Lab Assignment 2 (40 points)

<u>Building a CPU scheduler</u>

Due by 2 PM, March 25 (Monday), 2024

1. The goal of this programming assignment

This lab assignment's primary goal is to understand various CPU scheduling algorithms and gain experience building a simulator for CPU scheduling algorithm to evaluate their performances.

2. Lab assignment description

Write a C/C++ program to simulate a CPU scheduler that selects a process from a ready queue and executes the process by a given scheduling algorithm. Your simulation displays its activities and evaluates its performance based on measurements such as average turn-around time, waiting time, and response time.

2.1 Specification

You are asking to implement **FCFS** (First Come First Serve), **SRTF** (Shortest Remaining Task First), and **RR** (Round Robin) scheduling algorithms. The detailed algorithms are well described in Chapter 5 of the textbook. Implementing FCSF, SRTF, and RR algorithms to receive full credit would be best**.**

- **YOU <u>MUST CREAT</u> PCB Structures**
- **DO <u>NOT USE ANY STANARD (STD or STL) LIBRARY</u> to build and manage 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 you design and implement a CPU simulator.

- There is only one CPU
- All processes perform only CPU operations.
- The new arrival process is directly stored in the ready queue.
- We use only the ready queue for this simulation.
- **We take into consideration context switching costs. We assume the context switching cost is 0.5 milliseconds when the context switching occurs.**

- **The context switch is only performed when a current process is moved to the ready queue. The context switch does not happen if the current process is terminated, and DO NOT ADD the contexted switch overhead.**
- **In Round-Robin, a new arrival process should enter into the ready queue before the reentered process if the new arrival process arrives simultaneously (context switch time + the preempted time of the current process).**
- The time of the unit is milliseconds.
- We use the FCFS policy to break the tie. (For Round-Robin case)

2.3 Measurements and Evaluation

You should collect the following information about each process:

- Time of completion
- Waiting time
- Turn around time
- Response time
- No. of Context occurred

You should calculate the following information using collected measurements:

- Average CPU burst time
- Average waiting time
- Average turn-around time
- Average response time
- Total number of Context Switching performed

Please refer to Chapter 5 in the textbook and lecture notes for all detailed information about the measurements: waiting time, turn around time, response time, average waiting time, average turn around time, and average response time.

2.4 Simulator Input

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

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

The process information will be read from a text input file. The information 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
Priority: assigned priority of arrival process (0-99, 0-highest, 99-lowest)

An example of an input file:

```
1   0   10  20
2   1    2  10
3   2    9   1
4   3    5   5
```

Note: The time unit *for arrival and CPU burst times* is a millisecond. 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 three scheduling algorithms and select one algorithm when you start the simulator as the 2nd argument. You will enter an integer value from one of the following:

- FCFS: unixprompt> myscheduler test_input_file 0
- SRTF: unixprompt> myscheduler test_input_file 1
- RR: unixprompt> myscheduler test_input_file 2 quantum-size

3. Time Quantum size

You will define the time quantum when the simulator begins as the first argument. The *time quantum* is essential for the RR scheduling, and the *time quantum* value is an integer value and should be 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 2 4
```

2.5 Output (Sample)

The simulator will display both histories of the process, like Gantt Chart-like outputs and measurements. The anticipated output format is as follows:

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

pid	arrival	CPU-burst	Priority	finish	waiting time	turn around	response time	No. of Context
1	0	10	20	10.0	0.0	10.0	0.0	0
2	1	2	10	12.0	9.0	11.0	9.0	0
3	2	9	1	21.0	10.0	19.0	10.0	0
4	3	5	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: 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
```

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

pid	arrival	CPU-burst	Priority	finish	waiting time	turn around	response time	No. of Context
1	0	10	20	27..5	17.5	27.5	0.0	2
2	1	2	10	6.5	3.5	5.5	3.5	0
3	2	9	5	28.5	17.5	26.5	4.5	2
4	3	5	0	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

2.6 testing requirements

You should test all your three algorithms. Especially when you test your RR, you should test for **three (3) different time quantum values (e.g., 4, 16, 64)** for each input arrival information file. **<u>These tests should be submitted as your sample output</u>.**

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 at the **csegrid server** and be able to be tested without errors.
(3) **DO NOT USE any built-in queue library or class (<u>do not use the STL library in C++ or C</u>). 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 rubrics

(1) Deliverables (3 points)

i.	Submitting all required deliverables	2 points
ii.	Sample Output (hard copy)	1 point

(2) Completeness (35 points)

i. Program structures (5 points)

(a)	Define PCB structure	2 points
(b)	Quality of building ready queue	3 points

ii. Implementing scheduling algorithms (including correctness)

(a)	FF	5 points
(b)	SRTF	10 points
(c)	RR	10 points

(3) Testing and Output (2 points)

i. Submitting Output for all required testing results and display correctly all required 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. <u>DO NOT INCLUDE EXECUTABLES AND OBJECT FILES.</u>
    - Please use the following convention when you create a tar file
        - First three letters of your last name + the last four digits of your student ID
        - For example, if a student's name is "Bill Clinton" and his ID is 999-34-5678, then his 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.

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