Project 4 - Scheduling

Project Overview and Coding Tasks

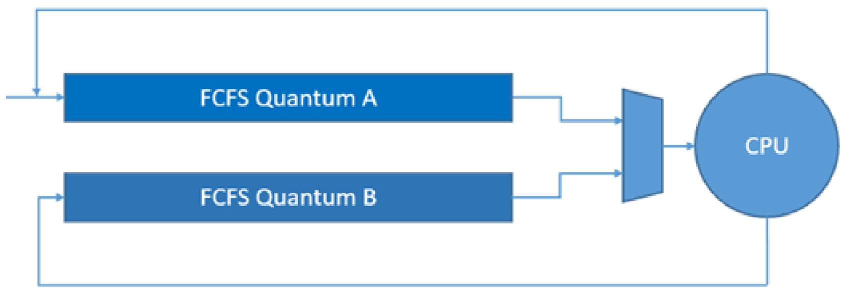
In this programming assignment, you will implement a multilevel feedback queue scheduling algorithm. You will be provided with an input file that includes simulated arrival times of many processes. ~~The goal of the assignment is to determine the best **demotion criteria** and the best **dipatch ratio** among the queues.~~

~~The Best value would be the one that minimizes the average wait time of the dispatched processes. Therefore, you will need to keep track of the wait time of every process in the system and provide the average wait time for all the processes on completion.~~

To make things easier, for this assignment you only need to implement the scheduler for a **dispatch ratio of 5 and 10**, and a **demotion criteria of 5**.

Multilevel Queue Structure

You will implement a mulivel feeback queue with 2 queues. We will refer to the queues as Queue A and B. Both queues operate in a multi level fashion, but Queue A has higher priority than Queue B and processes will be selected from Queue A more often.



- **Both Queue A and B impelement FCFS Round Robin.**
- **Queue A quantum = 5**
- **Queue B quantum = 40**

Diptach Ratio

This will depend on the chosen dispatch ratio. For example, if the dispatch ratio is 5, then after 5 dispatches from Queue A, one process is dispatched from Queue B. IF the dispatch ratio is 0, that means Queue A has abolute priority and processes in Wueue B will only be dispatched when Queue A is empty.

Demotion Criteria

A process is demoted from Queue A to Queue B if it requires more quanta turns than the defined demotion threshold. The demotion threshold determines how many quanta can a process use in Queue A before getting demoted to Queue B. Once a process is demoted it cannot be promoted back. For example, if the demotion threshold is 3 and the process requires 16 seonds of execution, that process will be dispatched from Queue A 3 times from Queue A before it gets demoted to Queue B with 1 more second of execution time remaining.

Input Files

Your program must prompt the user to input the name of the input file, the demotion criteria and the dispatch ratio. The input file with be formatted as follows:

32

6

idle

..

Each line represents a clock tick (e.g. 1 sec) and the arrival of a new process (Process ID). A line with a value of "idle" means that no process arrived at this clock tick. The input file is provided with the project description on Brightspace.

Implementation Details

Your main objective is to explore the best demotion threshold value between the queues A and B along with and best dispatch ratio between the two queues. This section presents some implementation details that you must include in your design. Job fetching is when you add a process to the queue, job dispatching is when a process is chosen for execution at the cpu.

- Job fetching and process dispatching are done at the START of a time tick. This means, that if a process P45 arrives at t = 72, it can join the queue in the same clock tic t = 72 before the wait times are increased. Hence, at the end of t = 73, process P45 would have a wait time of 1.
- At each clock tick, wait times, execution times and any statistics need to be updated.
- Demotion and re-queuing, when applicable, are done at the end of the clock tick, after the wait time and run time are updated.

- New processes always join the higher priority queue A.
- A new process may not use the CPU in the same cycle that it is fetched in even if the queues are empty. Hence a process arriving at t = 5 won't be displaced until t = 6 even if queues are empty. Process dispatch from a queue to the CPU is performed before any fetching or wait time updates. Hence, no process can be fetched and dispatched in the same ticks.

High level order of execution for each time tick should be as follows:

1. Check the status of the current process in execution if any and decide whether to continue executing (it time in cpu < quantom) or exit (if the process finished execution or quantom is used and then decide whether it is terminated or should be requeued to Queue A or B)
2. Dispatch process from Queue if CPU is not busy
3. Queue/Fetch Job from file
4. Update all times

Command line Arguments

Your main function "**Simulation"** (in **Simulation.c** or **Simulation.java)** should prompt the user to input arguments in the following order:

1. Name of the file used.
2. An integer value indicating demotion threshold
3. An integer value indicating the dispatch ratio

As an example: jobs.txt 2 10

An example output (**these are made-up numbers**) would look like this:

```
--------------------------------------------

End Time: 447

Processes Completed: 45

Total execution time: 325

Idle time: 142

Wait time average: 21

--------------------------------------------
```

Files to submit

A zipped file containing the entirety of the implemented code with all source files included.

A document that includes your results ~~(optimal parameter values for the scheduling algorithm) and analyses (How each parameter affects the average wait time).~~

You will be graded on the following criteria:

Code readibilty / routine accepts command-line arguments	20%
Correctness (scheduler algorithm)	60%
Code is compilable	20%