EE442 PROGRAMMING ASSIGNMENT 1

Chemical Reaction Simulation Using Pthread Library

Due: April, 30, 2023, 23:59

*** For your questions use forum or send an email to** ksert@metu.edu.tr.

Submission

- Send your homework compressed in an archive file with name "**eXXXXXXX_ee442_pa1.tar.gz**", where X's are your **7-digit student ID number**. You will **not** get full credit if you fail to submit your work as required.
- Your work will be graded on its correctness, efficiency, clarity and readability as a whole.
- Comments will be graded. You should insert comments in your source code at appropriate places without including any unnecessary detail.
- Late submissions are welcome, but are penalized according to the following policy:
  - 1 day late submission : HW will be evaluated out of 70.
  - 2 days late submission : HW will be evaluated out of 40.
  - Later submissions : HW will NOT be evaluated.
- The homework must be written in **C** (**not** in C++ or any other language).
- You should **not** call any external programs in your code.
- **Check** what you upload. Do not send corrupted, wrong files or unnecessary files.
- The homework is to be prepared **individually**. Group work is **not** allowed. Your code will be **checked** for cheating.
- The design should be your original work. However, if you partially make use of a code from the Web, give proper **reference** to the related website in your comments. Uncited use is unacceptable.
- **METU honor code is essential**. Do **not** share your code. Any kind of involvement in cheating will result in a **zero** grade, for **both** providers and receivers.

Part 1

1.1 Introduction

In the first part of this assignment, you are asked to simulate chemical reactions to form carbondioxide, nitrogen dioxide, sulfur dioxide and thorium dioxide. Due to synchronization problems it is hard to obtain these molecules properly. In order to obtain carbondioxide molecule, one C atom and two O atoms must get all together at the same time. Similarly, one N atom-two O atoms, one S atom-two O atoms, and one Th atom-two O atoms must get all together simultaneously to form nitrogen dioxide, sulfur dioxide and thorium dioxide molecules, respectively.

We need to think atoms as threads. When we get all the required atoms at the same time, the related chemical reaction happens. You are expected to *use only mutexes, locks and condition variables for synchronization.* You should avoid starvation, busy waiting etc.

In order to change the parameters of the program, command-line options should be used in this assignment.

1.2 Atoms

Atoms are represented by the following struct:

```
struct atom {
        int atomID;
        char atomTYPE; // C, N, S or Th
}
```

The main thread will generate M_C carbon atoms, M_N nitrogen atoms, M_S sulfur atoms, M_{TH} thorium atoms, and M_O oxygen atoms in total. M_C, M_N, M_S, M_{TH}, and M_O is selected with –c, -n, -s, -th and -o options, respectively. After each generation, the main thread will write to the terminal about the atom as follows:

"<atomTYPE> with ID: <atomID> is created."

Each atom id must be one larger than the one before.

If the atom is not used, the main thread will write to the terminal about the atom as follows:

"<atomTYPE> with ID: <atomID> is wasted."

1.3 Atom Generation Rate

The main thread generates C, N, S, Th and O atoms randomly. The total number of generated atoms will be $M_C+M_N+M_S+M_{TH}+M_O$. Between each atom generation, the main thread should sleep for a random amount of time with an exponential distribution (the rate is selected with -g option standing for generation time).

From a continuous uniform distribution x between 0 and 1, the exponential distribution can be generated as follows:

$$f(y,\lambda) = -\frac{1}{\lambda}\ln(1-x)$$

where λ is the rate.

1.4 Chemical Reactions

In order to compose the molecules you have only 4 composer threads. Composer_CO2 thread generates carbondioxide CO_2 molecule and must wait until there are two O atoms and one C atom available. Similarly, Composer_NO2, Composer_SO2 and Composer_TH02 threads generate other molecules.

When all the required atoms to make a molecule are available, chemical reaction will happen and then the related composer thread will update the information variable. For example; Composer_SO2 thread generates SO_2 molecule when one S and two O atoms are available, and then updates the information variable.

