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.

```
const MaxIF=6
range IF=0..MaxIF-1
const MaxCPU=5
range CPUT = 0..MaxCPU

GENERATOR=GENERATOR[0],
GENERATOR[num:IF] = (enqueue[num] -> GENERATOR[(num+1)%MaxIF]).

QUEUE = (enqueue[num:IF] -> QUEUE[num]),
QUEUE[n0:IF] = (
    enqueue[num:IF] -> QUEUE[n0][num]
    | cpu.load[n0] -> QUEUE),
QUEUE[n0:IF][n1:IF] = (
    enqueue[num:IF] -> QUEUE[n0][n1][num]
    | cpu.load[n0] -> QUEUE[n1]),
QUEUE[n0:IF][n1:IF][n2:IF] = (
    enqueue[num:IF] -> QUEUE[n0][n1][n2][num]
    | cpu.load[n0] -> QUEUE[n1][n2]),
QUEUE[n0:IF][n1:IF][n2:IF][n3:IF] = (
    enqueue[num:IF] -> QUEUE[n0][n1][n2][n3][num]
    | cpu.load[n0] -> QUEUE[n1][n2][n3]),
QUEUE[n0:IF][n1:IF][n2:IF][n3:IF][n4:IF] = (
    enqueue[num:IF] -> QUEUE[n0][n1][n2][n3][n4][num]
    | cpu.load[n0] -> QUEUE[n1][n2][n3][n4]),
QUEUE[n0:IF][n1:IF][n2:IF][n3:IF][n4:IF][n5:IF] = (
    cpu.load[n0] -> QUEUE[n1][n2][n3][n4][n5]).

CPU = CPU[0],
CPU[t:CPUT] = (
    when (t==0) load[c:IF] -> quantums[i:1..MaxCPU] -> CPU[i]
    | when (t!=0) tick -> CPU[t-1]).

||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. These processes, however, are 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.