

# CS & IT ENGINEERING



## Operating System

Process Synchronization

Lecture – 03

By– Vishvadeep Gothi sir





# Recap of Previous Lecture



**Topic**

**Synchronization**

**Topic**

**Race Condition**

**Topic**

**Mutual Exclusion**

# Topics to be covered



**Topic**

**Mutual Exclusion**

**Topic**

**Progress**

**Topic**

**Bounded Waiting**

**Topic**

**Two-Process Solution for Critical Section**



## Topic : Solution of Critical Section Problem

Requirements of Critical Section problem solution:

1. Mutual Exclusion
2. Progress
3. Bounded Waiting





## Topic : Solution of Critical Section Problem

### **Mutual Exclusion:**

If one process is executing the critical section, then other process is not allowed to enter into critical section.



## Topic : Solution of Critical Section Problem

### Progress:

If no any process is in critical section and any process wants to enter into critical section, then the process must be allowed.





## Topic : Solution of Critical Section Problem

### **Bounded Waiting:**

If a process  $p_1$  is executing in critical section and other process  $p_2$  is waiting for critical section, then the waiting time of  $p_2$  must be bounded. Which means  $p_1$  must not enter in to critical section again and again by keeping  $p_2$  in waiting for long.



## Topic : 2-Process Solution

S/W sol<sup>n</sup> :-

Process  
{

Entry section

critical section

Exit section

}

→ To implement all 3 requirements

→ for announcement of critical section execution completion.





lock = false denotes that c.s. is free.  
= true // c.s. is occupied

## Solution 1

✓ shared variable

Boolean lock=false;

P1

```
while(true)
{
    while(lock);
    lock=true;
    //CS
    lock=false;
    RS;
}
```

P2

```
while(true)
{
    while(lock);
    lock=true;
    //CS
    lock=false;
    RS;
}
```

# checking for mutual exclusion:-

## Case 1:-

one process in C.S.  
and other process wants  
to enter into C.S.

lock = ~~false~~ True

if P1 in C.S. then lock = True,  
P2 can not enter into C.S.  
because it runs while(lock);

## Case 2:-

P1 and P2 runs while(lock);  
one after another.

P1	P2	P1	P2
while( <u>lock</u> );	<u>while(lock)</u> ;	lock = T C.S.	lock = T C.S.

lock = ~~f~~ T Both processes are in C.S.

No Mutual Exclusion





## 2 mins Summary

**Topic**

**Mutual Exclusion**

**Topic**

**Progress**

**Topic**

**Bounded Waiting**

**Topic**

**Two-Process Solution for Critical Section**



**Happy Learning**

**THANK - YOU**