Assignment 03 - Scheduling

This assignment complements Lab 02 on scheduling. It requires you to modify the scheduling policy simulator in order to implement more advanced scheduling algorithms.

Question 1 - Round Robin

Extend the simulator so that it enforces time sharing among processes. The quantum duration is a command line parameter entered by the user upon starting a simulation.

For instance:

```
$ bin/sched-simulator tasksX RR 4
```

will simulate a round robin policy on the tasks defined in file `tasksX` and with a quantum of 4 CPU cycles.

Question 2 - Multilevel Feedback Queue

Extend the simulator so that it enforces the following multilevel feedback queue policy.

Every queue enforces FCFS, and is associated with a priority level L. There are 3 queues: level 1 (highest priority), level 2, and level 3 (lowest priority). Upon submission, every process gets inserted in the queue with the highest priority (L = 1). Upon election, the allocated processing time depends of the priority level of the queue: a process gets L * D units of time (D a command line parameter) on the processor once it is elected. If the task does not complete within this quantum, then it gets evicted from the processor and relegated to the next priority level queue. Processes from queue level 3 that do not complete in their allocated quantum get pushed back to level 1.

The following command line:

```
$ bin/sched-simulator tasksX MFQ 3
```

will simulate this multilevel feedback queue policy on the tasks defined in file `tasksX` and with a quantum of 3 CPU cycles.

Question 3 (Bonus) - Interactive Tasks

This is a bonus question. You may hand in your assignment with solutions for questions 1 and 2 only, and still get full points (24 out of 24) if your solutions are correct. If you provide a solution for question 3, we will grade this question also: the result will be added to your total, up to the 24 points maximum. In other words, question 3 is a bonus question in the sense that it acts as a safety net for your assignment grade.

Extend the simulator so that the Round Robin policy can also take I/O requests into account. Task descriptions must also be extended to allow this kind of simulation: they now include the duration and periodicity of the I/Os.

Here is an example of extended task file content:

```
T1 10 0 3 5
T2 12 2 2 4
T3 5 3 0 0
```

In this example: T1 gets inserted at time 0, and completes after 10 CPU cycles and two I/O requests that will last 3 cycles each (the first one directly after the 5th running cycle, and the

second one directly after the 10th); T3 gets inserted at time 3 and completes after 5 CPU cycles (it never makes any I/O request). In other words, T1 makes an I/O request every time it runs for 5 (possibly non-consecutive) cycles, and must wait 3 clock cycles for each I/O to complete.

The following command line:

```
$ bin/sched-simulator tasksY IORR 4
```

will simulate a round robin policy that accounts for I/Os on the tasks defined in file `tasksY` and with a quantum of 4 CPU cycles.

In order to compensate for the fact that Round Robin favors CPU-bound tasks over I/O-bound tasks, the IORR algorithm shall work as follows. Any new task gets inserted at the end of the queue. The processor gets allocated to the first task from the queue that is ready. The elected task gets evicted either when it requests an input/output or when it has exhausted its quantum: it is then reinserted at the end of the queue. If a task that requested an I/O reaches the front of the queue before the awaited response, the scheduler leaves it there and looks further down the queue for a task that is ready.