Operating Systems Laboratory (CS39002)

Assignment - 5

Usage of Semaphores

Group - 15
Utsav Mehta - 20CS10069
Sidharth Vishwakarma - 20CS10082
Pranav Kulkarni - 20CS30029
Swapnil Yasasvi - 20CS30054

Data Structures Used

Room

Members:

1. `room_id (int)`: unique id for each room.
2. `guest_id(int)`: unique id for each guest who is in the room.
3. `guest_count(int)`: to keep track of the number of guests who have already used the room, thus, after 2 guests have stayed in the room, we activate cleaner threads.
4. `stay_time(pair<int,int>)`: **stay_time.first** stored the stay duration for the first guest who has entered the room after it has been cleaned or during the start of the program and **stay_time.second** stores the stay duration for the second guest who has entered the room after the first guest has left the room.

Global Variable

1. `stay_count`: To indicate the number of times all rooms used by the guests , when stay_count become equal to 2*N i.e when each of the rooms has been used twice then the cleaner threads will be activated by unblocking the clean_start semaphore by setting it to 1 such that the cleaner threads start cleaning.

Semaphores Used:

N - Number of Rooms

1. `bin_room_sems[N] (binary semaphore)`: To lock the rooms so that only one guest can access the room at a time.
2. `sig_room_sems[N] (semaphore-Init value 0)`: Used for blocking the room when used by some guest and free it when guest leaves the room or a higher priority guest has to be allocated the room.
3. `cleaning (counting semaphore)`: To block guest threads and facilitate the functioning of cleaner threads till all rooms are cleaned.
4. `stay_count_sem (binary semaphore)`: Binary semaphore to lock stay_count variable while incrementing by one guest thread.
5. `clean_start (binary semaphore)`: To indicate the cleaner threads to start cleaning the room.
6. `clean_end (binary semaphore)`: To indicate that cleaner threads have finished cleaning the room thus allowing guests to enter the room, by signaling the bin_room_sems semaphores.

Implementation of threads using semaphores

1. All `bin_room_sems` are initialized to 1, after a room 'i' is allocated to a guest the guest count is increased and the wait by `bin_room_sem[i]` consumers the value of the semaphore, thus making it 0, so now no other guest could access this room till the `room id = -1` (empty).
2. `All sig_room_sems` are initialized to 0 so when each guest thread will be using the room, it will block till the sleeping time or when some other guest sends a signal to unblock the timedwait() call, by setting the value of semaphore to 1.
3. `stay_count_sem` is used to lock the `stay_count` for increment, when each room has been used 2 times, the `stay_count` becomes 2*N , so now using an if condition in this critical section locked by semaphore `stay_count_sem` , we make the binary semaphore `clean_start` , `clean_end` set in order to activate the cleaner threads. Also setting the value of `cleaning` semaphore to N.
4. `cleaning` - This counting semaphore has been incremented to N by the guest threads after the room has been used twice each so now `sem_wait(&cleaning)` will gets unblocked for N times in order to facilitate cleaning for N rooms, once one of the cleaner gets unblocked and `clean_start` is set to 1 then we do a `sem_post` on `sig_room_sems[i]` for each occupied room 'i' such that in case a room is currently occupied by a guest who is in sleep then he has to leave the room when the cleaning starts.
5. Once all the rooms are cleaned, a `sem_trywait` on `clean_end` is made. After the cleaning is done we unlock all the rooms by signaling to `bin_room_sems` and thus all guest threads start acquiring the rooms.