Project 4: Scheduler

Course: CECS 326 SEC 02 - Operating Systems
Project: Interprocess Communication
Team Members:

- Teammate 1: Phu Quach (ID: 029475548)
- Teammate 2: Ryan Tran (ID: 031190716)

Date: December 2, 2025

I . Summary

This project implements three classic CPU scheduling algorithms: First-Come, First-Served (FCFS), Priority Scheduling and Round Robin (RR).

See the demo here: https://youtu.be/JScJUH1y37w
See the repo here: https://github.com/harvest7777/proj4

II. Design Architecture

A. Objective

The primary objective was to write a C program that copies src to dst files using pipes for inter-process communication so we don't store the whole contents in memory all at once.

B. System Architecture and Program Workflow

1. Driver is the entry point for every algorithm
    a. Reads a file (e.g., schedule.txt) line by line
    b. Parses each line: name, priority, burst
    c. Calls add() to add each task to a linked list
    d. Calls schedule() to run the scheduler
2. The different scheduling algorithms (
    a. FCFS (First-Come-First-Served) - schedule_fcfs.c
        i. Runs tasks in the order they were added
        ii. Each task runs to completion before the next starts
        iii. Simple queue behavior
    b. Priority Scheduling - schedule_priority.cw
        i. Runs the highest priority task first
        ii. Higher numeric priority = runs first
        iii. Each task runs to completion before selecting the next highest priority
    c. Round Robin - schedule_rr.c

        i. Gives each task a time slice (quantum = 10 units)
        ii. Cycles through tasks in order
        iii. If a task isn't done after its slice, it goes back to the queue
        iv. Continues until all tasks finish
3. CPU sim
    a. Fake simulator to replicate actually running a task for some amount of time
4. Output
    a. Each scheduler prints…
        i. Which task runs and for how long
        ii. Wait time, turnaround time (and response time for Round Robin)
        iii. Summary statistics (averages)

C. Implementation

a. FCFS: For first come first serve, we have to start by reversing the list to get the correct task order. It's a linked list data structure so the solution to reverse is the classic reverse linked list algorithm with prev and next pointers. After the list is reversed and we have the correct ordering of nodes (tasks), we simply iterate through the linked list by updating our pointer then running and printing results from the run task.
b. Priority scheduling: For priority scheduling, we define a helper mynexttask() that returns a pointer to the next task. This simply iterates through the linked list and returns the task with the highest priority. After getting the task with the highest priority, we run the task then remove it from the list.
c. Round robin: For round robin, we use the defined quantum of 10. As per the instructions, we have a task that takes a non multiple of 10 amount of time to complete so we can see the round robin algorithm in full effect. It iterates through the linked list and makes QUANTUM (10) amount of progress on each task. If the task can be completed within that quantum we remove it from the list. We keep iterating through tasks in fifo order until they're all finished.

III. Contribution

1. **Phu:**

- Primary responsibility for the **core technical architecture** and **implementation** aspects of the project. He designed the overall system architecture, determining how to structure the parent-child process model and establish the pipe-based communication system.
- Implemented all the functions files such as **schedule_rr.c, schedule_fcfs.c,** and **schedule_pri.c**.

2. **Ryan:**

- **Extensively tested program**, to make sure outputs were sensible
- Documented code for **project report**

- Verbal explanation and code showcase in **video**

IV. Conclusion

We learned how to implement and simulate fcfs, priority, and round robin scheduling in C. We did not use real tasks, but rather, a pre defined function that pretends to run tasks and mimics the task data structure.

V. References

FCFS scheduling

References:
1. Lecture Slide: CPU Scheduling From Professor slide number 13 and 14
2. FCFS Definition
:https://www.geeksforgeeks.org/dsa/first-come-first-serve-cpu-scheduling-non-preemptive/
3. Implementation :https://www.scaler.com/topics/first-come-first-serve
4. Formula to calculate average waiting time and turnaround time :
https://www.geeksforgeeks.org/operating-systems/difference-between-turn-around-time-tat-and-waiting-time-wt-in-cpu-scheduling/
5. Memory allocation and deallocation in C :
https://www.geeksforgeeks.org/c/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/
6.Free Memory in C : https://stackoverflow.com/questions/9069205/how-do-i-free-memory-in-c

Note for project from professor description:
-It is assumed that all tasks arrive at the same time, so your scheduler algorithms do not have to support
higher-priority processes preempting processes with lower priorities.

Priority scheduling Algorithm

References:
1. Lecture Slide: CPU Scheduling From Professor slide number 20 and 21
2. Scheduling Algorithms - Priority Scheduling :
https://www.youtube.com/watch?v=yKD3pcFvGmY
3. Program for Priority CPU:
https://www.geeksforgeeks.org/operating-systems/program-for-priority-cpu-scheduling-set-1/
4. Formula to calculate average waiting time and turnaround time :
https://www.geeksforgeeks.org/operating-systems/difference-between-turn-around-time-tat-and-waiting-time-wt-in-cpu-scheduling/
5. Free Memory in C : https://stackoverflow.com/questions/9069205/how-do-i-free-memory-in-c

Note for implementation from professor description:

-“Priorities range from 1 to 10, where a higher numeric value indicates a higher relative priority.”

Round-robin scheduling

References:
1.Lecture Slide: CPU Scheduling From Professor slide number 22 to 25
2. Coding Implementation Reference :
https://www.geeksforgeeks.org/operating-systems/program-for-round-robin-scheduling-for-the-same-arrival-time/
3. Free Memory in C : https://stackoverflow.com/questions/9069205/how-do-i-free-memory-in-c
4. Formula to calculate average waiting time and turnaround time :
https://www.geeksforgeeks.org/operating-systems/difference-between-turn-around-time-tat-and-waiting-time-wt-in-cpu-scheduling/

Note for implementation from professor description: :
- For round-robin scheduling, the length of a time quantum is 10 milliseconds