Department of Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY
UNIVERSITY OF LUCKNOW
LUCKNOW

Dr. Zeeshan Ali Siddiqui
Assistant Professor
Deptt. of C.S.E.

CLASSICAL PROBLEMS OF SYNCHRONIZATION

(**Sleeping Barber Problem**)

Sleeping Barber Problem 1/3

- Consider one *barber*, one barber chair, and n chairs for waiting for customers, if there are any, to sit on the chair.

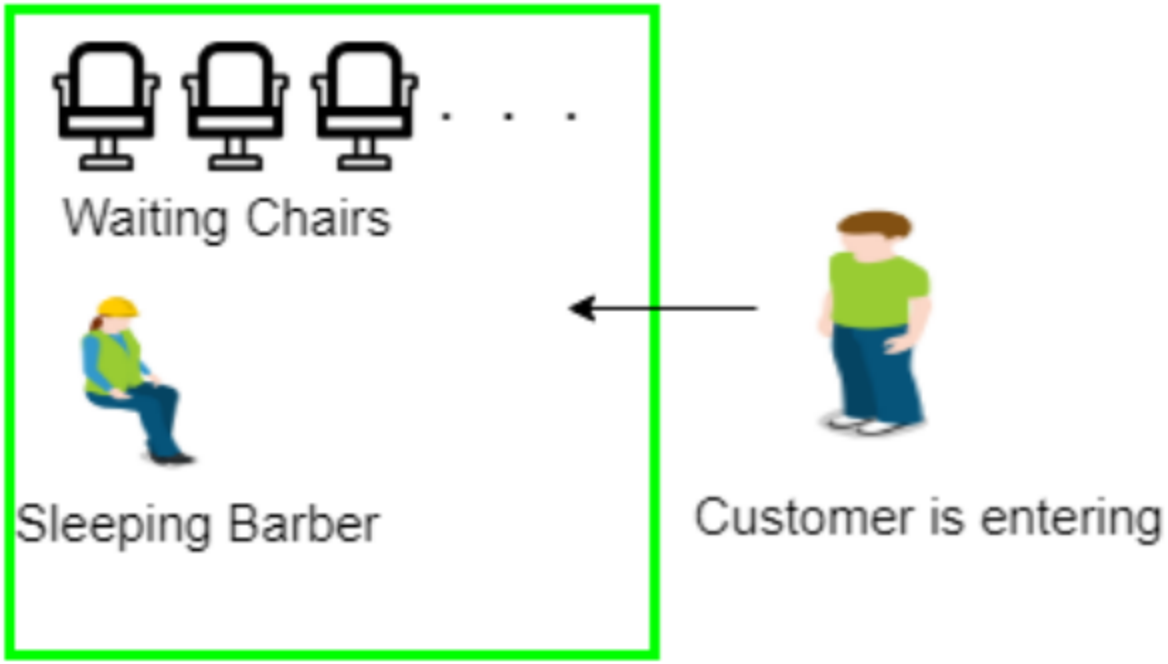
- **Rules**:

  - If there is no customer, then the barber *sleeps* in his own chair.

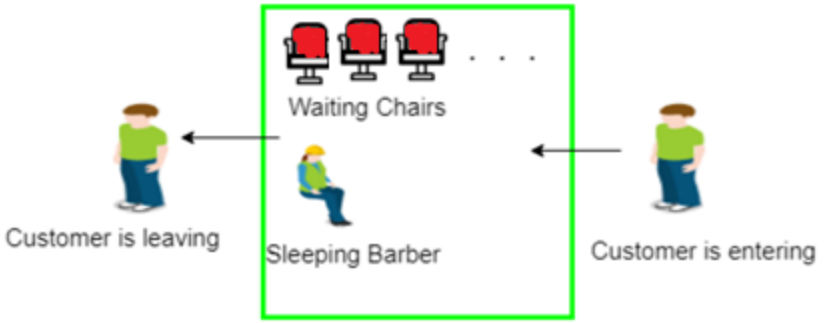
  - When a customer arrives, he has to *wake* up the barber.

