School of Information and Physical Sciences

COMP2240/COMP6240 - Operating Systems

Assignment 1 (10%)

Submit using Blackboard by **11:59 pm, 27th August 2021**

Assignment Task:

Write a program that simulates **First Come First Serve (FCFS)**, **Shortest Remaining Time (SRT)**, **Multi-level Feedback (Variable) (FBV)** and **Lottery (LTR)** scheduling algorithms. For each algorithm, the program should **list the order and time** of the jobs being loaded in the CPU and compute the **turnaround time** and **waiting time** for every job as well as the **average turnaround time** and **average waiting time** for the set of processes.

The average values should be consolidated in a table for easy comparison (*check the sample outputs*).

Two sample input data sets and the corresponding outputs have been supplied. Additional datasets will be used to test your program. The format of the input data will be the same as in the supplied sample files.

Each input data set contains the following information (*check the sample input files for exact format*):

1. Time for running the dispatcher (`DISP`) which can be zero or more
2. For each process: process id (`ID`) , arrival time (`Arrive`), service time (`ExecSize`) and number of tickets the process holds (`Tickets`). It can be assumed that process `P_i` will always arrive before or at the same time of process `P_(i+1)`.
3. A sequence of random numbers to be used (in the given order) in lottery scheduling algorithm. There should be enough random numbers for the LTR algorithm.

Dispatcher: It is assumed that the dispatcher runs to select the next process to run. The dispatcher should behave as follows:

(i) The time to run the dispatcher (context switching time) is fixed and taken as input (`DISP`) from the input file. No other time is wasted in switching between processes other than this.

(ii) If the dispatcher starts at t_1 and finishes at t_2 (*i.e. time to run the dispatcher is* t_2-t_1) then in that run it will choose from the processes that have arrived at or before t_1. It will not consider any process that arrives after t_1 for dispatching in that run.

(iii) If a process P1 is interrupted at t_1 and another process P2 arrives at the same time t_1 then the newly arrived process P2 is added in the ready queue first and the interrupted process P1 is added after that.

(iv) If two processes Px and Py have all other properties same (*e.g. arrival time*) then the tie between them is broken using their ID i.e. Px will be chosen before Py if x<y.

Some details about the scheduling algorithms are as follows:

FCFS: Standard FCFS scheduling algorithm. Process tickets is ignored in scheduling.
SRT: Standard SRT scheduling algorithm. Process tickets is ignored in scheduling.
FBV: Standard multi-level (*4 levels*) feedback algorithm with variable time quanta for different queues. The time quanta in different queues (from highest priority to lowest priority) to be 1, 2, 4 and 4 ms, respectively. The bottom queue (*i.e. the lowest priority queue*) is to be treated in round robin fashion. If a process stays (waiting or running) in the lowest priority queue for more than 32 ms then it should be transferred to the highest priority queue again and the whole process repeats for that process.
LTR: Lottery scheduling is a proportional-share scheduling algorithm. Each process has some *tickets* and the percentage of tickets a process has represents the share of the processor time the process should receive. Now the scheduler holds a lottery to determine which process should get to run next. The process has higher percentage of tickets has a higher chance to win the lottery and should run more often. When a process wins the lottery, it is dispatched for a fixed time quantum of 4ms.

A piece of code snippet (*in C++*) is given below to show how to run a lottery to choose the winner. You must use the random numbers (in order) in the input file instead of the **`.getrandom()`** function below to select the winner. If a process could not finish its execution within the allotted time quantum, then it will re-enter the queue at the end.

```
1   // counter: used to track if we've found the winner yet
2   int counter = 0;
3
4   // winner: use some call to a random number generator to
5   // get a value, between 0 and the total # of tickets
6   int winner = getrandom(0, totaltickets);
7
8   // current: use this to walk through the list of jobs
9   node_t *current = head;
10  while (current) {
11     counter = counter + current->tickets;
12     if (counter > winner){
13        break; // found the winner
14     }
15     current = current->next;
16  }
17  // 'current' is the winner: schedule it...
```

Programming Language:

The programming language is Java, versioned as per the University Lab Environment (**currently Java 11.0.10**). You may only use standard Java libraries as part of your submission.

