Indian Institute of Technology, Kharagpur
Department of Computer Science and Engineering

CS39002 : OPERATING SYSTEMS LAB

ASSIGNMENT 5: USAGE OF SEMAPHORES TO SYNCHRONIZE BETWEEN THREADS

24th March, 2023

Group Number : 24

Shivam Raj
20CS10056

Jatin Gupta
20CS10087

Kushaz Sehgal
20CS30030

Rushil Venkateswar
20CS30045

Contents

1. Data Structures **2**
1.1 Description of the Room struct 2
1.2 Other global variables used 2

2. The Main Thread **3**

3. The Guest Thread **3**

4. The Cleaning Staff Thread **3**

5. Usage of semaphore and mutexes in the Assignment **3**

1. Data Structures

1.1 Description of the Room struct

Data members :

- int **current_guest** : the guest currently residing in the room with two special values:
  - EMPTY (-1) : Indicating that the room is empty and has been occupied once
  - DIRTY (-2) : Indicating that the room is empty and has been occupied twice
- sem_t **room_occupancy** : the semaphore initialized to 2, so that only a maximum of two guests can occupy the room before it needs to be cleaned
- bool cleaned : boolean indicating whether or not the room has been cleaned
- int total_time : Time since last cleaning of the room

1.2 Other global variables used

- int n, x, y : Number of rooms, cleaning staff members and guests respectively
- int *priority : stores the priority of each guest
- Room *rooms : stores the rooms
- bool is_cleaning : flag used to denote whether or not cleaner threads should be active. It is periodically checked in the main thread and when set to true, it sends a pthread_cond_signal to all cleaning_staff threads to become active
- vector<int> rooms_to_clean : Vector storing the ids of the rooms which need to be cleaned

Mutexes used :

- pthread_mutex_t *guest_mutexes : Mutex locks pertaining to each guest thread
- pthread_mutex_t *cleaning_mutexes : Mutex locks pertaining to each cleaning staff thread
- pthread_mutex_t rooms_to_clean_mutex : lock applied on the rooms_to_clean vector

Condition variables :

- pthread_cond_t *guest_conds : Condition variables pertaining to the guest threads. This variable allows us to wait on the signal that guest threads should become active once cleaners are done cleaning
- pthread_cond_t *cleaning_conds : Condition variables pertaining to the cleaning staff threads. This variable allows us to wait on the signal that cleaning staff threads should become active once guests have occupied all the rooms exactly twice

2. The Main Thread

Takes n, x, y as input and initialises the mutexes, condition variables and threads to the required size using malloc. Initialises the priority vector and all rooms. Finally creates the threads and goes into an infinite loop where it monitors whether cleaners should be running or guests.
If is_cleaning is false currently, the main thread checks whether each room is empty and the semaphore is also locked. In the case that this condition is true for all rooms, we know that all rooms are now dirty and cleaners should be called. A signal is sent to all the cleaning threads and the rooms are now reset.
If is_cleaning is true currently, the main thread checks whether each room is cleaned and if that is true, then we know that all rooms are clean and signals should be sent to the guest threads.

3. The Guest Thread

Sleeps for a random time initially, and then waits for a signal on the guest_conds conditional variables.
Once that signal is received, each room is checked to see whether or not it is currently empty and the semaphore can be locked, and if true we call the check_in function to simulate the action of checking in to the hotel.
If all rooms have one guest currently and there is one more guest to be checked in to the hotel, then the first guest with lower priority than current guest is kicked out from the hotel. To do this, the check_in function is called on the same room that the kicked out guest was occupying.
The guest thread also keeps track of how long each room has been used since it was last cleaned.
In the check_in function, pthread_cond_timedwait is called which returns on two conditions :

- Either a signal is received to kick the guest out
- Or the timer runs out and the guest is supposed to vacate the hotel

In the first case the guest thread measures wall clock time and updates the total_time variable of the room.

4. The Cleaning Staff Thread

Waits for a signal on the cleaning_conds conditional variables.
Once that signal is received, the vector rooms_to_clean is checked to see if there are any rooms needing to be cleaned. A room is selected at random from this vertex and removed from the vector.
Now, cleaning begins and the cleaner thread sleeps for an amount of time proportional to the time for which the room was occupied. We mark the room as cleaned now and reset the total_time variable.

5. Usage of semaphore and mutexes in the Assignment

We use a semaphore called room_occupancy with the initial value of 2 which prevents any room from being occupied more than twice. This semaphore is locked when a guest resides in the room and unlocked when cleaners are called.
Mutexes/Condition Variables are used to signal the guest thread or the cleaning staff thread to become active. Apart from these, the guest_conds condition variables are also used to perform a timedwait on the guest (simulating the act of staying in a room for a fixed time or being kicked out by a higher priority guest on receiving a signal).