Assignment 3: Implementing CPU Scheduling Algorithms in C

Objective: The objective of this assignment is to implement and analyze different CPU scheduling algorithms: Round Robin (RR) and Shortest Remaining Time First (SRTF). Students will explore different process scenarios, analyze the performance metrics of each algorithm, and critically evaluate the trade-offs involved.

Instructions

1. Referring to the `schedule.c` source file provided for FCFS, implement the two CPU scheduling algorithms (RR and SRTF) in C.
2. Use structures to define the Process Control Block (PCB) and include additional fields as necessary.
3. Create interface for user inputs to facilitate testing of various scenarios.
4. Implement input validation to ensure that the data entered is valid.
5. Test your implementation with a variety of process inputs and edge cases (e.g., processes with the same arrival times, and varying burst times).
6. Answer the analytical questions at the end of this document and submit your code along with your answers via GitHub following the submission guidelines given at the end of this document.

General Requirements

- Use the following structure for the PCB.

```
struct PCB {
    int process_id;
    int arrival_timestamp;
    int total_bursttime;
    int remaining_bursttime;
    int execution_starttime;
    int execution_endtime;
    int waiting_time;     // For calculating average waiting time
    int turnaround_time; // For calculating average turnaround time
};
```

Specific Requirements

1. **Round Robin Scheduling**:
    - Implement a function to simulate the Round Robin scheduling algorithm.
    - Allow the user to specify the quantum time.
    - Keep track of waiting time and turnaround time for each process.
    - Display a Gantt chart representation of the process execution.
2. **Shortest Remaining Time First (SRTF)**:
    - Implement a function to simulate the SRTF scheduling algorithm.
    - Handle scenarios where processes arrive while others are executing.
    - Calculate and display waiting time and turnaround time for each process.
    - Include a visual representation such as Gantt chart showing the execution order.

Input and Output

- Create an interface for users to select the scheduling algorithm and input process details (Process ID, Arrival Time, Burst Time etc.).
- Ensure the program displays:
    - The order of process execution.
    - Gantt chart for each scheduling algorithm.
    - Average waiting time and turnaround time for all processes.

Sample Input (RR)

RR	SRTF

`Enter Quantum: 2`
`Enter the number of processes: 3`
`Enter Process ID: 1`
`Enter Arrival Time: 0`
`Enter Total Burst Time: 8`
 	
`Enter Process ID: 2`
`Enter Arrival Time: 1`
`Enter Total Burst Time: 4`
 	
`Enter Process ID: 3`
`Enter Arrival Time: 2`
`Enter Total Burst Time: 9`
`...`
	`Enter the number of processes: 3`
	`Enter Process ID: 1`
	`Enter Arrival Time: 0`
	`Enter Total Burst Time: 8`
	
	`Enter Process ID: 2`
	`Enter Arrival Time: 1`
	`Enter Total Burst Time: 4`
	
	`Enter Process ID: 3`
	`Enter Arrival Time: 2`
	`Enter Total Burst Time: 9`
	
	`...`

Analytical Questions

Following your implementation logic, answer the following questions:

1. **Round Robin Scheduling**:
    - Discuss how different quantum times affect the waiting time and turnaround time of processes.
    - In your experiments, what scenarios led to increased context switching, and how did that impact performance?
2. **Shortest Remaining Time First (SRTF)**:
    - Compare SRTF with non-preemptive Shortest Job First (SJF). What are the key differences in terms of responsiveness and turnaround time?
    - How did process arrival times influence the performance of the SRTF algorithm?
3. **General Questions**:
    - Based on your implementations, which scheduling algorithm exhibited the best average waiting time and turnaround time? Provide data to support your conclusion.
    - Discuss the trade-offs between CPU utilization and response time for each algorithm. Which algorithm would you recommend for a real-time system and why?

Submission Guidelines

- Submit your C source files for both RR and SRTF with names as `rr.c` and `srtf.c` via GitHub.
- Include a README file explaining how to run your program and any assumptions made. Also, provide a separate document with your answers to the analytical questions.
- Ensure your code compiles and runs without errors before submission.
- DUE DATE: **30th October** (Wednesday) by the end of day.