User Interface:

The output should be printed to the console, and strictly following the output samples given in the assignment package. While there are no marks allocated specifically for the Output Format, there will be a deduction when the result format varies from those provided.

Input and Output:

Your program will accept data from an input file of name specified as a command line argument. The sample files `datafile1.txt` and `datafile2.txt` (*containing the set1 and set2 data*) are provided to demonstrate the required input file format.

Your submission will be tested with the above data and will also be tested with other input files.

Your program should output to standard output (*this means output to the Console*). Output should be strictly in the order FCFS, SRT, FBV, LTR, Summary (*see the* `_output` *files below*).

The sample files `datafile1_output.txt` and `datafile2_output.txt` (*containing output for* `datafile1.txt` *and* `datafile2.txt` *respectively*) are provided to demonstrate the required output (*and input*) format which **must be strictly maintained**.

If output is not generated in the required format then your program will be considered incorrect.

Two Gantt's charts are provided to explain the corresponding behaviour of different algorithms and the dispatcher for the two sample data files.

Mark Distribution:

A general mark distribution can be found in the assignment feedback document (`Assign1Feedback2240.pdf` and `Assign1Feedback6240.pdf`); Note that this is a draft distribution and subject to change.

Additional Task for COMP6240 Students: [NOT for COMP2240 students]

In addition to the above simulations, you are to implement a **Round Robin** (**RR**) scheduling simulation for a **dual processor system**. Assume that two processors share the same ready queue and the **time quantum** for processor 1 is **2 milliseconds** and for processor 2 is **3 milliseconds**, and that the system starts loading processes from processor 1. For dispatching a process, the dispatcher will run on the processor that is idle. If both processors are idle at the same time and there are jobs in the ready queue then at first the dispatcher will run on processor 1 and it will dispatch a process in processor 2, if there are more processes, it will be dispatched to processor 1 in the same run of dispatcher. If one processor becomes idle, then the dispatcher will run in that processor and will dispatch a process for it. Output the same data as required for the other simulations. Additionally you will need to specify in which processor a job is assigned at a specific time [i.e. instead of `T1: p1(0)` use `P1-T1: p1(0)` to clarify that the *process 1* (`p1`) is running in *processor 1* (`P1`) starting at *time 1* (`T1`)].

Submission Information for COMP2240 and COMP6240:

Deliverable:

1. Your submission will contain your program source code, documentation (*below*) and a `readme.txt` (*containing any special instructions required to compile and run the source code*) in the root of the submission. These files will be zipped and submitted in an archive named **`c9876543.zip`** (*where* **`c9876543`** *is your student number*) – do not submit a `.rar`, or a `.7z`, or etc.

2. Your main class should be **`A1.java`** you program will compile with the command line **`javac A1.java`** and your program will be executed by running **`java A1 input.txt`**.

3. Brief **1 page** (*A4*) report, reviewing the results from your program and any interesting observations. Specifically, write a note about the relative performance of the algorithms based on your implemented versions of the algorithms; why did you get the relative results you got, were there any variances from what you expected, did you encounter any issues?

4. Completed Assignment Coversheet.

5. Any Adverse Circumstances Extension information you may have received.

NOTES:
1. Assignments submitted after the deadline (**11:59 pm 27th August 2021)** will have deducted 10% of the maximum marks possible, per day late, in line with UoN Policy. This means (*for example*) if you submit **two days late**, and score **80%** in the assignment, you mark will be **80** (mark) **– 20** (2 * 10% maximum possible mark (100%)) **= 60**.
2. If your assignment is not prepared and submitted following above instructions then you will lose most of your marks despite your algorithms being correct – *the specifications are not-negotiable!*

Dan and Nasim
02AUG21 v1.0