Lab Assignment 2 (40 points)

Building a CPU scheduler

Due by 2 PM, October 22 (Wednesday), 2025

1. Goal of this programming assignment

The primary goal of this lab assignment is to understand various CPU scheduling algorithms and gain experience building a simulator for CPU scheduling algorithms to evaluate their performance.

2. Lab assignment description

Write a C/C++ program to simulate a CPU scheduler that selects a processor from a ready queue and executes the process by a given scheduling algorithm, displays its activities, and evaluates its performance based on collected measurements such as average turnaround time, average waiting time, and average response time. When a process is scheduled, the simulator prints the process name, the time it is running, and collects measurement data, producing Gantt Chart-like outputs.

2.1 Specification

You are asking to implement the **FCFS** (First Come First Service), **RR** (Round Robin), and **SRTF** (Shortest Remaining Time First) scheduling algorithms. The detailed algorithms are well-described in Chapter 5 of the textbook. To get **full credit**, you should implement **FCSF**, **RR, and SRTF algorithms.**

- **YOU MUST CREATE PCB Structures**
- **DO NOT USE ANY STANARD (STD or STL) LIBRARY for building a queue and managing the ready queue. For example, you cannot use STD/STL for a queue or linked list. YOU HAVE TO BUILD YOUR OWN QUEUE AND LINKED LIST!!**

2.2 Assumptions

Use the following assumptions when designing and implementing a CPU simulator.

- There is only one CPU
- All processes perform only CPU operations.
- All processes have the same priority.
- The new arrival process is directly stored in the ready queue.
- We use only the ready queue for this simulation.

- **We take into consideration the costs of context switching. We assume the context switching cost is 0.5 milliseconds when a context switch occurs.**
- **The context switch is only performed when a current process is moved from the running queue to the ready queue. Please note that the process termination does not trigger context switching.**
- The time of the unit is milliseconds.
- We use the FCFS policy for breaking ties. (For Round-Robin and SRTF case)

2.3 Measurements and Evolution

You should collect the following information about each process:

- Time of completion
- Waiting time
- Turnaround time
- Response time
- No. of Context occurred

You should calculate the following information using the collected measurements:

- Average CPU burst time
- Average waiting time
- Average turnaround time
- Average response time
- Total number of Context Switches performed

Please refer to Chapter 5 in the textbook and lecture notes for detailed information on the measurements, including waiting time, turnaround time, response time, average waiting time, average turnaround time, and average response time.

2.4 Simulator Input

Three inputs will be given as arguments when the simulator begins.

1. Process arrival information (the total number of arrival processes is up to 100)

   The process information will be read from a text input file. The data for each process will include the following fields:

   Pid*:* a unique numeric process ID.
   Arrival time: the time when the task arrives in the unit of milliseconds
   CPU *burst time:* the CPU time requested by a process

An example of an input file:

```
1    0     10
2    1      2
3    2      9
4    3      5
```

Note: The time unit *for arrival and CPU burst times* is milliseconds. You can assume that all time values are integers and that the *pids* provided in an input file are unique.

2. Type of scheduling algorithm

You will implement four scheduling algorithms and select one algorithm when you start the simulator as the 3rd argument. You will enter an integer value from one of the following:

- FCFS: unixprompt> myscheduler test-input-file-name output-file-name 0
- RR: unixprompt> myscheduler test-input-file-name output-file-name 1 quantum-size
- SRTF: unixprompt> myscheduler test-input-file-name output-file-name 2

3. Time Quantum size

You will define the time quantum when the simulator begins as the first argument. The *time quantum* is essential for RR scheduling, and the time quantum value must be an integer greater than 0. Thus, your program should be able to take the time quantum value as the first input value. So, your program will be executed as follows:

unixprompt> myscheduler test-input-file-name output-file-name 2 4

2.5 Output (Sample): The simulator will display the simulation results as follows:

```
**********************************************************************
********** Scheduling algorithm: FCFS                    **********
**********************************************************************
```

pid	arrival	CPU-burst	finish	waiting time	turn around	response time	No. of Context
1	0	10	10.0	0.0	10.0	0.0	0
2	1	2	12.0	9.0	11.0	9.0	0
3	2	9	21.0	10.0	19.0	10.0	0
4	3	5	26.0	18.0	23.0	18.0	0

