EE 463 | **Winter 2023** | **Dr. A. M. Al-Qasimi**

Operating Systems

Homework #5

Date: 01/07/1444
23/01/2023
Due: 31/01/2023

Objectives:

The purpose of this homework is to exercise the use of Pthreads under Linux to create, manage and synchronize the actions of threads in C. In particular, you are asked to use thread synchronization tools like, Mutex locks, Condition variables, Semaphores, … etc. to solve a typical synchronization and mutual exclusion problem, being aware to prevent race conditions, deadlock and starvation possibilities.

Description:

Write a multithreaded C program for patient room access control in a hospital. The controller must obey the following simple rules:

1. Doctors can enter the room one at a time, and can enter only if no visitors are in the room.
2. A visitor can only enter the room if no doctors are in the room and if a maximum of 2 other visitors are in the room (three in total).
3. When a doctor arrives, he must wait for all the current visitors in the room to leave the room.
4. When a doctor arrives, no more visitors are allowed in, until a waiting doctor enters and leaves the room.
5. When a doctor leaves the room, only one waiting doctor or up to three waiting visitors can enter the room, depending on who comes first.

You shall do this by creating and synchronizing the following threads:

1. A random number of threads, no more than 50 at any given time, each representing a Visitor wishing to visit the patient in his room.
2. A random number of threads, no more than 10 at any given time, each representing a Doctor wishing to see the patient in his room.
3. One display thread which monitors the room status and periodically prints out the current number of doctors and visitors in the patient room, together with their IDs.

Requirements:

- Write your application program in C to create and manage the required threads and provide both synchronization and mutual exclusion as necessary.
- Implement mutual exclusion where appropriate.
- Avoid busy-waiting, starvation and deadlocks.
- Each doctor and visitor thread must pause for a fixed period of time before entering the room. Upon finishing the pause period, it must print a line with its ID, and the indication it wishes to enter.
- When a thread is allowed into the room, it must print a line with its ID, and that it is actually inside.
- Each doctor and visitor thread must pause for a fixed period of time before exiting the room. Upon exiting, it must print a line with its ID, and the indication it has left the room.
- The display thread shall continue working until it is interrupted by the user through a Ctrl-C signal.
- When, the display thread is interrupted, it must send cancellation signals to all the other threads and wait for their termination, clean up, and then terminate itself.

Note:

The pausing periods for all doctors and visitors must be fixed and equal, e.g. 1 second.
The output from all the threads must clearly show that your controller is working as specified above. Study the output of your program carefully to check it is operating and synchronizing properly, without any deadlock or starvation. A sample output follows:

```
Display -> Created 3 Doctors, and 15 Visitors.
Doctor, D1 wishes to see the patient, waiting.
Doctor, D1 entered the patient’s room, waited for 0 ms.
Visitor V1 wishes to visit the patient, waiting.
Doctor, D1 exited the patient’s room, stayed 42 ms.
Visitor V1 entered the patient’s room, waited for 45 ms.
Visitor V3 wishes to visit the patient, waiting.
Visitor V3 entered the patient’s room, waited for 0 ms.
Display -> In the room: 0 doctor [ ], and 2 visitors [ V1, V3 ].
Visitor V3 exited the patient’s room, stayed 10 ms.
```

What to turn in:

Submit one flat .zip file, named as your first initial and last name without spaces, containing:

a. Your program design document in .pdf form.
b. The fully documented, commented and working program, in source .c, .h file(s) only.
c. A meaningful sample of your program’s output in .pdf form.
d. A statement report of your observations and comments about the program output in .pdf.