Software Design III 3BB4

Assignment IV

Jadees Anton *1213386* | Connor Hallett *1158083* | Spencer Lee *1224941* | Nicolas Lelievre *1203446*

March 31, 2015

Round Robin Scheduler

This document elaborates the round-robin scheduler Java implementation and how it functions functions in order to schedule a series of processes.
Overview of usage of `RRScheduler`
To generate a single process, press start on Generator. The thread panel will turn 10 degrees. Multiple processes can be generated, turning the thread panel 10 degrees for each one.
To start loading and execution, press start on CPU
CPU thread panel will turn once for each cycle of execution. Process is removed from queue when loaded, executed for 1 cycle, then recycled to the back of the queue if there is remaining execution time.

Below is the according LTS model.

```
1 asdf
```

ReadyQueue.java

Utilised `ConcurrentLinkedList` Java library.
Implements a queue in a first in first out (FIFO) manner.
In our implementation, ReadyQueue holds processes (see process class).

Generator.java

Uses the `Random` java library in order to randomise the length of execution time of every process (1-5 cycles).
Class generates processes to be executed by the CPU.
Also assigns unique IDs to the processes.
Implementation: each time you click start, the image rotates 10 degrees demonstrating a process being generated. Not run until CPU is executed.

Dispatcher.java

Looks at the queue and loads the process at the head of the queue to the CPU.

CPU.java

Calls the dispatcher to load a process for execution.
Executes the process for 1 cycle.
If the process has not completed full execution after one run cycle, it is recycled to the back of the queue for

impending completion.
Otherwise, if there is no remaining run time, the process is removed.

GrimReaper.java

Looks at processes after the CPU has processed them for 1 cycle. If there is remaining runtime, Grimreaper recycles the process to the back of the queue. If there is no remaining runtime, the process is not recycled.

Process.java

Implements the processes that are scheduled and run.
Each process created has a total runtime (random number between 1-5 cycles) and a unique process ID.
Runtime decremented once by CPU each time it is run.
Processes stored in ReadyQueue.

RRScheduler.java

Applet class. Contains all labels specifying what is being executed.
Messages display:
What process is loaded in the CPU
How many cycles that process has been executed by the CPU
What processes is finished executing
What process is being recycled

Desirebale Properties

Many Processes are Generated

Generator allows multiple process to be generated. Simply press start multiple times. Each time the generator threadpanel turns 10 degrees displays a process being generated.

Deadlock Free

System is deadlock free. All processes in the queue will be always be serviced eventually. There is no specific notion of priority in execution between processes. Processes are processed by the CPU in the order they are in the queue.

No Incomplete Process Destruction

Processes will never be destroyed when there is still remaining execution time. Grimreaper will always recycle a process back to the queue if there is remaining execution time.

No Overuse of CPU Time

There is never an overuse of CPU time. The CPU will always execute the process at the front of the queue for one cycle (and only one cycle). Grimreaper will then see if execution time is remaining and recycle if needed.

No Incorrect Process Execution

Processes will never be recycled back to the queue if they have no remaining execution time. When being loaded to the CPU, the process is removed from the queue. After execution for 1 cycle, it is only then recycled IF there is remaining execution time. If there is no remaining execution time, it will not be recycled to the queue.

Testing

Used message labels to ensure appropriate workflow.