CEG 4350 Operating Systems

Project Assignment

Designing and Implementing a CPU Scheduling Simulator

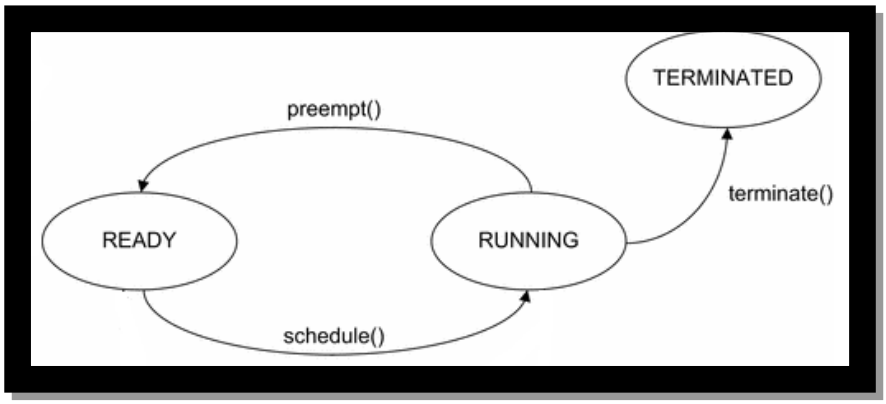
Due on July 15, 2018 (11:30pm)

(For the project, you can work with one partner in class.)

Objectives: The purpose of this project is to give you a chance to implement/simulate a few typical CPU scheduling policies discussed in the class. You will write a C/C++ program to implement a simulator with different scheduling algorithms, i.e., the simulator selects a task to run from ready queue(s) based on the scheduling algorithm selected. Since the project intends to simulate a CPU scheduler, so it does not require any actual process creation or execution. When a task is scheduled, the simulator will simply print out what task is selected to run at a time. It outputs the way similar to Gantt chart style.

Process States: In this CPU scheduling simulation, there are 3 possible states for a process:

- READY - The process is ready to execute, and is waiting to be scheduled on a CPU.
- RUNNING - The process is currently executing on a CPU.
- TERMINATED - The process has completed.



Scheduling Algorithms: For your simulator, you will implement the following three CPU scheduling algorithms:

- *First-Come, First Served (FCFS)* - Runnable processes are kept in a first-in, first-out ready queue. FCFS is non-preemptive; once a process begins running on a CPU, it will continue running until it either completes or blocks for I/O.
- *Round-Robin* - Each process is assigned a timeslice when it is scheduled. At the end of the timeslice, if the process is still running, the process is preempted, and moved to the tail of the ready queue. Use timeslice (i.e., quantum) = 8 ms.
- *Multi-Level Feedback Queue* scheduling algorithm:
    - A new process first enters Queue 0 which is served in RR. When it gains CPU, process receives 8 ms. If it does not finish in 8 ms, process is preempted and moved to Queue 1.
    - At Queue 1, process is again served in RR and receives 16 additional ms. If it still does not complete, it is preempted and moved to Queue 2.
    - At Queue 2, process is served in FCFS.
    - When a running process is preempted by a new process arriving at a higher priority queue, it will be put back to the end of its current queue. i.e., its priority keeps the same. A new full quantum will be given to it the next time it starts running.

Input (Process Information): The process information will be read from an input file. The format is *pid arrival_time burst_time* . All of fields are integer type where *pid* is a unique numeric process ID, *arrival_time* is the time when the task arrives in the unit of milliseconds, and *burst_time* the is the CPU time requested by a task, in the unit of milliseconds The time unit for *arrival_time, burst_time* is millisecond.

Command-line Usage Examples:

```
myCPUScheduler input_file [FCFS|RR|MLFQ]
```

Input file example:

```
% more input.txt
PID    Arrival  Burst
1      0        10
2      1        9
3      2        8
4      3        7
```

Output:

1. When a task is scheduled, the simulator will simply print out what task is selected to run at a time. Show the status of related process at each scheduling event, e.g.,: *starts running, has finished, is preempted into Queue #, or continues running*.
2. Print statistical information. As soon as all tasks are completed, the program should compute and print 1) average waiting time, 2) average response time, 3) average turnaround time.

   Example:

```
%myCPUScheduler input_file FCFS

Selected Scheduling algorithm: FCFS
PID    starts running      at     TIME
PID    has finished  at    TIME
PID    starts running      at     TIME
PID    has finished  at    TIME
…

================================
Average waiting time:     10.00 ms
Average response time: 3.00 ms
Average turnaround time: 13.00 ms
================================
```

What to submit in Pilot Dropbox:

1. The project in one ZIP file, including a readme file and a make file.
2. Copy all your source codes into a single text document and submitted for plagiarism check.
3. A project report on your design and implementation, problems unsolved if any, and etc. Typically, 2 pages.