Operating Systems Laboratory (CS39002)

Spring Semester 2021-2022

Assignment 5: Usage of Semaphores to synchronize between threads

Assignment given on: Mar 15, 2023

Assignment deadline: Mar 22, 2023, 10:00 am

In this assignment you will learn how to use semaphores. Semaphores are more versatile than mutex locks—they can allow multiple threads access critical sessions whereas a mutex lock only allows one thread at a time. With this background implement the following setup.

There is a hotel with N rooms (each room can be one data structure instance). Each room can have at max of 1 guest. The hotel also has X cleaning staff (C_1, C_2, C_3, … C_X) and is visited regularly by Y guests (G_1, G_2, G_3, … G_Y).
These Y guests all have different priorities which are randomly allocated. The initial priority order of the guests is $G_1 > G_2 > G_3 > \ldots > G_Y$. (Higher priority means more important). There will be X threads modeling the behavior of cleaning staff and Y threads modeling the behavior of guests.

Here are the functionalities of three main components in this problem:

- **Main thread** - Creates the guest and cleaning staff threads. Initialize semaphores which will control access to rooms for guests and cleaning threads.

- **Guest thread** - Sleeps for a random time between 10-20 seconds . After waking up, requests for a room in the hotel for a time to stay between 10-30 seconds. A guest thread G is allotted a room if either of the following conditions are met.
    a) The number of guests currently in the hotel < N. No guest is removed in this case and G will get an empty room.
    b) The number of guests currently in the hotel equals N. However, there exists at least 1 guest (H) with lower priority than G. In this case H is removed from the hotel and G takes the place of H.

  Naturally in either of the cases, the room allotted to G must have had at most 1 previous occupant.

  After a guest is allotted a room, then it sleeps for the duration of stay. A guest who is removed from the hotel first sleeps and then after waking up again tries to get a room.. The guest threads repeat the above steps indefinitely.

  Eventually all the rooms will have at least had 2 occupants and no room can be allotted further. This means that the hotel needs to be cleaned.

- **Cleaning Staff thread** - Wakes up when required (all rooms are occupied 2 times). Selects a random room which was occupied since the last time the hotel

was cleaned. The amount of time to clean a room is proportional to the time it was occupied by previous guests. The cleaning thread sleeps for the duration of cleaning a room and then marks the room as clean. This is done till all the rooms are cleaned.

NOTES

- *While the hotel is being cleaned, guests can still request a room. You will need to handle these requests (basically guests need to wait till all rooms are cleaned). After the hotel is cleaned the guests who were removed are added back to their previously allotted rooms.*
- **Constraint** - Y > N > X > 1.
- In your program you need to take X, Y and N as input.
- The priority of guests will be a random integer number between 1 and Y (both inclusive).

Submission Guideline:

- Create three programs (in C or CPP), one for the main thread, one for the guest, and one for the cleaning staff. Also create a Makefile that will compile all programs.
- Include a short report DESIGNDOC, as a pdf file. it should describe what is your design for the three threads, what data structures did you use and what semaphores did you use on them.
- Zip all the files into Ass5_<groupno>_<roll no. 1>_< roll no. 2>_< roll no. 3>_< roll no. 4>.zip (replace <groupno> and <roll no.> by your group number and roll numbers), and upload it on Moodle.
- You must show the running version of the program(s) to your assigned TA during the lab hours.
- **[IMPORTANT] Please make only one submission per group.**

Evaluation Guidelines:

Total marks for this assignment: 45

We will check features of your code (including but not limited to) if the computation is correct, you are creating and using the threads correctly, there is no race condition, your code is not mostly sequential and your log contains meaningful and logically valid messages.

Items	Marks
Correctness of the main thread	5
Guest threads	15
Cleaning staff threads	15
DESIGN DOC (data structure, usage of semaphores, overall design)	10
Total	**45**