UNIVERSIDAD DE MÁLAGA | uma.es

Dpt. of Languages and Computer Science

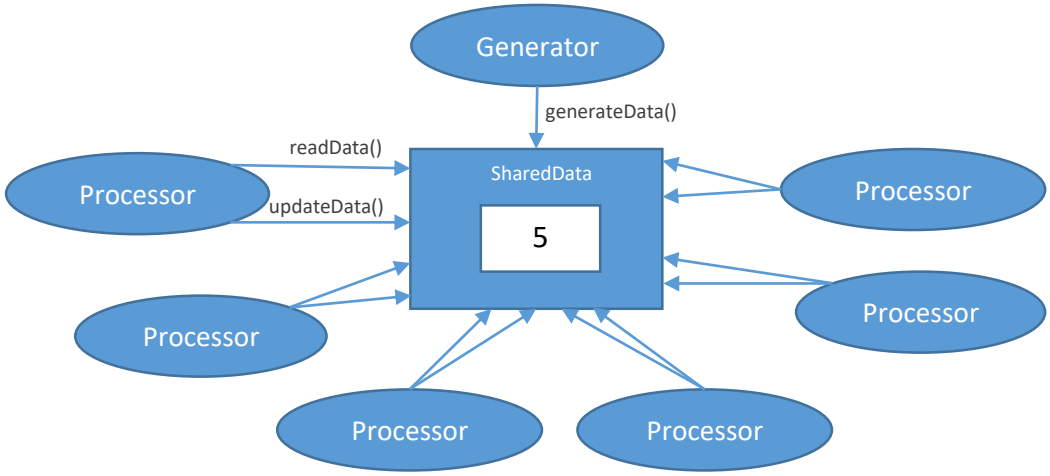
Systems Programming and Concurrency

2nd Official Exam. Course 2020-2021
September 1st, 2021.

Block 1. Concurrency

Description of the system

We have to develop a system to process data concurrently. Such a system is composed by **a** thread that *generates data* and **N** threads that *consumes data*. Communication among threads is carried out by means of a shared resource where is stored the piece of data currently being processed. To simplify this exercise, let's assume that the piece of data is a single integer and that each processor (consumer) simply increments in 1 the piece of data read.



Each entity of the previous figure behaves as follows:

- Class `Generator`. Thread that generates a datum and calls to the method `generateData()` to store it into the shared resource `SharedData`. Once a datum has been generated, no new one may be generated until all the processor had processed it. The result of the whole processing will be returned as a result of the same call to `generateData()`.This class is already implemented.
- Class `Processor`. Thread that reads a datum by calling to the method `readData()` and process it. Once it has processed the datum, this thread stores again the resulting value into the shared resource by using the method `updateData()`. This class is already implemented.
- Class `SharedData`. Shared resource to synchronize both the data generator and their processors. It provides to them the next methods:
  - `public SharedData(int numProcessors)`. The constructor of this class takes as a parameter the amount of processors that will manipulate each generated datum. Such a value must be greater than 0.
  - `public int generateData(int data)`. A `Generator` uses this method to store a new datum. Once stored, the `Processor` threads must be informed of the new datum available. The generator must wait inside this method until the datum is fully processed. Once that happens, the result of processing will be returned, so a new datum could be generated.

- `public int readData(int id)`. The `Processor` with identifier **id** uses this method to read a datum to process. It will wait if there is no datum available or if the same processor **id** has already processed that datum.
- `public int updateData(int id, int newValue)`. The `Processor` with identifier **id** stress into the shared resource the result of processing the datum. Once this done, it will behave the next way: 1) it will notify any pending processors, or 2) it will notify the generator if it was the last pending processor.

In addition, the class `Driver` includes an already implemented `main()` method that creates a `Generator` and **N** `Processors`.

Please, note that this exercise includes three synchronization conditions:

- CS-Generator. Once the datum is stored, the `Generator` waits for every `Processor` to process it before generating a new one.
- CS1-Processor. A `Processor` waits if there is no new datum to process. This may happen because the `Generator` has not put yet any datum or because that datum has already been processed by that `Processor`.
- CS2-Processor. A `Processor` waits if there is a datum to process but other `Processor` is processing it.

You have to develop two implementations of the class `SharedData`:

a) Using binary semaphores (5 points).
b) Using monitors or locks (5 points).

Following is shown a trace with N=10 processors and M=3 data:

```
Datum to process: 88
Amount of remaining processors: 10
       Processor 5 has processed the datum. New datum: 89
Number of remaining processors: 9
       Processor 4 has processed the datum. New datum: 90
Number of remaining processors: 8
       Processor 3 has processed the datum. New datum: 91
Number of remaining processors: 7
       Processor 2 has processed the datum. New datum: 92
Number of remaining processors: 6
       Processor 0 has processed the datum. New datum: 93
Number of remaining processors: 5
       Processor 1 has processed the datum. New datum: 94
Number of remaining processors: 4
       Processor 9 has processed the datum. New datum: 95
Number of remaining processors: 3
       Processor 8 has processed the datum. New datum: 96
Number of remaining processors: 2
       Processor 7 has processed the datum. New datum: 97
Number of remaining processors: 1
       Processor 6 has processed the datum. New datum: 98
Number of remaining processors: 0
Result of processing = 98
------------------------------
```

```
Datum to process: 59
Amount of remaining processors: 10
       Processor 6 has processed the datum. New datum: 60
Number of remaining processors: 9
       Processor 7 has processed the datum. New datum: 61
Number of remaining processors: 8
       Processor 0 has processed the datum. New datum: 62
Number of remaining processors: 7
       Processor 2 has processed the datum. New datum: 63
Number of remaining processors: 6
       Processor 3 has processed the datum. New datum: 64
Number of remaining processors: 5
       Processor 4 has processed the datum. New datum: 65
Number of remaining processors: 4
       Processor 1 has processed the datum. New datum: 66
Number of remaining processors: 3
       Processor 5 has processed the datum. New datum: 67
Number of remaining processors: 2
       Processor 9 has processed the datum. New datum: 68
Number of remaining processors: 1
       Processor 8 has processed the datum. New datum: 69
Number of remaining processors: 0
Result of processing = 69
------------------------------
Datum to process: 75
Amount of remaining processors: 10
       Processor 8 has processed the datum. New datum: 76
Number of remaining processors: 9
       Processor 9 has processed the datum. New datum: 77
Number of remaining processors: 8
       Processor 6 has processed the datum. New datum: 78
Number of remaining processors: 7
       Processor 7 has processed the datum. New datum: 79
Number of remaining processors: 6
       Processor 0 has processed the datum. New datum: 80
Number of remaining processors: 5
       Processor 2 has processed the datum. New datum: 81
Number of remaining processors: 4
       Processor 3 has processed the datum. New datum: 82
Number of remaining processors: 3
       Processor 1 has processed the datum. New datum: 83
Number of remaining processors: 2
       Processor 5 has processed the datum. New datum: 84
Number of remaining processors: 1
       Processor 4 has processed the datum. New datum: 85
Number of remaining processors: 0
Result of processing = 85
------------------------------
```