Operating Systems Laboratory (CS39002) -- Assignment 5

Group - 15
Utsav Mehta - 20CS10069
Siddharth Viswkarma - 20CS10082
Pranav Kulkarni - 20CS30029
Swapnil Yasasvi - 20CS30054

Data Structures USed:
1. Room
Members:
->int room_id
unique id for each room.

->int guest_id
unique id for each guest who is in the room.

->int guest_count
to keep track of number of guest who have already used the room so after 2 guest we need to activate cleaner to clean the room.

-> pair<int,int> stay_time
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.

Priority Heuristic:

We are randomly alloting unique priority to each guest between 1 to Y(Number of guests).

Global Variable:
->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

Used Semaphores:
->bin_room_sems[N] :
To lock the rooms so that only one guest can access the room at a time.

->sig_room_sems[N] :
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.

->cleaning :
To block guest threads and facilitate the functioning of cleaner threads till all rooms are cleaned.

->stay_count_sem :
Binary semaphore to lock stay_count varible while incrementing by one guest thread.

->clean_start :

To indicate that cleaner threads has started cleaning the room and no guest can
enter the room.

->clean_end : To indicate that cleaner threads has finished cleaning the room an
d guest can enter the room.

Semaphores Application

1.All bin_room_sems are initially initialized to 1 , after a room 'i' is alloted
 to a guest the guest count is increased and the wait by bin_room_sem[i] becomes
 unblocking and bin_room_sem[i] becomes 0 so now no other guest could access thi
s room till the room id = -1 (empty).

2.sig_room_sems all are intially intialized to 0 so when each guest thread will
be using the room till the sig_room_sem[i] for a room 'i' is et to 1 either when
 a more priority guest comes and also the guest count for that room is leass tha
n 2 or when the guest currently in the room leaves the room after some given sle
ep time.

3.stay_count_sem is used to lock the stay_count for increment , now when each ro
om has been used 2 times each that the stay_count has become 2*N , so now using
a if conditon in this critical section locked by semaphore stay_count_sem , we m
ake the binary semaphore clean_start , clean_end set in order to activate the c
leaner threads. Also within the stay_count_sem Semaphores when the above if (sta
y_count == 2*N) conditon is satisfied then set the cleaning counting semaphore
to N.

4.cleaning so this counting semaphore has been incremented to N by the guest thr
eads after the room has been used twice each so now sem_wait(&cleaning) will get
s unblocked for N times in order to facilitate cleaning for N rooms , now if it
gets unblocked then if clean_start is set to 1 then we do a sem_post on sig_room
_sems[i] for each room 'i' such that in case a room is currently occupied by a g
uest who is in sleep then he has to leave the room when the cleaning starts. Als
o we make the clean_start to 0 so that no other guest could enter the room till
the cleaning is done. Now we do a sem_wait on clean_end so that the cleaner thre
ad could wait till the cleaning is done for all the N rooms. After the cleaning
is done we make the clean_end to 0 so that the guest threads could enter the roo
m again and we make the cleaning to 0 so that the cleaner threads could wait for
 the cleaning to be done again for N rooms.

5. After all rooms have been cleaned we will just check if all_clean is true and
 here the blocking call by clean_end binary semaphore will become unblocking as
clean_send was set so now we will just make all the bin_room_sems[i] for each ro
om[i] to be set such that the rooms could be occupied again now.