Technical Documentation

1. Implementation Overview

This project simulates and compares four CPU scheduling algorithms such as First-Come First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling (PS), and Round Robin (RR) using process sets generated with random and uniform distributions. The simulation is implemented in the C programming language and focuses on evaluating key performance metrics as follows:

- Average Waiting Time (AWT)
- Average Turnaround Time (ATAT)
- Average Response Time (ART)
- Waiting Time Variance Deviation (WTVD)

2. Code Structure

The given code is flexible and organized into the following key components:

main.c

This is the entry part of the program and it handles user input and coordinates process generation and scheduling execution. The program fulfils very user-friendly approach by giving options to select the distribution type for process generation and the number of processes to generate.

process_generator.c

Contains functions to generate processes with attributes such as arrival time, burst time, and priority using Random and Uniform distributions.

scheduler_fcfs.c / .h

Implements the First-Come-First-Serve algorithm.

scheduler_sjf.c / .h

Implements the Shortest Job First scheduling (non-preemptive and/or preemptive versions).

scheduler_priority.c / .h

Implements Priority Scheduling (with and without preemption).

scheduler_rr.c / .h

Implements Round Robin Scheduling with configurable time quantum.

metrics.c / .h

Calculates and stores the performance metrics for analysis and comparison.

utils.c / .h

Contains helper functions for sorting, printing Gantt charts, and logging results.

3. Instructions for Execution

Requirements:

- GCC compiler
- Linux or Windows with a terminal or console environment

To Compile:

gcc -o dr.exe dr.c

To Run:

.\dr.exe

Program Flow:

1. User selects the number of processes to be generated (max 10) and process generation method (random or uniform).
2. The generator creates a set of specified number of processes (max 10) with varying attributes.
3. All 4 selected CPU scheduling algorithm is applied to the process set.
4. Performance metrics are calculated and printed (C implementation).
5. Gantt chart and comparison data are optionally visualized or saved (python implementation).