**Overview**

This project implements and compares two fundamental CPU scheduling strategies:

* **First-Come, First-Served (FCFS)**
* **Shortest Job First (SJF)**

A dataset of six processes is used, where each algorithm is assessed based on:

* Waiting Time (WT)
* Turnaround Time (TAT)
* Average Waiting Time (AWT)
* Average Turnaround Time (ATAT)

**Implementation Highlights**

The project is developed in C using a structured, modular approach. The data is loaded from a text file and the scheduling logic is implemented in dedicated functions. Key aspects include:

* **Process Representation**: Each process stores arrival time, burst time, and status-related metrics.
* **FCFS Logic**: Sorts by arrival time and runs each process in order.
* **SJF Logic**: Chooses the process with the shortest burst time from all processes that have arrived.
* **Unified Output Function**: Generates a Gantt chart, individual metrics, and overall averages.

**Sample Input File: processes2.txt**

PID Arrival\_Time Burst\_Time Priority

1 0 5 2

2 2 3 1

3 4 2 3

4 6 4 2

5 8 1 1

6 10 2 2

**Execution Results**

**FCFS Output**

**Sequence:** P1 → P2 → P3 → P4 → P5 → P6  
**Timeline:**

| P1 | P2 | P3 | P4 | P5 | P6 |

0 5 8 10 14 15 17

|  |  |  |
| --- | --- | --- |
| **PID** | **WT** | **TAT** |
| 1 | 0 | 5 |
| 2 | 3 | 6 |
| 3 | 4 | 6 |
| 4 | 4 | 8 |
| 5 | 6 | 7 |
| 6 | 5 | 7 |

**Avg WT:** 3.67  
**Avg TAT:** 6.5

**SJF Output**

**Sequence:** P1 → P3 → P2 → P5 → P6 → P4  
**Timeline:**

| P1 | P3 | P2 | P5 | P6 | P4 |

0 5 7 10 11 13 17

|  |  |  |
| --- | --- | --- |
| PID | WT | TAT |
| 1 | 0 | 5 |
| 3 | 1 | 3 |
| 2 | 5 | 8 |
| 5 | 2 | 3 |
| 6 | 3 | 5 |
| 4 | 7 | 11 |

**Avg WT:** 3.0  
**Avg TAT:** 5.83

**Comparison Summary**

|  |  |  |
| --- | --- | --- |
| Metric | FCFS | SJF |
| Avg WT | 3.67 | 3.0 |
| Avg TAT | 6.5 | 5.83 |

SJF provides lower waiting and turnaround times, making it more efficient in this case.

**Development Challenges and Solutions**

* **Dynamic Selection in SJF**: Required conditionally evaluating available processes at each time step. Resolved using runtime checks.
* **Handling CPU Idle Time**: Managed by advancing time when no process is ready.
* **Maintaining Input Consistency**: Reloaded process data before running each algorithm to ensure fairness.