Lab Assignment 2 (40 points)

Building a CPU scheduler

Due by 2 PM, October 29 (Monday), 2018

1. Goal of this programming assignment

The primary goal of this lab assignment is to understand various CPU scheduling algorithms and gain some 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 which selects a processor from a ready queue and executes the process by given scheduling algorithm, display its actives and evaluate its performance based on collected some measurements such as average turn-around time, average waiting time, average response time.

2.1 Specification

You are asking to implement **FCFS** (First Come First Serve), **SJF** (Shortest Jobs First), **SRTF** (Shortest Remaining Task First), **RR** (Round Robin) scheduling algorithms. The detailed algorithms are well described Chapter 5 in textbook. In order to **full credit**, you should Implementing **FCSF**, **SJF**, **SRTF, and RR algorithms.**

- **YOU MUST CREAT A PCB Structures**
- **DO NOT USE ANY STANARD (STD or STL) LIBRIRY for building queue and manage ready queue. For example, you cannot use STD/STL for 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.
- All processes have the same priority.
- The new arrival process is directly stored at the ready queue.
- We use only ready queue for this simulation.
- **We take into consideration of context switching cost. We assume the context switching cost is 0.5 milliseconds when a context switching is occurred.**

- **Context switch is performed only when a current process is moved to the ready queue.**
- **In Round-Robin, a newly arrival process should enter into the ready queue before the reentered process if the arrival time of newly arrival process is the same as the arrival time of reentered process (context switch time + the preempted time of current process).**
- The time of unit is milliseconds.
- We use FCFS policy for breaking the tie. (For Round-Robin case)

2.3 Measurements and evolution

You should collect the following information of 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 Chapter 5 in textbook and lecture notes for all detail 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

There are three inputs will be given as arguments when the simulator begins.

1. <u>Process arrival information</u> (expect total number of arrival processes are up to 100)

The process information will be read from <u>a text input file</u>. 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

An example of input file:

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

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

2. Type of scheduling algorithm

You will be implementing two scheduling algorithms, and you will select one algorithm when you start the simulator as the 2nd argument. You will enter an integer value from one of the followings:

- FCFS: unixprompt> myscheduler test_input_file 0
- SJF: unixprompt> myscheduler test_input_file 1
- SRTF: unixprompt> myscheduler test_input_file 2
- RR: unixprompt> myscheduler test_input_file 3 quantumsize

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 3 4
```

2.5 Output (Sample)

The simulator will display both history of process, like Gantt Chart-like outputs, and measurements. The expected output format is 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 : SJF   **********************
****

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	26.0	15.0	24.0	15.0	0
4	3	5	17.0	9.0	14.0	9.0	0

Average CPU burst time = 6.50 ms,   Average waiting time = 8.25 ms
Average turn-around time = 14.75 ms,   Average response time = 8.25 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	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

2.6 testing requirements

You should test your all 4 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. **<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 Linux environment
(2) Running Environment: Your program should be compiled at **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 stl library in C++ or C</u>). You have to implement your own queue management functions (or method).**

4. Deliverables

(1) Program source codes includes 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)

i. Program structures (5 points)

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

ii. Implementing scheduling algorithms (include correctness)

(a)	FF	5 points
(b)	SJF	7 points
(c)	SRTF	8 points
(d)	RR	10 points

(3) Testing and Output (2 points)

i. Submitting output for required all testing results and display properly all required information 2 points

6. How to turn in your work

Please do the followings 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 3 letters of your last name + last 4 digits of your student ID
        - e.g.: If a student 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 "tar" command, type "man tar" at Unix prompt.

- Then compressed the created tar file using "gzip" command.
    - (e.g) gzip cli5678.tar
- **Upload your zipped tar file(e.g. cli5678.tar.gz) to class Canvas web-site**.