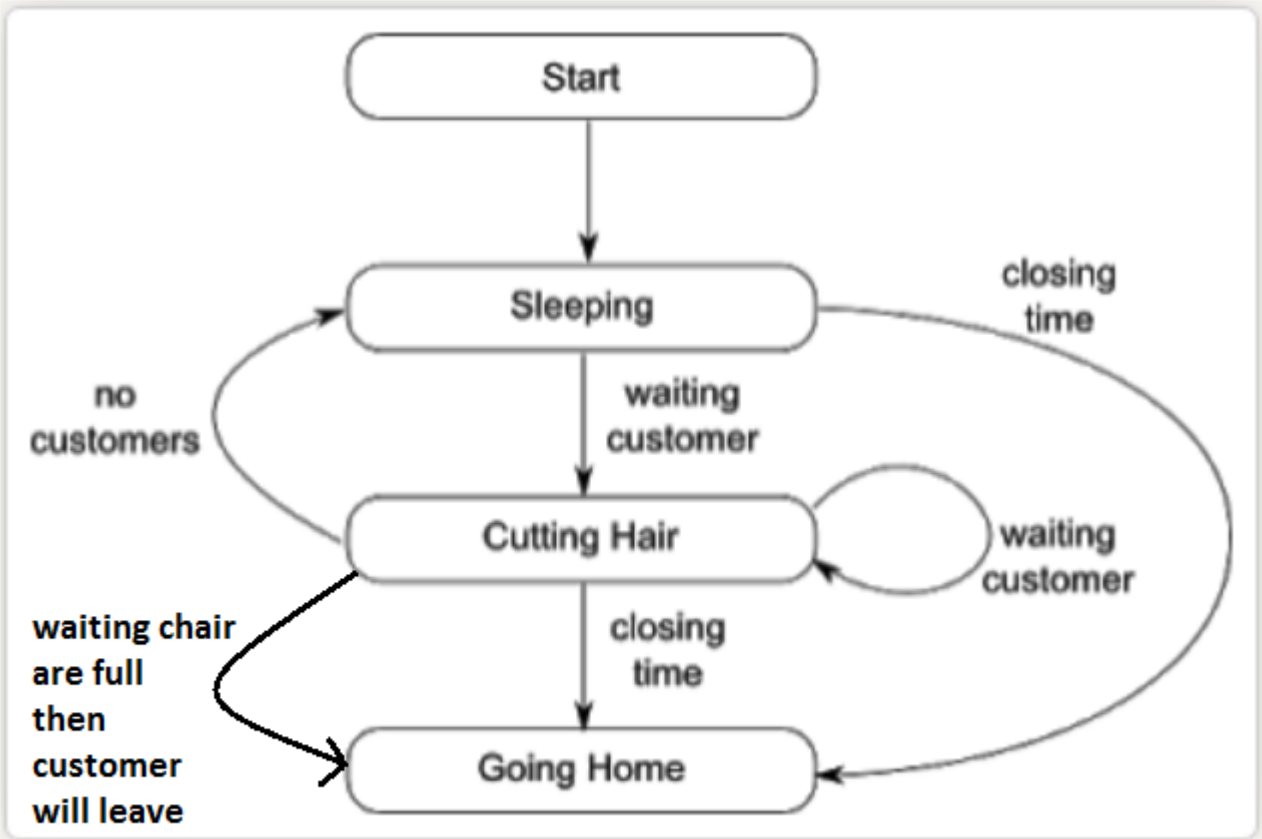
  - If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they *leave* if no chairs are empty.

Sleeping Barber Problem 2/3



Sleeping Barber Problem 3/3





Flow chart for Sleeping barber problem

Solution 1/3

- Three semaphores:

  - *Semaphore Customers:* It counts the number of customers present in the waiting room (customer in the barber chair is not included because he is not waiting).

  - *Semaphore Barber:* The barber 0 or 1 is used to tell whether the barber is idle or is working.

  - *Semaphore Mutex:* Mutex is used to provide the mutual exclusion which is required for the process to execute.

Solution 2/3

```
semaphore customers = 0;  /* # of customers waiting for service */
semaphore barber = 0; /* # of barbers waiting for customers */
semaphore mutex = 1; /* for mutual exclusion */
int Freeseats = N;
#define N 3

barber()
{
 while (true)
 {
    wait(customers); /* go to sleep if # of customers is 0 */
    wait(mutex); /* acquire access to "waiting' */
    FreeSeats++; /* a chair gets free */
    signal(barber); /* barber is now ready to cut hair */
    signal(mutex); /* release 'waiting' */
    cut_hair(); /* cut hair (outside critical region) */
 }
}
```

Solution 3/3

```
customer()
{
    wait(mutex); /* enter critical region */
    if (FreeSeats>0)
    {
        /* if there are no free chairs, leave */
        FreeSeats--; /* Sitting down */
        signal(customers); /* wake up barber if necessary */
        signal(mutex); /* release access to 'waiting' */
        wait(barber); /* go to sleep if # of free barbers is 0 */
        get_haircut(); /* be seated and be served */
    }
    else
    {
        signal(mutex); /* shop is full; do not wait */
    }
}
```

References

1. Modern Operating Systems 2nd Ed by Tanenbaum.

2. Silberschatz, Galvin and Gagne, “Operating Systems Concepts”, Wiley.

3. William Stallings, “Operating Systems: Internals and Design Principles”, 6th Edition, Pearson Education.

4. D M Dhamdhere, “Operating Systems: A Concept based Approach”, 2nd Edition, TMH.

5. https://www.geeksforgeeks.org/sleeping-barber-problem-in-process-synchronization/

Thank You.