Fundamentos de Sistemas de Operação

MIEI 2018/2019

Homework Assignment 1

Deadline and Delivery

This assignment is to be performed ***individually*** by each student – any detected frauds will cause failing the discipline. The code has to be submitted for evaluation via the Mooshak system (`http://mooshak.di.fct.unl.pt/~mooshak/`) using each student's individual account -- the deadline is ***17h00, October 19th, 2018 (Friday)***.

Description

The goal of this assignment is to implement a process's scheduling algorithm in an OS simulator. The simulator is written in Java and includes the simulation of several hardware and OS concepts in a very simplified way. In this assignment, you will only have contact with a small part of the simulator, namely concerning process scheduling.

Scheduling algorithm

The scheduler to be implemented features two levels of priority, materialized into two queues, and behaves as follows:

1. Every process starts with the maximum priority (and hence is placed in the high priority queue);
2. The next process to be dispatched for execution is selected from the ones with highest priority, using Round-Robin among the processes in the same queue;
3. Process execution is limited by a *time-slice* (or *quantum*);
4. A process always runs the quantum assigned to it until the end, if it that means exceeding the quota. For example, if a process' quota is 7 and the quantum is 10, the process runs 10 clock ticks and not 7.
5. After running an accumulated *time-quota* of CPU time, a process drops to the low priority queue;
6. When the IO of a process finishes, that process is added to the high priority queue, and its quota is renewed;

The high and low priority queue use *time-slices* of 10 and 20 clock ticks, respectively. In turn, the scheduler uses *time-quotas* of 20 clock ticks. Note that, when a process is blocked or preempted, you should decrement the used time of that process' quota; if the quota becomes zero or bellow, then the process gets its priority lowered and a new quota;

The Simulator

Your work will be confined to the implementation of class caoss.simulator.os.scheduling.FSOScheduler, of which a skeleton is already given. This class implements the caoss.simulator.os.Scheduler interface that comprises the following 5 methods (that trigger the several processes' state changes):

- newProcess(Program prog) – This method is called by the simulator upon a program's execution request and must create a process to execute the program prog.
- ioRequest(ProcessControlBlock<SchedulingState> pcb) – It is invoked when the process (with control block pcb) that is running in the CPU requests an input/output operation.
- ioConcluded(ProcessControlBlock<SchedulingState> pcb) – This one is called when the input/output operation requested by the process (with control block pcb) ends.
- quantumExpired(ProcessControlBlock<SchedulingState> pcb) – This method is invoked when the process (with control block pcb) running in the CPU exhausts its quantum (*time-slice*).
- processConcluded(ProcessControlBlock<SchedulingState> pcb) – It is called when the process (with control block pcb) has concluded its execution.

To accomplish your assignment, you must also study other classes of the simulator, namely:

- caoss.simulator.Program – Describes the program executed by a process.
- caoss.simulator.os.ProcessControlBlock – Defines the information required by the system to manage a process' execution. For instance, the *pid* and the process' time of arrival to the system.
- caoss.simulator.os.scheduling.SchedulingState – The scheduling information that must be kept for each process in the system is represented in this class. Examples of the information you may use are the *time-quota* given to the process, its priority, and the last time when the process was scheduled.
- caoss.simulator.os.Dispatcher – Loads the execution context of a process in the target CPU. If there is no process to dispatch, your code will **mandatorily** have to dispatch the idle process, i.e. **null**:

  Dispatcher.dispatch(some_pcb) or Dispatcher.dispatch(**null**)

- caoss.simulator.hardware.Clock – Defines the computer's clock. The method to obtain the current time is:

  Hardware.clock.getTime()

- caoss.simulator.hardware.Timer – Implements the computer's timer. It may be programmed to notify the scheduler that the quantum (*time-slice*) assigned to a process has expired.
  - To obtain the simulator's timer use the following line of code:

    Timer timer = (Timer) Hardware.devices.get(DeviceId.TIMER);

  - To program the timer, use the set() method. For instance, timer.set(10) programs the timer to interrupt the CPU and gives control to the Operating System, in 10 time units.
- caoss.simulator.os.Logger – This is the Operating System's logger. You must use it to output the result of your scheduling algorithm. Concretely, you must use the static method Logger.info(). You may change the detail level of the logger by altering line 34 of class caoss.simulator.os.Logger to LOGGER.setLevel(Level.ALL). This modification will not have impact on your submission to Mooshak.

To simplify your understanding of the simulator, you may find the implementation of a Round Robin scheduler in class caoss.simulator.os.scheduling.RoundRobinScheduler. You can test this scheduler by copying its code to the caoss.simulator.os.scheduling.FSOScheduler class or by altering the field scheduler of class caoss.simulator.os.FSOOS.

