guided by:

Professor: Wasiha Tasmeen

Report on Shortest Remaining Time First (SRTF) Scheduling Algorithm

Introduction

The Shortest Remaining Time First (SRTF) scheduling algorithm is a preemptive CPU scheduling technique. It is a variation of the Shortest Job Next (SJN) algorithm, but with the key difference that it allows a process to be interrupted if a new process arrives with a shorter burst time than the currently running process. The goal of SRTF is to minimize the average waiting time of all processes by prioritizing the processes that require less CPU time to execute.

Steps in SRTF Scheduling

Step 1: Understanding SRTF Scheduling

SRTF is a preemptive scheduling algorithm where the CPU is assigned to the process with the shortest remaining burst time. At every unit of time, the algorithm compares the remaining burst times of all processes that have already arrived and selects the one with the shortest remaining time for execution. If a new process arrives with a shorter burst time than the current running process, the CPU is preempted, and the new process is executed.

Step 2: Gathering Input

To implement the SRTF algorithm, certain details for each process must be collected:

- **Process ID**: A unique identifier for each process.
- **Arrival Time**: The time at which the process arrives in the system.
- **Burst Time**: The total CPU time required by the process for execution.

These details are crucial as they define when a process is eligible for execution and how long it needs to run.

Step 3: Initializing Variables

Several variables must be initialized to start the process scheduling:

- **Remaining Time**: Initially set to the burst time of each process. This value will be reduced as the process executes.
- **Current Time**: The clock that tracks the time within the system, starting from zero.
- **Completed Processes**: A count of how many processes have been completed.
- **Total Waiting Time and Total Turnaround Time**: These are accumulated to calculate average waiting and turnaround times later.

Step 4: Process Execution Loop

The scheduling algorithm works by continuously selecting the process with the shortest remaining burst time:

1. **Selecting the Process**: Among the processes that have arrived and are ready to execute, the one with the shortest remaining time is selected. If multiple processes have the same remaining time, the one that arrived first is chosen.
2. **Executing the Process**: The selected process executes for one time unit, and its remaining burst time decreases by 1.
3. **Completion**: When the remaining burst time of a process becomes zero, the process is considered completed. The completion time is recorded, and the waiting and turnaround times are calculated.
4. **Incrementing the Time**: The current time is increased by one unit, and the process repeats until all processes are completed.

Step 5: Repeat Until All Processes Are Completed

The loop continues until every process has been executed. The algorithm keeps track of the processes that have arrived and ensures that at each unit of time, the process with the shortest remaining time is always executed.

Step 6: Calculate Averages

After all processes have been executed, the following averages are calculated:

- **Average Waiting Time**: The total waiting time of all processes divided by the number of processes.
- **Average Turnaround Time**: The total turnaround time of all processes divided by the number of processes.

These averages provide insights into the efficiency of the algorithm.

Example and Analysis

Consider a set of processes with their burst times and arrival times as shown below:

Process ID	**Burst Time**	**Arrival Time**
P1	6	0
P2	8	1
P3	7	2
P4	3	3

Explanation of Execution:

- At time 0, process P1 arrives and begins execution as it is the only process.
- At time 1, process P2 arrives, but since P1 still has a remaining time of 5 units, it continues executing.

- At time 2, process P3 arrives. Process P1 has a remaining burst time of 4, and process P3 has a burst time of 7. Since P1 has the shortest remaining time, it continues executing.
- At time 3, process P4 arrives. Process P4 has the shortest burst time of 3 units, so it preempts process P1 and starts execution.
- This process of selecting the process with the shortest remaining burst time continues until all processes are completed.

The final waiting times, turnaround times, and Gantt chart can be generated to visualize the process execution and analyze performance.

Average Waiting Time and Turnaround Time

After all processes are executed, the average waiting time and average turnaround time are computed. These metrics are crucial for understanding the efficiency of the scheduling algorithm:

- **Average Waiting Time**: The average time processes spend in the waiting queue before they get executed.
- **Average Turnaround Time**: The average time it takes for a process to go from arrival to completion, including both waiting time and execution time.

Conclusion

The SRTF scheduling algorithm is highly effective in reducing average waiting times, as it always prioritizes the process with the shortest remaining burst time. This preemptive approach allows it to minimize the time a process waits in the queue, thus improving the overall system throughput and efficiency. However, the algorithm can be more complex to implement compared to non-preemptive algorithms, especially when managing processes with varying burst times and arrival times.

