1. Description of the Program:

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

CPU Scheduling is a comprehensive project aimed at implementing various CPU scheduling algorithms. It facilitates the simulation and analysis of First-Come-First-Served (FCFS), Round-Robin (RR), Shortest Process Next (SPN), and Shortest Remaining Time (SRT) scheduling policies. The program accepts input data containing process arrival times and service times, executes the specified scheduling algorithm, and produces output detailing process scheduling and performance metrics.

2. Algorithm:

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

The CPU Scheduling project employs efficient algorithmic structures to ensure optimal execution and analysis of scheduling policies.

a. Input Parsing and Initialization:

- Parsing input data and initializing data structures operate in linear time complexity, O(n), where n is the number of processes.

b. Scheduling Algorithm Execution:

- The execution of scheduling algorithms varies:

- FCFS: Operates in linear time complexity, O(n), where n is the number of processes.

- RR: Exhibits linear time complexity, O(n \* m), where n is the number of processes and m is the time quantum.

- SPN and SRT: Utilize sorting algorithms with time complexity O(n log n), followed by linear time complexity for scheduling, O(n).

c. Output Generation:

- Generating output detailing process scheduling and performance metrics operates in linear time complexity, O(n), where n is the number of processes.

3. Analysis of the Algorithm and Output:

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

The algorithmic analysis provides insights into the efficiency and effectiveness of CPU scheduling policies.

a. Scheduling Policy Performance:

- Each scheduling policy demonstrates distinct performance characteristics, influenced by factors such as process arrival times and service times.

b. Turnaround Time and Waiting Time:

- Analysis of turnaround time and waiting time distributions offers insights into scheduling policy effectiveness and resource utilization.

c. Average Performance Metrics:

- Calculation and display of global average turnaround time, normalized turnaround time, and waiting time provide a comprehensive overview of scheduling policy performance.

4. Performance Description for the Entire Project:

Evaluation of the project's performance extends beyond algorithmic efficiency to encompass broader aspects of usability and effectiveness.

a. Scalability and Responsiveness:

- The project demonstrates scalability and responsiveness, accommodating varying numbers of processes and scheduling policies with minimal impact on performance.

b. Output Clarity and Interpretability:

- Output detailing process scheduling and performance metrics is clear and interpretable, facilitating analysis and decision-making.

c. Error Handling and Resilience:

- Robust error handling mechanisms ensure program resilience, enhancing user experience and reliability.

Graphs & Data:

SRT:

sajjadalsaffar@Sajjads-MacBook-Air project2 % ./mydispatcher input.dat FCFS

Average turnaround time = 4760.70

Average normalized turnaround time = 0.48

Average waiting time = 4755.76

RR q= 2:

sajjadalsaffar@Sajjads-MacBook-Air project2 % ./mydispatcher input.dat RR 2

Average turnaround time = 13013.80

Average normalized turnaround time = 1.30

Average waiting time = 13008.85

RR q = 4:

sajjadalsaffar@Sajjads-MacBook-Air project2 % ./mydispatcher input.dat RR 4

Average turnaround time = 7958.60

Average normalized turnaround time = 7958.60

Average waiting time = 7953.65

SPN:

sajjadalsaffar@Sajjads-MacBook-Air project2 % ./mydispatcher input.dat SPN

Average turnaround time = 4760.70

Average normalized turnaround time = 4760.70

Average waiting time = 4755.76

SRT:

sajjadalsaffar@Sajjads-MacBook-Air project2 % ./mydispatcher input.dat SRT

Average turnaround time = 1288.34

Average normalized turnaround time = 1288.34

Average waiting time = 1283.40

A graph with a line

Description automatically generatedFCFS:

A graph with a line

Description automatically generatedA graph with a blue line

Description automatically generated

A graph with a line going up

Description automatically generatedA graph with a line going up

Description automatically generatedRR quantom size 4:

A graph with a line going up

Description automatically generated

SRT:

A graph with a line going up

Description automatically generatedA graph with a line going up

Description automatically generatedA graph with a line going up

Description automatically generated

SPN:

A graph with a line

Description automatically generatedA graph with a line

Description automatically generatedA graph with a blue line

Description automatically generated

5. **Comparison of Scheduling Algorithms**

In our analysis of scheduling algorithms, we examined four key performance metrics: average turnaround time, average waiting time, throughput, and responsiveness. Each metric provides valuable insights into how well a scheduling algorithm manages system resources and processes.

