Istanbul Technical University

January 3, 2022

ISE302 Operating Systems
ASSIGNMENT - 2

Due Date: Sunday, January 16, 2022, 23:59.

• Please write and draw neatly in your report and add comments in your source code.

• **Consequences of plagiarism:** Any cheating will be subject to disciplinary action.

• **No late submissions** will be accepted.

• **Submissions:** Submit your source code(hw2.c) to Ninova. Please do not forget to write your full name (first name and last name) and Student ID in your source code.

If you have any questions, please e-mail teaching assistant **Esin Ece Aydın (aydinesi16@itu.edu.tr)**.

1 You are asked to write a program using threads in order to solve the given problem below:

Water is formed when one oxygen atom forms a single covalent bond with two separate hydrogen atoms. Imagine that threads represent oxygen and hydrogen atoms. In order to assemble these threads into water molecules, we have to create a barrier that makes each thread wait until a complete molecule is ready to proceed. As each thread passes the barrier, it should invoke bond().

There are three main restrictions to follow:

- You must ensure that all threads from one molecule call bond() before any threads from the next molecule.
- If an oxygen thread arrives at the barrier when no hydrogen threads are present, it has to wait for two hydrogen threads.
- If a hydrogen thread arrives at the barrier when no other threads are present, it has to wait for an oxygen thread and another hydrogen thread.

The variables to be used in the solution:

```
mutex = Semaphore (1)
oxygen = 0
hydrogen = 0
hydroBonded = Semaphore (0)
oxyQueue = Semaphore (0)
hydroQueue = Semaphore (0)
```

You will code the solution for the explained problem by using the following pseudo codes.

The pseudo code for Oxygen atom:

```
mutex.wait()
oxygen += 1
if hydrogen >= 2:
hydroQueue.signal(2)
hydrogen -= 2
oxyQueue.signal()
oxygen -= 1
else:
mutex.signal()

oxyQueue.wait()

bond()

hydroBonded.wait(2)
mutex.signal()
```

The pseudo code for Hydrogen atom:

```
mutex.wait()
hydrogen += 1
if hydrogen >= 2 and oxygen >= 1:
hydroQueue.signal(2)
hydrogen -= 2
oxyQueue.signal()
oxygen -= 1
else:
mutex.signal()

hydroQueue.wait()

bond()

hydroBonded.signal()
```

Implementation Details

1. Program inputs are the number of oxygen atoms**(n)** and the number of hydrogen atoms**(m)**. These inputs should be received as a command line argument.

2. Your program should create n + m threads where n and m provided by the user as command line arguments.

3. Oxygen and Hydrogen threads should print the following messages when **appropriate** (also **replace** the characters “i”, “k” and “n” with the **corresponding number**):

    - Oxygen **i**: No available hydrogen atoms, so I wait. There are other **k** oxygen atoms and **n** hydrogen atoms waiting.
    - Oxygen **i**: **k** oxygen atoms and **n** hydrogen atoms are waiting, so I signal the next oxygen and hydrogen atoms in the queue.
    - Hydrogen **i**: No available hydrogen or oxygen atoms, so I wait. There are other **k** oxygen atoms and **n** hydrogen atoms waiting.
    - Hydrogen **i**: **k** oxygen atoms and **n** hydrogen atoms are waiting, I signal the next oxygen and hydrogen atoms in the queue.
    - Oxygen **i**: We are bonding together now.
    - Hydrogen **i**: We are bonding together now.
    - Oxygen **i**: I have bonded with two hydrogen atoms, and become a water molecule.

4. "bond()" line in the pseudo code should be replaced with usleep(d) where d is 500000.

5. Also add a random delay (using usleep() function) to the beginning of the thread functions. Delay duration should be between 250000 and 1000000. It should be different for each thread and in each run.

6. The threads must be running in parallel.

7. You will submit a single source code file to the ninova. Make sure that GNU C/C++ compiler (g++ or gcc) compiles your project, and the application runs in Linux smoothly. You can use ITU ssh server to compile and test your application. This is important because we will evaluate your homework in Unix using g++ or gcc.

The following commands can be used to compile and run the program:

```
$ gcc assignment2.c -o assignment2 -pthread -w
$ ./a.out 5 6
```

You can check following web-pages to learn about reading command-line arguments, converting strings to numbers and generating random numbers in C

- https://rosettacode.org/wiki/Command-line_arguments
- https://www.geeksforgeeks.org/converting-strings-numbers-cc/
- https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/