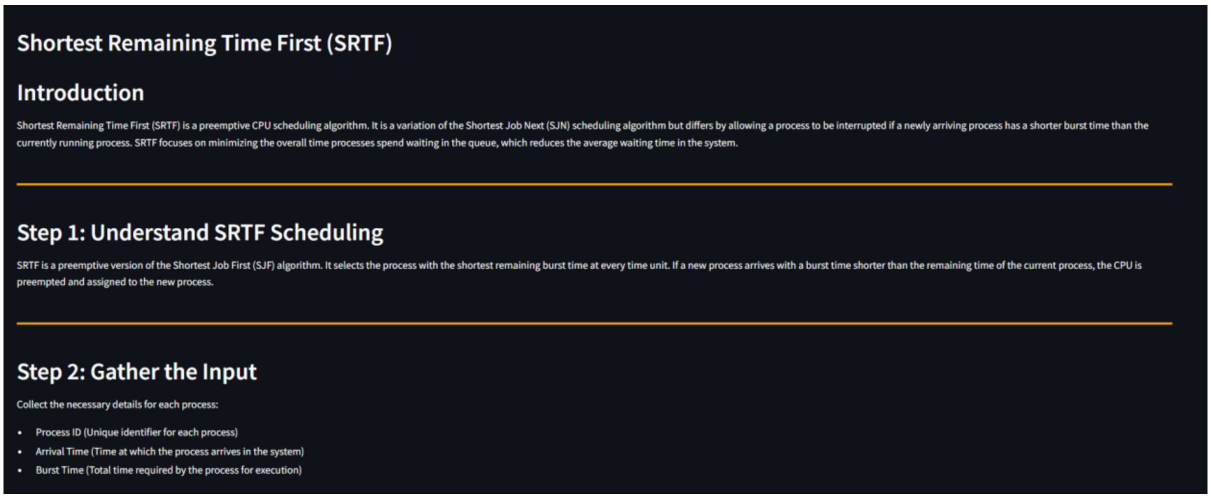
This scheduling method is particularly beneficial in environments where short tasks arrive frequently and need to be processed quickly, minimizing delays for short jobs while also keeping longer jobs in the queue.

Further Reading

For more detailed information on CPU scheduling algorithms, you can explore the following resources:

- Operating Systems: Three Easy Pieces
- Scheduling Algorithms in Operating Systems

GUI:



Shortest Remaining Time First (SRTF)

Introduction

Shortest Remaining Time First (SRTF) is a preemptive CPU scheduling algorithm. It is a variation of the Shortest Job Next (SJN) scheduling algorithm but differs by allowing a process to be interrupted if a newly arriving process has a shorter burst time than the currently running process. SRTF focuses on minimizing the overall time processes spend waiting in the queue, which reduces the average waiting time in the system.

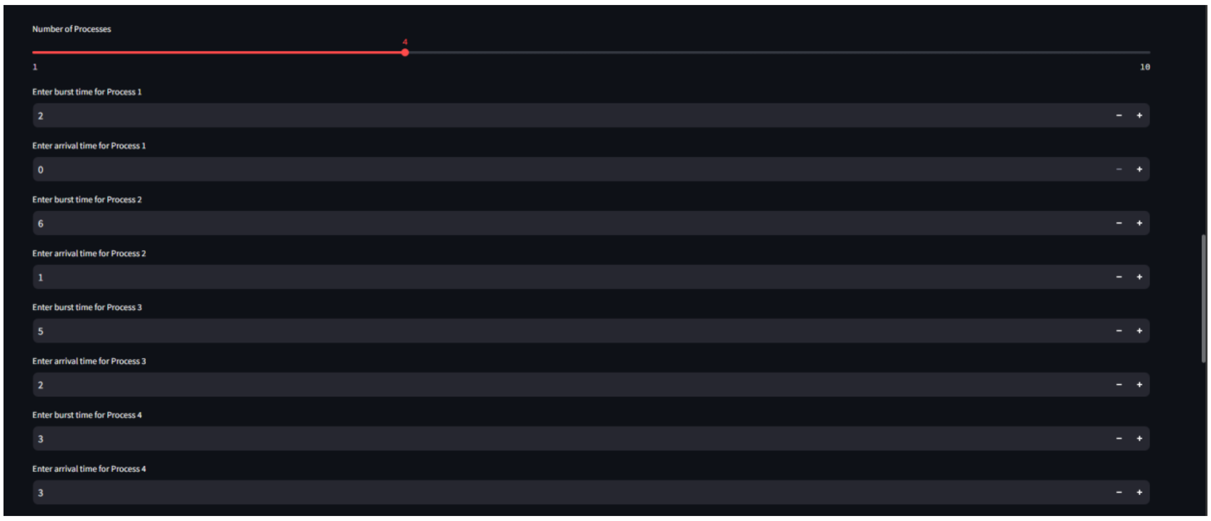
Step 1: Understand SRTF Scheduling

SRTF is a preemptive version of the Shortest Job First (SJF) algorithm. It selects the process with the shortest remaining burst time at every time unit. If a new process arrives with a burst time shorter than the remaining time of the current process, the CPU is preempted and assigned to the new process.