**1. Average Turnaround Time Comparison:**

* **FCFS (First-Come, First-Served):** FCFS scheduling exhibits a straightforward approach, executing processes in the order of their arrival. While FCFS is simple to implement and fair to all processes, it often results in longer average turnaround times, especially when there's a mix of short and long processes. This is due to the phenomenon known as the "convoy effect," where short processes get stuck behind long ones.
* **RR (Round-Robin):** RR scheduling introduces time slicing, allowing each process to execute for a fixed time quantum before being preempted. This mechanism ensures fairness and prevents starvation. However, RR may lead to higher average turnaround times compared to more efficient algorithms, especially with shorter time quantum sizes.
* **SPN (Shortest Process Next):** SPN selects the process with the shortest expected remaining time to execute next. This approach minimizes turnaround times by prioritizing short processes, leading to better overall performance. However, SPN may suffer from starvation for longer processes if new short processes continuously arrive.
* **SRT (Shortest Remaining Time):** SRT dynamically selects the process with the shortest remaining execution time to run next. This scheduling algorithm aims to minimize the turnaround time by preempting the currently running process when a shorter one becomes available. However, SRT may suffer from frequent context switches and overhead due to its aggressive preemption strategy.

**2. Average Waiting Time Comparison:**

* **FCFS (First-Come, First-Served):** FCFS scheduling tends to have higher average waiting times, especially in scenarios with a mix of short and long processes. Long processes occupying the CPU for extended periods can cause shorter processes to wait excessively, resulting in increased average waiting times.
* **RR (Round-Robin):** RR scheduling aims to provide fair access to the CPU by allocating time slices to each process. While RR can prevent starvation and ensure all processes receive some CPU time, it may lead to higher average waiting times due to frequent context switches and overhead associated with time slicing.
* **SPN (Shortest Process Next):** SPN prioritizes shorter processes, allowing them to execute sooner and reducing overall waiting times. By selecting the shortest process available for execution, SPN minimizes waiting times and improves system responsiveness. However, longer processes may experience increased waiting times or starvation.
* **SRT (Shortest Remaining Time):** SRT aggressively schedules processes with the shortest remaining execution time, minimizing waiting times for short tasks. However, it may increase waiting times for longer processes due to frequent preemptions, leading to potential performance degradation.

**3. Throughput Comparison:**

* **FCFS (First-Come, First-Served):** FCFS scheduling may exhibit moderate throughput, particularly in scenarios with a balanced mix of short and long processes. However, it may suffer from reduced throughput in cases where long processes dominate the CPU, leading to increased turnaround times and lower overall efficiency.
* **RR (Round-Robin):** RR scheduling offers reasonable throughput by ensuring fair access to the CPU for all processes. The fixed time quantum allows for predictable scheduling and prevents individual processes from monopolizing resources. However, RR may experience decreased throughput when handling a large number of short processes, as frequent context switches can introduce overhead.
* **SPN (Shortest Process Next):** SPN scheduling can achieve high throughput by prioritizing short processes for execution. By selecting the shortest available process, SPN minimizes waiting times and expedites job completion. However, SPN may struggle with longer processes, leading to potential starvation and decreased overall system throughput.
* **SRT (Shortest Remaining Time):** SRT aims to maximize throughput by favoring short tasks. However, its aggressive preemption strategy may introduce overhead and reduce overall system throughput, especially in scenarios with a mix of short and long processes.

**Conclusion:**

In conclusion, each scheduling algorithm offers unique advantages and trade-offs in terms of average turnaround time, average waiting time, throughput, and responsiveness. While FCFS prioritizes fairness and simplicity, RR ensures fairness through time slicing, SPN prioritizes efficiency by selecting the shortest processes, and SRT aggressively minimizes waiting times for short tasks. The choice of scheduling algorithm depends on system requirements, workload characteristics, and desired performance objectives.