Software Design III 3BB4

Assignment III

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

March 17, 2015

Round Robin Scheduler

This document elaborates on how the `rr.lts` LTS functions in order to schedule a series of processes using round-robin scheduling.

Below is said LTS model.

```
1  const MaxIF=6
2  range IF=0..MaxIF-1
3  const MaxCPU=5
4  range CPUT = 0..MaxCPU
5
6  GENERATOR = GENERATOR[0],
7  GENERATOR[num:IF] = (enterarrq[num] -> GENERATOR[(num+1)%MaxIF]).
8
9  QUEUE=(enterarrq[num:IF] -> QUEUE[num]),
10 QUEUE[n0:IF] = (
11     enterarrq[num:IF] -> QUEUE[n0][num]
12     | cpu.serve[n0] -> QUEUE),
13 QUEUE[n0:IF][n1:IF] = (
14     enterarrq[num:IF] -> QUEUE[n0][n1][num]
15     | cpu.serve[n0] -> QUEUE[n1]),
16 QUEUE[n0:IF][n1:IF][n2:IF] = (
17     enterarrq[num:IF] -> QUEUE[n0][n1][n2][num]
18     | cpu.serve[n0] -> QUEUE[n1][n2]),
19 QUEUE[n0:IF][n1:IF][n2:IF][n3:IF] = (
20     enterarrq[num:IF] -> QUEUE[n0][n1][n2][n3][num]
21     | cpu.serve[n0] -> QUEUE[n1][n2][n3]),
22 QUEUE[n0:IF][n1:IF][n2:IF][n3:IF][n4:IF] = (
23     enterarrq[num:IF] -> QUEUE[n0][n1][n2][n3][n4][num]
24     | cpu.serve[n0] -> QUEUE[n1][n2][n3][n4]),
25 QUEUE[n0:IF][n1:IF][n2:IF][n3:IF][n4:IF][n5:IF] = (
26     cpu.serve[n0] -> QUEUE[n1][n2][n3][n4][n5]).
27
28 CPU = CPU[0],
29 CPU[t:CPUT] = (
30     when (t==0) serve[c:IF] -> quantums[i:1..MaxCPU] -> CPU[i]
31     |when (t!=0) tick -> CPU[t-1]).
32
33 ||RR_SCHEDULER = (cpu:CPU || GENERATOR || QUEUE).
```

GENERATOR

The `GENERATOR` creates processes to be executed by the `CPU`. As long as there is enough space for a new process in the queue, `GENERATOR` enqueues it.

Design Decision I
Processes to be executed have run-times that are later defined by the system.

The `GENERATOR` pushes processes to the queue in a *first-in-first-out* (FIFO) manner. As per the requirements outlined by the assignment, processes are immediately enqueued but not loaded into the CPU at this point in execution.

QUEUE

The `QUEUE` holds processes (generated by `GENERATOR`) that are ready for execution by the `CPU`.

Design Decision II
The `QUEUE` *currently has space for 6 processes that are ready for execution. It is, however, infinitely expandable.*

Our `QUEUE` holds and enqueues the processes waiting for execution in a *first-in-first-out* (FIFO) manner.

CPU

The `CPU` takes the first process from `QUEUE` that is ready to be executed and loads it, similar to the functionality theorised by `DISPATCHER`. Process execution time is defined by the user at this step.

Design Decision III
Any process that has been selected to execute will run until completion. Note that the process will not be re-queued at the end of said queue.

Once a process execution has been completed, the `CPU` unloads it, similar to the functionality of `GRIMREAPER`.

Design Decision IV
`CPU` *also contains the functionalities of* `DISPATCHER` *and* `GRIMREAPER`.

RR SCHEDULER

`RR_SCHEDULER` is a parallel composition of the above three processes. Hence, it assembles/concatenates these defined processes in order to create a comprehensive model including all necessary components of the described system.

Also note that our implementation of `RR_SCHEDULER` is completely deadlock-free due to the user-scalable quantum defined within the system.