1.5 Order of Molecule Composition

Molecules must be created in the following order:

CO_2 – NO_2 – CO_2 – SO_2 – ThO_2

You are expected to *use only mutexes, locks and condition variables for synchronization* to compose the molecules in this order. You should avoid starvation, busy waiting etc.

1.6 Information Variable

Every time Information variable is updated, main thread will write to the terminal the following:

"<moleculeTYPE> is composed in tube <tubeID>."

moleculeTYPE is CO2, NO2, SO2 or THO2.

There must be single Information variable in the program. All composer threads will update this variable.

1.7 Command-line arguments

The program should use five optional arguments to change parameters:

- -c: The total number of carbon atoms to be generated (default 20)
- -n: The total number of nitrogen atoms to be generated (default 20)
- -s: The total number of sulfur atoms to be generated (default 20)
- -t: The total number of thorium atoms to be generated (default 20)
- -o: The total number of oxygen atoms to be generated (default 20)
- -g: The rate of generation time (default 100)

Instead of parsing the command-line arguments directly, you may want to use `getopt()`.

1.8 POSIX thread (pthread) library

You may find useful information about pthread library (thread create, thread synchronization etc.) and examples in the following web pages.

- https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html#SYNCHRONIZATION

- https://www.educative.io/answers/how-to-create-a-simple-thread-in-c
- https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-pthread-create-create-thread
- https://pubs.opengroup.org/onlinepubs/9699919799/

1.9 Pthread Mutex

An example code to show how pthread mutexes are used for thread synchronization is given below.

```
int count = 0;                          /* shared count variable */
pthread_mutex_t mutex;                  /* pthread mutex */

int main()
{
        ……
        ……    /* main code */
        ……
}

/* Each thread executes this function. */
void * countFunction(void *arg)
{
        int i;
        for (i = 0; i < 5; i++)
        {
                /* Enter critical section. Acquire mutex lock. */
                Pthread_mutex_lock(&mutex);
                count++;
                /* Exit critical section. Release mutex lock. */
                Pthread_mutex_unlock(&mutex);
        }
        return NULL;
}
```

Part 2

2.1 Introduction

In the second part of this assignment, you are asked to simulate the same chemical reactions, but you are expected to *use only semaphores for this synchronization problem*. In order to change the parameters of the program, command-line options are used again.

Produce_C(), Produce_N(), Produce_S(), Produce_TH() and Produce_O() threads generate C, N, S, Th and O atoms by increasing semaphores; C, N, S, Th and O by one, which are initially zero. Each of these threads will generate M atoms in total (M is multiple of 6 and selected with –m option). All types of atoms except for O atoms must be produced in the same number, and the number of O atoms must be twice that of the others. After each generation, the related thread will write to the terminal about the atom as follows, and sleeps for a random amount of time like in the first part.

<atomTYPE> with ID: <atomID> is created.

Each atomID must be one larger than the one before.

Composer_CO2, Composer_NO2, Composer_SO2 and Composer_THO2 threads generate the molecules and increments CO2, NO2, SO2 and THO2 semaphores by one, when all required atoms to

make a molecule are available. After each molecule generation, the related thread will update the information variable. Every time Information variable is updated, main thread will write to the terminal the following:

```
<moleculeTYPE> is composed.
```

2.2 Command-line arguments

The program should use two optional arguments to change parameters:

- -m: The total number of atoms to be generated (default 60)
- -g: The rate of generation time (default 100)

Hints

- If you have main.c, atom.c and atom.h as source files, you can compile your code with this command: `gcc -o molecule main.c atom.c –pthread`
- If you have only main.c as a source file, you can compile your code with this command:
  `gcc -o molecule main.c –pthread`
- Some libraries need to be linked explicitly. One example is math.h library where gcc needs -lm option.

Remarks

- There should be **no deadlock or starvation.**
- Synchronization variables should be used **only for critical sections.**
- There should be **no memory leak.**
- You can find information about headers, functions and types here.
- Name main files as main1.c and main2.c for part 1 and part 2, respectively.
- Send only the source code (`atom.c, atom.h, main.c etc.`). Do not send any executable, since your code will be recompiled.