**Advanced Operating System – Project Final Report**

Scheduling algorithms are crucial in operating systems for managing processes and optimizing resource utilization. In this report, we will analyze four different scheduling algorithms based on their performance metrics: Average Waiting Time and Average Turnaround Time. The algorithms considered are:

1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR) with Quantum 6ms
4. Shortest Remaining Time First (SRTF)
5. Priority Scheduling

**Output:**

**A black screen with white text

Description automatically generated**

1. **First Come First Serve (FCFS)**

**Pros:**

1. Simple to implement.
2. Easy to understand and manage.

**Cons:**

1. Can lead to the "convoy effect," where short processes get stuck waiting behind long processes.
2. Results in high average waiting time and turnaround time, especially with large numbers of processes or varying process lengths.
3. **Shortest Job First (SJF)**

**Pros:**

1. Minimizes average waiting time and turnaround time compared to FCFS.
2. Efficient for batch processing systems where process lengths are known in advance.

**Cons:**

* 1. Requires knowledge of the burst time of processes, which may not be available.
  2. Can cause starvation of longer processes if short processes continue to arrive.

1. **Round Robin (RR) with Quantum 6ms**

**Pros:**

1. Fairly allocates CPU time among processes.
2. Good for time-sharing systems where processes are interactive and require frequent CPU access.

**Cons:**

1. Average waiting time and turnaround time are higher compared to SJF and SRTF due to context switching.
2. Quantum size affects performance; too small or too large quantum can lead to inefficiencies.
3. **Shortest Remaining Time First (SRTF)**

**Pros:**

1. Provides the lowest average waiting time and turnaround time among the algorithms tested.
2. Dynamically adjusts to the shortest remaining process time, improving overall efficiency.

**Cons:**

1. Requires knowledge of remaining burst times, which may not be feasible.
2. Can lead to starvation for longer processes if short processes keep arriving.
3. **Priority Scheduling**

**Pros:**

1. Processes with higher priority get executed first, which can be useful in certain scenarios.
2. Can be customized with different priority schemes (e.g., static, dynamic).

**Cons:**

1. Can lead to high average waiting time and turnaround time if not managed well.
2. Risk of starvation for low-priority processes if high-priority processes keep arriving.

**Code Snippets:**

1. **First Come First Serve**

**A screen shot of a computer program

Description automatically generated**

1. **Shortest Job First**

**A screen shot of a computer program

Description automatically generated**

1. **Round Robin**

**A screen shot of a computer program

Description automatically generated**

1. **Shortest Remaining Time First**

**A screen shot of a computer program

Description automatically generated**

1. **Priority Scheduling**

**A screen shot of a computer program

Description automatically generated**