CS 3500 – Operating Systems; July 2018 semester
Krishna Sivalingam
Lab 8: Multi-level CPU Scheduling with Feedback
Due Date: Sunday, **Oct. 14, 11.00PM**; via Moodle
NO LATE SUBMISSIONS

The objective of this project is to implement commonly used scheduling algorithms (FCFS, SJF, RR) as discussed in class, in a system that supports multi-level queues with feedback.

1 Requirements

The system has four levels: Level-1 is highest priority and Level-4 has lowest priority. The CPU schedules from Level-2 only if there are no processes in Level-1; from Level-3 only if there are no processes in Level-1 and in Level-2; and so on.

Level-1 uses RR; Level-2 uses SJF; Level-3 uses SJF; Level-4 uses FCFS. All algorithms will be in non-preemptive mode (including SJF); A process under RR will execute under the specified time quantum runs out or it completes its burst.

After spending some specified threshold time in the system, the system moves to the immediate higher level. Processes in Level-1 Queue cannot move to a higher level.

Here is a brief specification; you can make additional assumptions as needed. All units are taken to be in milli-seconds.

The command-line arguments are:

- ▷ -Q: time quantum for Robin-robin scheme (a number between 10 and 20)
- ▷ -T: time threshold for moving a process to the next level (a number between 100 and 50000)
- ▷ -F: input filename
- ▷ -P: output filename
- ▷ any other inputs that you wish.

The input file (specified by the command line), has a set of entries, one entry per line, as shown below:

```
1 3 100
2 4 70
3 2 35
4 1 40
...
```

The specified fields are, in order: ID, Initial Queue Level, CPU Burst time (ms). Assume that each process has only one burst to run on the CPU.

What to Measure: For each process, measure the time between completion time and the time of addition to the first queue it was assigned to. For the system, measure the throughput (no. of processes completed per unit time).

What to Output: For each process, upon its completion, print the following in the output file:

```
ID: xy; Orig. Level: ab; Final Level: cd; Comp. Time(ms): ;TAT (ms):
```

At the end of running all processes specified in the file, print the following information in the output file:

```
Mean Turnaround time:  abc.de (ms); Throughput: xyz.gh processes/sec
```

2 Miscellaneous

1. WARNING ABOUT ACADEMIC DISHONESTY: Do not share or discuss your work with anyone else. The work YOU submit SHOULD be the result of YOUR efforts. The academic conduct code violation policy and penalties, as discussed in the class, will be applied.
2. You can use C/C++ STL or any available Java package for implementing Queues, Heaps, etc.
3. You can get *gettimeofday()* function to determine when a process was added to a given queue.
4. You can use *nanosleep* function to emulate a process' execution for its specified burst duration (given in milli-seconds).
5. If your system crashes often because of this, please don't blame the instructor or TAs.
6. If you find this assignment to be not as challenging and prefer to implement this in a real kernel, please talk to the instructor.