Sleeping Barber problem

Solution pseudocode :-

int counts = 0 //number of customers;

mutex = Semaphore(1);

customer = Semaphore(0);

barber = Semaphore(0);

void barber (void){

wait(customer);

signal(barber);

cutHair();

}

void customer (void){

wait(mutex);

if (counts==n+1) {

signal(mutex);

leave();

}

counts +=1;

signal(mutex);

signal(customer);

wait(barber);

getHairCut();

wait(mutex);

counts -=1;

signal(mutex);

}

The sleeping barber problem solution given in this code is a multithreading scenario which ensures that there is no deadlock and any starvation and provides an efficient solution to the problem. Firs the user will input some initial values for the program to start which are:

Number of barbers

Number of waiting chairs

Number of customer

Once these values are inserted the simulation will start. Initially, all barbers are sleeping as the queue is null, once the first customer arrives in the waiting room, he will directly go to the nearest barber and will start getting his haircut, same goes for the customers arriving after that, once the chairs in waiting area are full the customers will leave without getting the haircut.

From barber’s perspective, each barber will check the queue and if there are any customers in the queue then barber will start with the first customer present in the queue and if there are no customers then barbers will sleep.

Deadlock :-

deadlock occurs when two or more threads wait forever for a lock

For this scenario, the deadlock will occur if the customer ends up waiting for the barber and the barber ends up waiting for the customer to arrive

To handle this problem I used Semaphore and after a thread acquires a lock it sleeps for few seconds and then release the lock.

The code ensures that each thread releases the lock after performing the critical section.

I have also used try-catch blocks to handle exceptions. Using reentrant locks, the critical section which is inside the locks can only be accessed by one thread at a time

Starvation :

For this scenario problem of starvation will occur if the Customer don’t follow any order for getting a haircut , as some won’t get a haircut even though even after waiting for a long time.

To handle this problem in my code I have inserted the customer (Processor) in a linked list which follows the first in first out property. So, every time a patient (Processor) sits in a waiting room, they will be selected by the Doctor in first come first serve basis. We could have also used other data structures like a stack, but the linked list seems like the best choice for this scenario.

Explanation for real world application and how did apply the problem .

Explanation for real world application is the clinic.

I have numbers of doctors and number of patients and number of chairs each patient asks one doctor and There is no two patient ask the doctor at the same time or there is no patient after asking the doctor to stay in the clinic forever .