EE442 PROGRAMMING ASSIGNMENT 1

Chemical Reaction Simulation Using Pthread Library

Due: April, 26, 2022, 23:55

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

Submission

- Send your homework compressed in an archive file with name "**eXXXXXXX_ee442_hw1.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 water, carbondioxyde, nitrogen dioxide and ammoniac. Due to synchronization problems it is hard to obtain these molecules properly. In order to obtain water molecule, two H atoms and one O atom must get all together at the same time. Similarly, one C atom-two O atoms, one N atom-two O atoms, and one N atom-three H atoms must get all together simultaneously to form carbondioxyde, nitrogen dioxide and ammoniac 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, but 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, H, O or N
}
```

The main thread will generate M_C carbon atoms, M_H hydrogen atoms, M_O oxygen atoms, and M_N nitrogen atoms in total. M_C, M_H, M_O, and M_N is selected with –c, -h, -o, and -n 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.

1.2.1 Atom Generation Rate

The main thread generates C, H, O and N atoms randomly. The total number of generated atoms will be $M_C+M_H+M_O+M_N$. 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.

1.3 Chemical Reactions

Each C, H, O and N atom invokes a procedure C(), H(), O() and N() when it's ready to react, respectively. The procedures must wait until there are two H atoms and one O atom available to create water (H_2O) molecule.

Similarly, we need one C - two O atoms, one N - two O atoms, and one N - three H atoms to create carbondioxyde (CO_2), nitrogen dioxide (NO_2), and ammoniac (NH_3) molecules, respectively.

There are three test tubes in which chemical reactions takes place. Test tubes are represented by the following struct:

```
struct tube {
	int tubeID;
	int tubeTS; // time stamp (ID of the atom spilled first)
	int moleculeTYPE; // 1:H2O, 2:CO2, 3: NO2, 4: NH3
}
```

When an atom is created, it will be;

- spilled to the first empty test tube (if all test tubes are empty).
- spilled to the proper* tube with the smallest time stamp.
- wasted (if there is no proper* tube).

*You must check proper chemical reaction conditions. For example, if the tube contains C atom, it only reacts with O atoms.

When all required atoms to make a molecule are available, chemical reaction will happen and then the tube will be emptied. After the removal, it will update the information represented by the following struct:

```
struct Information {
	int tubeID;
	struct mytube tube;
}
```

There must be a single Information variable in the program. All test tube threads will update this variable.

1.4 Information Variable

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

```
<moleculeTYPE> is created in tube <tubeID>.
```

1.5 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)
- -h: The total number of hydrogen atoms to be generated (default 20)
- -o: The total number of oxygen atoms to be generated (default 20)
- -n: The total number of nitrogen 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()`.

1.6 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;
}
```

1.7 Example Output for Part 1

```
ee442@ee442-VirtualBox: ~/HW1$ ./molecule -c 5 -h 10 -o 10 -n 5 -g 100
GENERATION_RATE: 100.000000
O with ID:1 is created.
C with ID:2 is created.
H with ID:3 is created.
O with ID:4 is created.
CO2 is created in tube 1.
O with ID:5 is created.
H with ID:6 is created.
H2O is created in tube 2.
N with ID:7 is created.
C with ID:8 is created.
H with ID:9 is created.
H with ID:10 is created.
O with ID:11 is created.
O with ID:12 is created.
CO2 is created in tube 2.
N with ID:13 is created.
C with ID:14 is created.
H with ID:15 is created.
NH3 is created in tube 1.
O with ID:16 is created.
O with ID:17 is created.
NO2 is created in tube 2.
N with ID:18 is created.
H with ID:19 is created.
H with ID:20 is created.
O with ID:21 is created.
N with ID:22 is created.
C with ID:23 is created.
C with ID:23 wasted.
O with ID:24 is created.
CO2 is created in tube 3.
H with ID:25 is created.
NH3 is created in tube 1.
C with ID:26 is created.
H with ID:27 is created.
H with ID:28 is created.
O with ID:29 is created.
N with ID:30 is created.
```

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_O(), Produce_H(), Produce_C() and Produce_N() threads generate O, H, C and N atoms by increasing semaphores; O, H, C and N by one, which are initially zero. Each of these threads will generate M atoms in total (M is multiple of 4 and selected with –m option). All types of atoms must be produced in the same number. 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 `atom` id must be one larger than the one before.

In this part test tubes will not be implemented. Composer_H2O, Composer_CO2, Composer_NO2 and Composer_NH3 threads generate the molecules and increments H2O, CO2, NO2 and NH3 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 generated.
```

2.2 Command-line arguments

The program should use five optional arguments to change parameters:

- -m: The total number of atoms to be generated (default 40)
- -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.