COMP3520 Operating Systems Internals Assignment 3: Stage 1 - Multi-Level Queue Dispatcher

General Instructions

This is Stage 1 of Assignment 3. It is about multi-level queue dispatcher and consists of two compulsory tasks:

1. **Revise Programming Exercise 3, or Assignment 2 program** and implement a Multi-Level Queue Dispatcher for a uniprocessor system (**70%**); and
2. Answer the discussion document questions (**30%**).

This assignment is an individual assignment. Whilst you are permitted to discuss this assignment with other students, the work that you submit must be your own work. You should not incorporate the work of other students (past or present) into your source code or discussion document.

Your work for Stage 1 will be marked (30% of the total marks for Assignment 3). You are required to submit your **source code**, **job dispatch list files**, **makefile**, and **discussion document** together in a **zip/tar file** to the submission inbox in the COMP3520 Canvas website.

Multi-Level Queue Dispatcher

In Stage 1 of Assignment 3, you will have a list of randomly generated jobs that you need to schedule for execution depending on the job's time of arrival, its allowed execution time and priority. Unlike Programming Exercise 3/Assignment 2, there are two types of jobs, i.e., **real-time** jobs, and **normal** jobs. Real-time jobs are given a priority value 0 and normal jobs are initially given a priority value 1. The larger the priority value, the lower is the priority. The job dispatch list format is as follows:

<arrival time>, <cputime>,<priority>

0, 3, 1

2, 6, 0

4, 4, 1

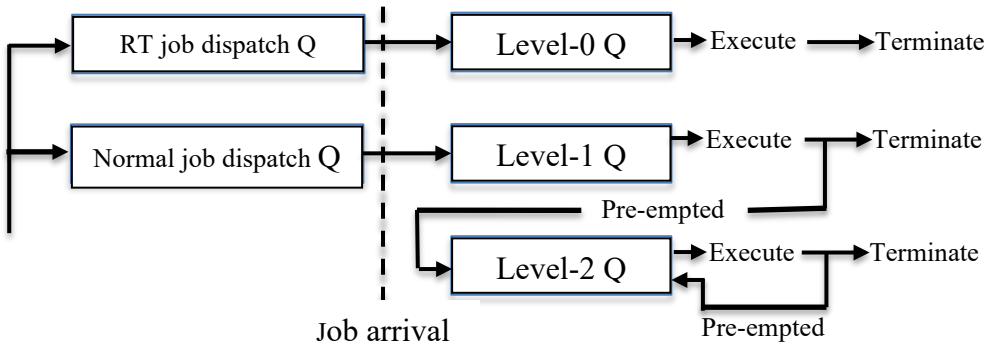
6, 5, 1 ,

8, 2, 0

For Stage 1 of this assignment, you need to use **TWO** Job Dispatch queues **and** a **THREE LEVEL** ready queue to manage and dispatch jobs.

1. At the beginning all jobs in the job dispatch list file are loaded to the Job Dispatch queues, with real-time jobs being loaded in the RT job dispatch queue and normal jobs loaded in the Normal job dispatch queue.
2. When jobs have "arrived", they will be moved from the job dispatch queues to the three-level ready queue and ready to be scheduled for execution. Real-time jobs are

moved to level-0 queue at the top and normal jobs are moved to level-1 queue in the middle of this three-level ready queue.



3. Real-time jobs in Level-0 queue are dispatched in a FCFS manner. When a real-time job is dispatched, it will run to the completion without interruption.
4. Normal jobs in Level-1 queue are also dispatched in a FCFS manner. However, a normal job in Level-1 queue is only given a time-quantum for execution when being dispatched. It will run without interruption within the given time-quantum. If it cannot complete in the time-quantum, the job will be pre-empted and moved to the end (or tail) of Level-2 queue, and its priority value changed to 2 (i.e., its priority is reduced).
5. Jobs in Level-2 queue can be scheduled to run only when both Level-0 and Level-1 queues are empty. When a new job arrives (which will be placed in either Level-0, or Level-1 queue), the currently running Level-2 job will be pre-empted immediately. Instead of moving it back to the end (or tail) of Level-2 queue, it will be placed in the front of Level-2 queue. Therefore, jobs in Level-2 queue are scheduled in a strictly FCFS manner, though the running Level-2 job may be pre-empted by higher priority jobs.
6. The dispatcher will immediately schedule another job for execution when the currently running process is terminated (completed) or suspended (pre-empted), and it always selects a job with the highest priority for execution.

Requirements

Source Code

1. Your program must ask the user to enter an integer value for time-quantum for Level-1 queue.
2. You must keep the logical structure of the program for FCFS dispatcher in Programming Exercise 3, or RR dispatcher in Assignment 2 intact, i.e., you can only make necessary changes to those programs for Multi-Level Queue dispatcher.
3. Your program must be able to correctly calculate and print out on the screen the average turnaround time and average waiting time.
4. Your program must be implemented in the C programming language. C++ features are not permitted except those that are part of an official C standard.
5. Your source code needs to be properly commented and appropriately structured to allow another programmer who has a working knowledge of C to understand and easily maintain your code.

6. You need to include a *makefile* that allows for the compilation of your source code using the *make* command on the School of Computer Science servers. It is crucial that you ensure that your source code compiles correctly on the School of Computer Science servers. If your source code cannot be compiled on the School servers, you will receive a **failing** mark for it.

Discussion Document

1. Write a pseudo code for your Multi-Level Queue Dispatcher. Your pseudo code must have a similar style to that for Round Robin Dispatcher given in Assignment 2 description.
2. Give **2, or 3** job dispatch list files, each list containing **at least 6** jobs, you designed for testing and debugging various aspects of your source code to make sure your code is bug free and can produce correct results. Your must explain **how** these job dispatch lists are designed for **what** purposes.
   Hint: suppose that you want to test if jobs are running as expected in Level-1 and Level-2 queues only. Based on the requirements in the assignment description, you designed a job dispatch list such as the following:
   0, 12, 1
   3, 7, 1
   10, 6, 1
   …
   (You need to give reasons what exactly you want to test in this design.)
   The expected outcome should be:
   a. First set time-quantum = 2. (Give reasons why you set time-quantum = 2 and the expected outcome.)
   b. Then set time-quantum = 4. (Give reasons why you set time-quantum to another value and the expected outcome.)

   …

   (Run your program and see if it can produce the desired outputs.)