Average CPU burst time = 6.50 ms, Average waiting time = 9.25 ms
Average turn around time = 15.75 ms, Average response time = 9.25 ms
Total No. of Context Switching Performed =0

****
********* Scheduling algorithm: Round Robin **********
********* (No. of Task = 4 Quantum= 4) **********
****

pid	arrival	CPU-burst	finish	waiting time	turn around	response time	No. of Context
1	0	10	27.5	17.5	27.5	0.0	2
2	1	2	6.5	3.5	5.5	3.5	0
3	2	9	28.5	17.5	26.5	4.5	2
4	3	5	25.5	17.5	22.5	8.0	1

Average CPU burst time = 6.50 ms, Average waiting time = 14.0 ms
Average turn around time = 20.5 ms, Average response time = 4.00 ms
Total No. of Context Switching Performed = 5

****
********** Scheduling algorithm : SRTF **********
****

pid	arrival	CPU-burst	finish	waiting time	turn around	response time	No. of Context
1	0	10	17.5	7.5	17.5	0.0	1
2	1	2	3.5	0.5	2.5	0.5	0
3	2	9	26.5	15.5	24.5	15.5	0
4	3	5	8.5	0.5	5.5	0.5	0

Average CPU burst time = 6.50 ms, Average waiting time = 6.0 ms
Average turn-around time = 12.5 ms, Average response time = 4.125 ms
Total No. of Context Switching Performed =1

2.6 Testing Requirements

You should test all 3 of your algorithms with **three (3) different types of input arrival information files**. Especially when you test your RR, you should also use **three (3) different time quantum values (e.g., 4, 16, 64)** for each input arrival information file. **These tests should be submitted as your sample output.**

3. Programming Requirements

(1) Programming language: You have to use either C or C++ to develop your simulator on a Linux environment
(2) Running Environment: Your program should be compiled on the CSEGrid server and be able to run without errors.

(3) **DO NOT USE any built-in queue library or class (do not use STL library in C++ or C). You have to implement your own queue management functions (or methods).**

4. Deliverables

(1) Program source codes include all source program files, Makefile, and Readme
(2) Sample output (hard copy)

5. Evaluation criteria

(1) Deliverables (3 points)
- **i.** Submitting all required deliverables — 2 points
- **ii.** Sample Output (hard copy) — 1 point

(2) Completeness (35 points)
- (2.1) Program Structure
  - (a) Design PCB — 2 points
  - (b) Own Queue Management — 3 points
- (2.2) Implementing scheduling algorithms (including correctness)
  - (a) FF — 5 points
  - (b) RR — 12 points
  - (c) SRTF — 13 points

(3) Testing and output (2 points)
- **i.** Submitting output for all required testing results and display correctly all the necessary information — 2 points

6. How to turn in your work

Please do the following when you submit your programming assignment.

- Create a tar file that contains your written source code, makefile, and readme. DO NOT INCLUDE EXECUTABLES AND OBJECT FILES.
  - Please use the following convention when you create a tar file
    - First 3 letters of your last name + last 4 digits of your student ID
    - For example, if a student's name is "Bill Clinton" and their ID is 999-34-5678, then their tar file name is "cli5678.tar".
    - e.g.) tar –cf cli5678.tar main.cpp command.cpp Makefile readme
    - If you want to know more about the "tar" command, type "man tar" at the Unix prompt.

- Then, compress the created tar file with the "gzip" command.
  - (e.g.) gzip cli5678.tar
- **Upload your zipped tar file(e.g., cli5678.tar.gz) to the class Canvas website**.