You also have *javadoc* information at your disposal in folder *doc*.

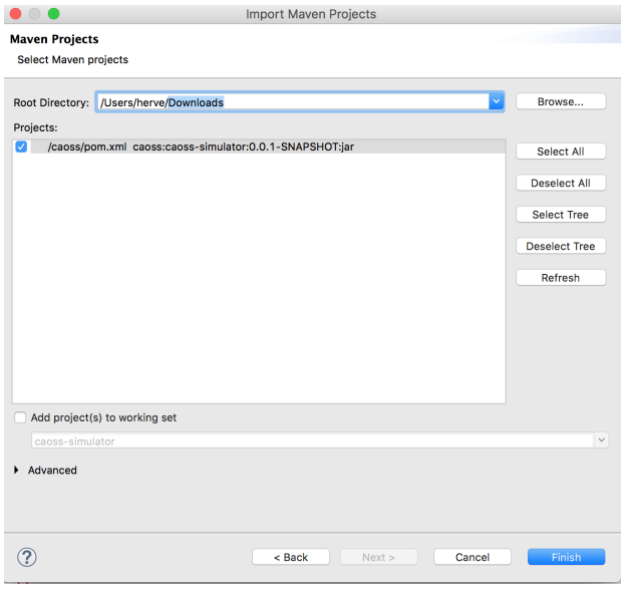
Output issues:

1. All processes waiting for input/output operations must be placed in queue blocked, so that their status is printed by the logQueues method. See example below.
2. You must output the information that a process has expired its quota. To that end, use the following code line:

   Logger.info("Process " + pcb.pid + ": quota expired");

Development environment, compilation and execution

The simulator's source code (available from CLIP) is a Maven managed project. You may import it to Eclipse or some other IDE. For instance, in Eclipse, use *File* → *Import ...* → *Maven* → *Existing Maven Project*, select the *caoss* project, and click on the *Finish* button.



Subsequently, you will be able to execute the simulator by running class caoss.simulator.CAOSS.

If you prefer to compile and run the project from the command line, compile it using the `mvn` command:

```
cd caoss
mvn compile
```

To run the caoss.simulator.CAOSS class type (in directory *caoss*):

```
java -cp target/classes caoss.simulator.CAOSS
```

The simulator presents a command line from where you may simulate a program execution:

```
exec examples/ex1.caoss
```

or multiple programs at once:

```
exec examples/ex1.caoss examples/ex2.caoss examples/ex3.caoss
```

To terminate the simulator's execution type:

```
shutdown
```

Bibliography

[1] Chapter 8 of the recommended book, "Operating Systems: Three Easy Pieces Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau"

Submission to Mooshak

You only have to submit your implementation of class caoss.simulator.os.scheduling.FSOScheduler (FSOScheduler.java).

Example of a Schedule

See below the example of a schedule using the algorithm that you will have to implement.

Command: `exec examples/ex1.caoss examples/ex1.caoss examples/ex4.caoss`

Result:

```
Create process 0 to run program examples/ex1.caoss
Run process 0 (quantum=10, quota=20)
Queue 0: []
Queue 1: []
Blocked []
Create process 1 to run program examples/ex1.caoss
Queue 0: [1]
Queue 1: []
Blocked []
Create process 2 to run program examples/ex4.caoss
Queue 0: [1, 2]
Queue 1: []
Blocked []
Process 0: quantum expired
Run process 1 (quantum=10, quota=20)
Queue 0: [2, 0]
Queue 1: []
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: [0, 1]
Queue 1: []
Blocked []
Process 2: IO request
Run process 0 (quantum=10, quota=10)
Queue 0: [1]
Queue 1: []
Blocked [2]
Process 0: quantum expired
Process 0: quota expired
Run process 1 (quantum=10, quota=10)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Process 1: quota expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
```

```
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
```

```
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
```

```
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: IO request
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [0, 1]
Blocked []
Process 2: IO request
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked [2]
Process 2: IO concluded
Queue 0: [2]
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 2 (quantum=10, quota=20)
Queue 0: []
Queue 1: [1, 0]
Blocked []
Process 2: execution concluded
Process 2: turnarround time: 482
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked []
Process 1: quantum expired
```

```
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked []
Process 0: quantum expired
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: [0]
Blocked []
Process 1: quantum expired
Run process 0 (quantum=20, quota=20)
Queue 0: []
Queue 1: [1]
Blocked []
Process 0: execution concluded
Process 0: turnarround time: 677
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: []
Blocked []
Process 1: quantum expired
Run process 1 (quantum=20, quota=20)
Queue 0: []
Queue 1: []
Blocked []
Process 1: execution concluded
Process 1: turnarround time: 704
Queue 0: []
Queue 1: []
Blocked []
```