

April 25th, 2024

Oblig 5: Synchronization

Mandatory Assignment 5 in IN3030 / IN4330 - v2024

Eric Jul

Deadline: May 2nd, 2024 23:59:00

This oblig is about implementation of one synchronization primitive using another.

You are to implement a variant of `waitNext` from last year's exam.

You are welcome to base your solution on the published `WaitNextC.java`.

You are to write a program that implements a new synchronization primitive called `waitAndSwap` with the following semantics:

When a thread makes the first call of `waitAndSwap`, it waits. When a second thread calls `waitAndSwap`, it does not wait. When a third thread calls `waitAndSwap`, it releases the first thread and then it waits. When a fourth thread calls `waitAndSwap`, it does not wait... and so forth, i.e., the effect is that the threads are released from `waitAndSwap` in the following call order: 2, 1, 4, 3, 6, 5, 8, 7, ...

We can say that the threads are "swapped" pair-wise.

You are welcome to base your solution on the published `WaitNextC.java`.

You are to write a test program that demonstrates that your `waitAndSwap` works.

You are welcome to do this using the test output method `debugPrintln` and the `variSpeed` method from the `WaitNextC.java` program.

Hand in the test program including the implementation of `waitAndSwap` and suitable output showing your synchronization method works as intended. Explain in a report that follows the report guidelines.

NOTE: you are to use semaphores only for synchronization and you can use *only* the `acquire` and `release` methods except that you can use access methods such as `availablePermits` but *only inside test output* similar to the `printSems` method.

IMPORTANT: there must be substantial amounts of code and a good deal of the report done before you can qualify for a second try. This means that if either the code or the report is missing, you will automatically fail with no offer of a second try.