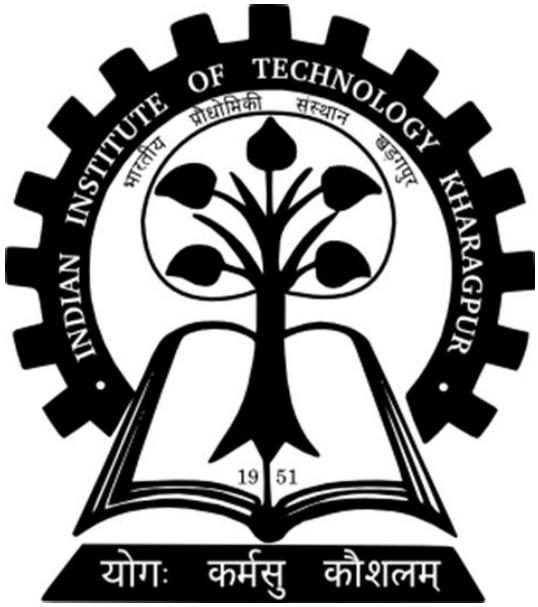
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



LABORATORY ASSIGNMENT 5

(Operating Systems Laboratory)

GROUP 31

Likhith Reddy Moreddigari 20CS10037

Shivansh Shukla 20CS10057

Venkata Sai Suvvari 20CS10067

Boorgu Shashank Goud 20CS30013

March 21, 2023

Data Structures Used

- **Room:** This is the data structure, which stores the information about room viz., number of times it is used after it is cleaned, total time used after cleaning, priority of the guest currently residing in the room, id the guest currently using.

```
class Room
{
public:
    int guests_used; // number of guests used
    int time_used; // total time used
    int priority; // priority of the guest staying in the room
    pthread_t guest_thread; // thread id of the guest staying in the room
    Room(int priority = -1, int guests_used = 0, int time_used = 0) : guests_used(guests_used
    ), time_used(time_used), priority(priority) {}
};
```

- **freeRooms:** This is a queue which stores <Room*>. Those rooms are stores which are free(either used 0 times, or once already)
- **occupiedRooms:** This is a set of pairs storing <priority, Room*>. This is used so as to take the Room which has guest whose priority is just less than the priority of guest we have using lower_bound. Here priority queue is not used because, suppose, there are 4 rooms filled with guests having priorities 3, 5, 7, 9 and now guest with priority 8 comes. It's ideal to replace 7 as if the next guest is 4, and we replace 3 instead of 7 in previous case where 8 came, the guest with priority 4 needs to wait.
- **doneRooms:** This is a queue which stores <Room*>. This contains all the Rooms which were used by 2 guests.

Semaphores and Usage

- **cleaner_sem:** This is used for the cleaners to clean the room and also for maintaining the mutual exclusion between the guests and cleaners.
- **room_sem:** This is initialised to 1. When a guest wants to enter into a room, waits on this semaphore then find a free room if available or gets a room which is occupied by a lower priority guest and then posts this semaphore then goes to sleep.

Threads functionalities

- **main thread:** This takes input for the number of cleaners, rooms and guests, then creates the threads for cleaners with same count as the number of cleaners and creates guest with same count as the number of guests. For guests, it initializes each guest thread randomly generated priority. Finally it waits for the all the threads to complete their processes. This initializes the semaphores in such a way that the guest will enter first i.e., the room_sem is initialized with 1 and cleaner_sem is initialized with 0 so the guest will acquire the access to the rooms first.
- **cleaner thread:** This thread first wait on a cleaner_sem and since it is initilalised to 0 at start, all the cleaner threads will be waiting on it. When it becomes 1, one of the cleaner threads pops a room from the doneRooms data structure and then posts cleaner_sem and then goes to sleep the amount of time the room caries(which is same as the total time guests slept in the room). If doneRoom is empty then it will post room_sem and makes it's value 1(how it became zero is in guest thread section) and then again waits on cleaner_sem.
- **guest thread:** This thread first goes into a loop, then sleeps for a random time. When it wakes up, goes and waits on room_sem(which was initialised to 1). Then it finds if a room is empty. If empty then pushes it to occupiedRooms(if number of times used is == 1) set setting correct values for the Room object and posts room_sem. If all rooms are occupied then uses lower bound to check the room which has a guest with priority < the guest. If no room is found it goes to sem_wait(room_sem) back. If room is found, it sends a signal

to that thread using pthread_kill and sends a SIGUSR1 signal and evicts the other guest out and posts the room_sem. Now the guest will use sigtimedwait(masking all the other signals other than SIGUSR1) to sleep for a random time and then waits again on room_sem(because of this the room_sem value is 0 in cleaner thread above). It then pushes the room into doneRooms if the room is used by 2 guests, else pushes to freeRooms queue. Then doneRoom.size() is checked and if it is equal to n, sem_post on cleaner_sem is done which makes it's value 1, and the cleaners in cleaner thread can use this to clean rooms. If the doneRooms.size() is not n then sem_post on room_sem is done to facilitate other guests to enter.