Step 2: Gather the Input

Collect the necessary details for each process:

- Process ID (Unique identifier for each process)
- Arrival Time (Time at which the process arrives in the system)
- Burst Time (Total time required by the process for execution)



Number of Processes: 4 (1–10)

Enter burst time for Process 1: 2

Enter arrival time for Process 1: 0

Enter burst time for Process 2: 6

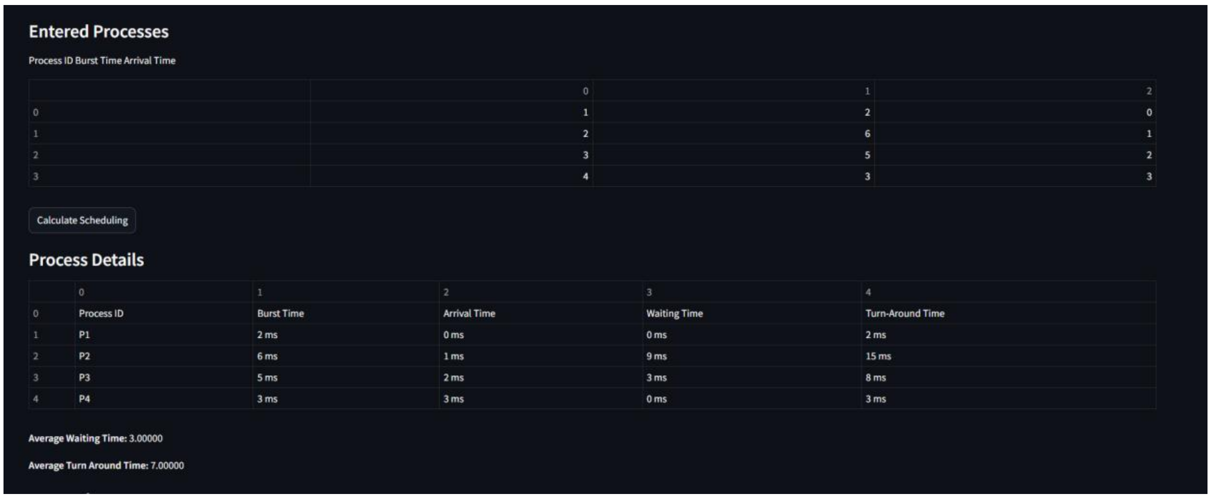
Enter arrival time for Process 2: 1

Enter burst time for Process 3: 5

Enter arrival time for Process 3: 2

Enter burst time for Process 4: 3

Enter arrival time for Process 4: 3



Entered Processes

Process ID Burst Time Arrival Time

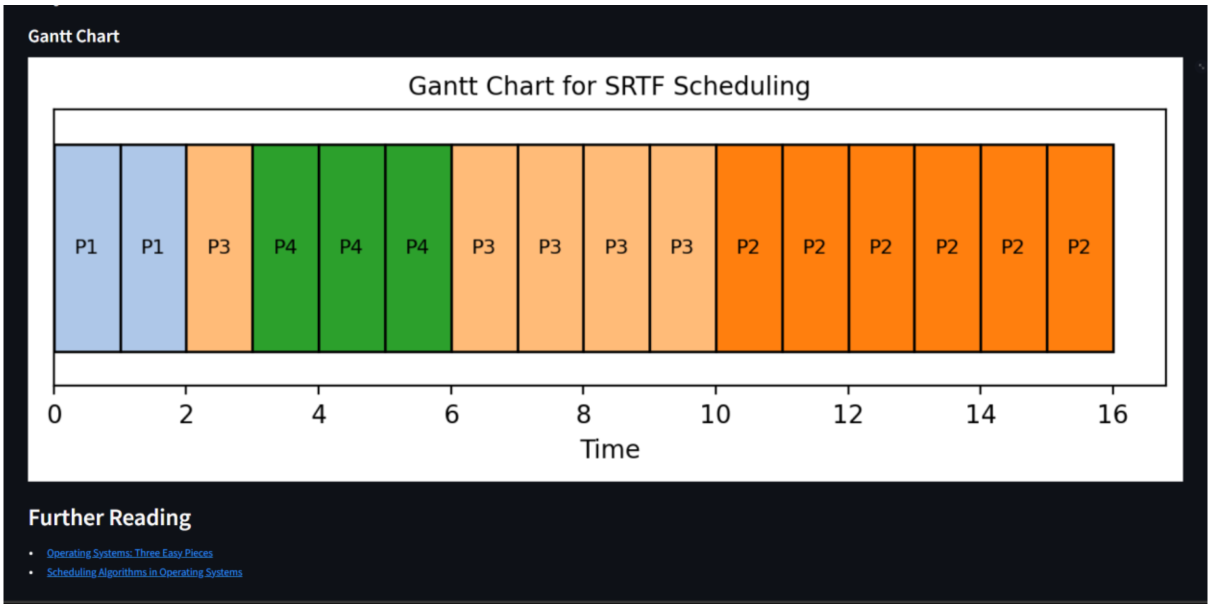
	0	1	2
0	1	2	0
1	2	6	1
2	3	5	2
3	4	3	3

Process Details

	0	1	2	3	4
0	Process ID	Burst Time	Arrival Time	Waiting Time	Turn-Around Time
1	P1	2 ms	0 ms	0 ms	2 ms
2	P2	6 ms	1 ms	9 ms	15 ms
3	P3	5 ms	2 ms	3 ms	8 ms
4	P4	3 ms	3 ms	0 ms	3 ms

Average Waiting Time: 3.00000

Average Turn Around Time: 7.00000



Gantt Chart

Further Reading

- Operating Systems: Three Easy Pieces
- Scheduling Algorithms in Operating Systems

Link: https://github.com/Prasadkannawar/OS-Project