COMP3520 Operating Systems Internals Assignment 2 – Uniprocessor Scheduling

General Instructions

This assignment is about uniprocessor scheduling. It consists of two compulsory tasks:

1. Implement a three-level dispatcher as described in the section "A Very Simple Experimental Dispatcher (VSED)"; and
2. Answer the discussion document questions that are provided in a separate document.

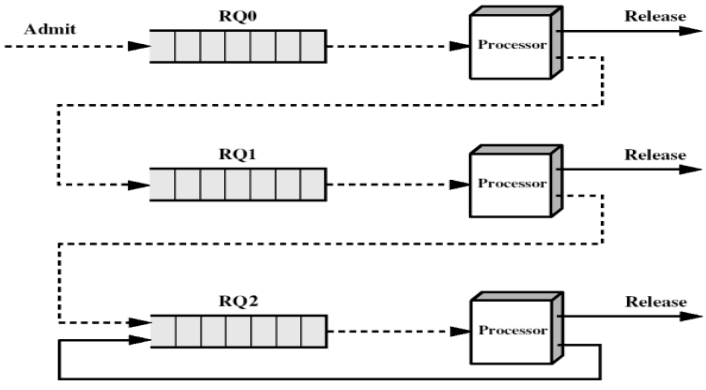
This assignment is an individual assignment. Whilst you are permitted to discuss this assignment with other students, the work that you submit must be your own work. You should not incorporate the work of other students (past or present) into your source code or discussion document.

You will be required to submit your source code and discussion document to *Turnitin* for similarity checking as part of assignment submission. The examiner may use other similarity checking tools in addition to *Turnitin*. Your source code may also be checked.

Submit your source code and discussion document to the appropriate submission inboxes in the COMP3520 Canvas website.

A Very Simple Experimental Dispatcher (VSED)

In this assignment, you will implement a dispatcher called the "Very Simple Experimental Dispatcher (VSED)" that meets the required specifications. VSED is a multiprogramming system with a three-level priority dispatcher that employs user-defined quanta for all priority levels.



The above figure is a three-level feedback dispatcher from Stallings Figure 9.10. Read the textbook and lecture notes to understand how this dispatcher works. (The logic is very similar to Lab Exercise 4 except that instead of enqueueing an incomplete process back onto the same (RR) queue that the process came from, the process is enqueued onto a lower priority process queue (if there is one to receive it). Obviously, the lowest priority queue operates in the same way as a simple Round Robin queue.)

A complete program for the three-level feedback dispatcher with quanta RQ0 = RQ1 = RQ2 = 1 is provided. In this assignment you are asked to modify this program to meet the following requirements:

1. The time quanta RQ0, RQ1 and RQ2 are now three integers and their values (in seconds) will be specified by the user as input parameters to the program. Therefore, a process will run RQ*i* seconds (instead of just 1 second) from level *i* queue.
    a. If the process has not finished execution yet **and** there are waiting jobs of equal or higher priority, it will be pre-empted, its priority will be reduced by one level (i.e. the priority is **increased** by one) and it will be enqueued onto the appropriate priority queue (if the process is already at the lowest priority level, it will simply be enqueued onto the lowest priority queue, which behaves as a simple RR queue).
    b. If completed within RQ*i* seconds, it will be terminated immediately and a new process is selected to run if any.
2. Each process is explicitly assigned an initial priority and should be enqueued onto an appropriate level queue. For example, the following job list (in <arrival time>, <priority>, <processor time> format)
   12, 1, 4
   12, 0, 2
   13, 2, 6

   which indicate:
   1st Job: Arrival at time 12, priority 1 (should be enqueued onto medium level queue), requiring 4 second of CPU time

   2nd Job: Arrival at time 12, priority 0 (onto top level queue), requiring 2 seconds of CPU time

   3rd Job: Arrival at time 13, priority 2 (onto bottom level queue), requiring 6 seconds of CPU time
3. Your VSED needs to print statistics as follows:
    a. Whenever a process is terminated, output the total waiting time and turnaround time for this process in an output file; and
    b. After all jobs in the dispatch list have terminated, print out the average waiting time and average turnaround time.

Additional Requirements

Source Code

The program for the three-level feedback dispatcher with quanta RQ0 = RQ1 = RQ2 = 1 is available on the COMP3520 Canvas website. **You may only modify this program for your VSED. A completely new program will not be accepted for assessment.**

Your solution must be implemented in the C language. C++ features are not permitted except those that are part of an official C standard.

Your source code needs to be properly commented in *Academic English* and appropriately structured to allow another programmer who has a working knowledge of C to understand and easily maintain your code.

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.

You should include your University student identification number (SID) (but **not** your name) in your source code and *makefile*.

You need to write a *readme* file to explain to the user how to compile and run your dispatcher. This document must be written in *Academic English* and be appropriate to audience, purpose and context. The file format should be Plain Text (txt) format.

Testing and Debugging

You are responsible for testing and debugging your source code.

It is crucial that you ensure that the source code you submit compiles correctly on the School of Computer Science servers and that the resulting binary functions as intended. If you submit source code that cannot be compiled on the School servers, you will receive a **failing** mark for it.

Discussion Document

You are required to answer all assigned questions in a separate written document. The questions and requirements specific to the discussion document will be provided in a separate document.

Other Matters

Non-serious attempts, non-attempts, and acts of academic dishonesty are contrary to the spirit of this assignment and COMP3520; accordingly, such behaviour may be sanctioned with **DISQUALIFICATION** from this assignment.

Marking criteria for the source code, *readme* file and discussion document will be provided separately.

To maximize your chances of realizing your full potential in COMP3520, **please start work on this assignment promptly.**