CHAPTER 4.Synchronization

• Critical Section Problem:-

Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.

The critical section cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.

The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

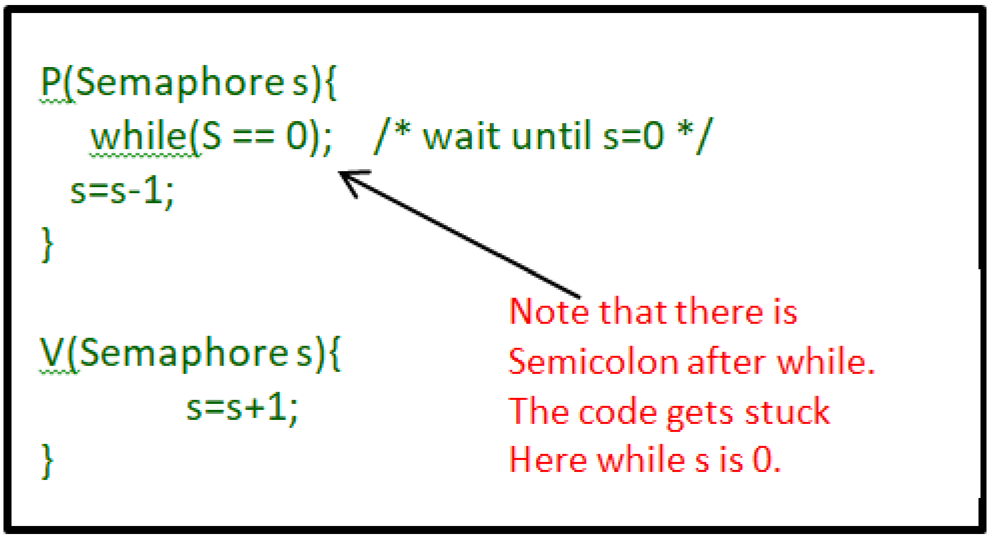
If you have a resource with three instances, the initial value of the Semaphore is 3(i.e., S=3). Whenever a process needs to access a critical section/resource, it calls the wait function and then decrements the semaphore value by one. The semaphore value is greater than 0.

Semaphour:-

The process of using Semaphores provides two operations: wait (P) and signal (V). The wait operation decrements the value of the semaphore, and the signal operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.
Semaphores are of two types:

1. **Binary Semaphore –**
   This is also known as a mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.

2. **Counting Semaphore –**
   Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances



```
P(Semaphore s){
    while(S == 0);   /* wait until s=0 */
   s=s-1;
}

V(Semaphore s){
          s=s+1;
}
```

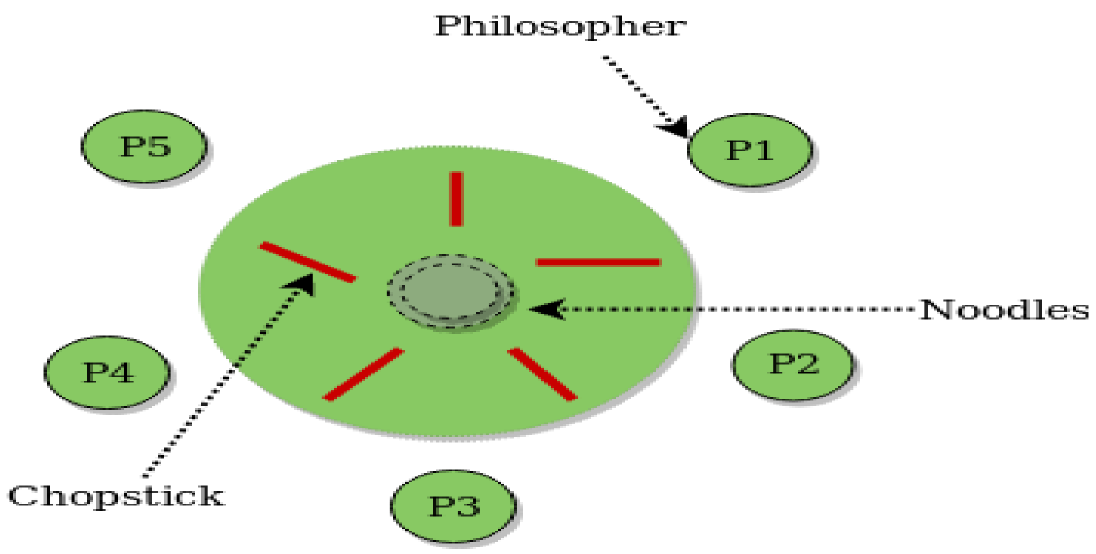
Note that there is
Semicolon after while.
The code gets stuck
Here while s is 0.

****** Classic Problems of Synchronization – The bounded buffer problem, The reader-writer problem, The dining philosopher problem:-**

The bounded buffer problem:- The Bounded Buffer problem is also called the producer-consumer problem. This problem is generalized in terms of the Producer-Consumer problem. The solution to this problem is, to create two counting semaphores "full" and "empty" to keep track of the current number of full and empty buffers respectively.
Producers produce a product and consumers consume the product, but both use of one of the containers each time

Dining-Philosophers Problem
The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pickup the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent

followers but not both. This problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.



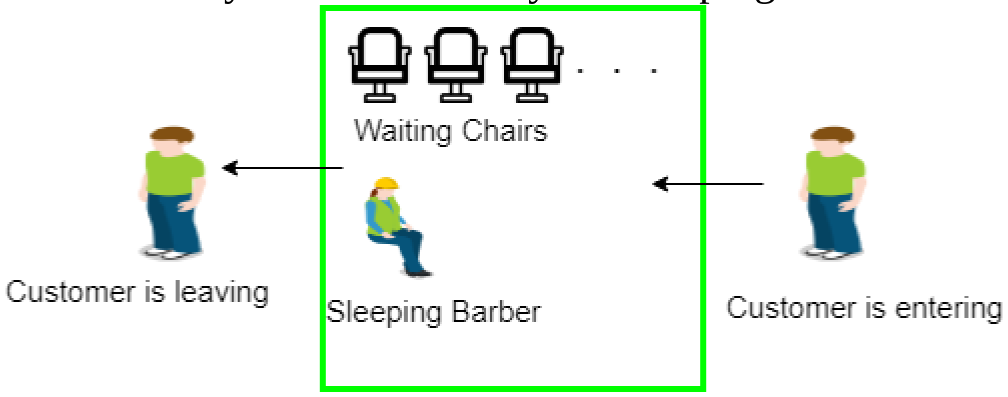
Readers and Writers Problem

Suppose that a database is to be shared among several concurrent processes. Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database. We distinguish between these two types of processes by referring to the former as readers and to the latter as writers. Precisely in OS we call this situation as the readers-writers problem. Problem parameters:

- One set of data is shared among a number of processes.
- Once a writer is ready, it performs its write. Only one writer may write at a time.
- If a process is writing, no other process can read it.
- If at least one reader is reading, no other process can write.
- Readers may not write and only read.

Sleeping Barber Problem

- Barber shop with one barber, one barber chair and N chairs to wait in. When no customers the barber goes to sleep in barber chair and must be woken when a customer comes in. When barber is cutting hair new customers take empty seats to wait, or leave if no vacancy. This is basically the Sleeping Barber Problem.



**********************Thank You*************************

Prepared by Prof.Y.J.Patangade
Department BCS
Hutatma Rajgurumahavidyaly
Rajgurunagar.pune

Ref link:- https://www.geeksforgeeks.org
https://www.tutorialspoint.com