Monitor assignment: double buffer

Students:

Federico Fabbri (0001025916)

Jacopo Merlo Pich (0001038025)

Daniele Simonazzi (0001058929)

Domain of the problem

We implemented the buffer as an array containing 2 integers. We considered each cell as "empty" when its value is zero. In order to keep track of the buffer state, we used an enumerator *state_t*, which contains the following states:

- EMPTY: both values equal to 0;
- HALF: one of the two values equal to 0;
- FULL: both values are not zero.

Using this enumerator each thread "knows" if it can execute or not, depending on the actual state of the buffer.

Design of the monitor

We designed the monitor as follows:

- a mutex lock *m* protect the critical sections;
- two integer variables *nw1, nw2* (number wait 1, number wait 2) to keep track of the number of processes waiting, respectively for *push_int*, *push_pair*;
- two condition variables *one, two* to handle each specific sleeping queue;
- a type *state_t* enumerator *statevar.*

Push_int

The integer *value* is copied in one of the buffer's cells only if the state of the buffer is not *FULL* and there is no *push_pair* waiting. These conditions are checked in the while cycle. If the process can proceed, *nw1* is not incremented and the *value* is copied in the first empty cell. If not, the process will wait in the sleeping queue *one* and increment *nw1*. The buffer state is eventually updated.

Push_pair

The pair of integers (*value1, value2*) is copied in the buffer's cells if the buffer is *EMPTY*. This condition is checked in the while cycle. If the process can proceed, *nw2* is not incremented and the

integers are respectively copied in the buffer. If not, the process will wait in the sleeping queue *two* and increment *nw2*. The buffer state is eventually updated.

Fetch

Returns one of the two numbers in the buffer, and it sets the specific cell to 0 (empty). If the buffer is *FULL*, the first cell value is returned. If the buffer is *EMPTY*, 0 is returned. Also, it eventually "wakes up" the correct processes (*push_int, push_pair*) if there is any waiting in the queues. We have three possible scenarios to handle:

1. Buffer *EMPTY*: nothing can be fetched from the buffer. *nw2, nw1* are respectively checked in order to send the "signal" to the correct queue (*push_pair* has priority);
2. Buffer *FULL*: the value of the first cell is fetched (we chose this policy). If there is a *push_int* waiting ($nw1 \neq 0$) and there is no *push_pair* waiting ($nw2 = 0$) a "signal" is sent to the *one* condition variable (*push int* woken up);
3. Buffer *HALF*: the value of the non-empty cell is fetched. *nw2, nw1* are respectively checked in order to send the "signal" to the correct queue.

Remarks

We added some sleep() inside the threads code in order to slow down their execution and get a more clear console output.

Running the program with sleeping times equal to 1 second, we noticed that a sort of starvation problem occurs: the *push_pair* is executed way more times than the *push_int*, due to the priority policy. In order to even out the number of executions of the two types of thread we could change the sleeping times, making the *push_pair* sleep much longer (3 sec.) than the *push_int.*