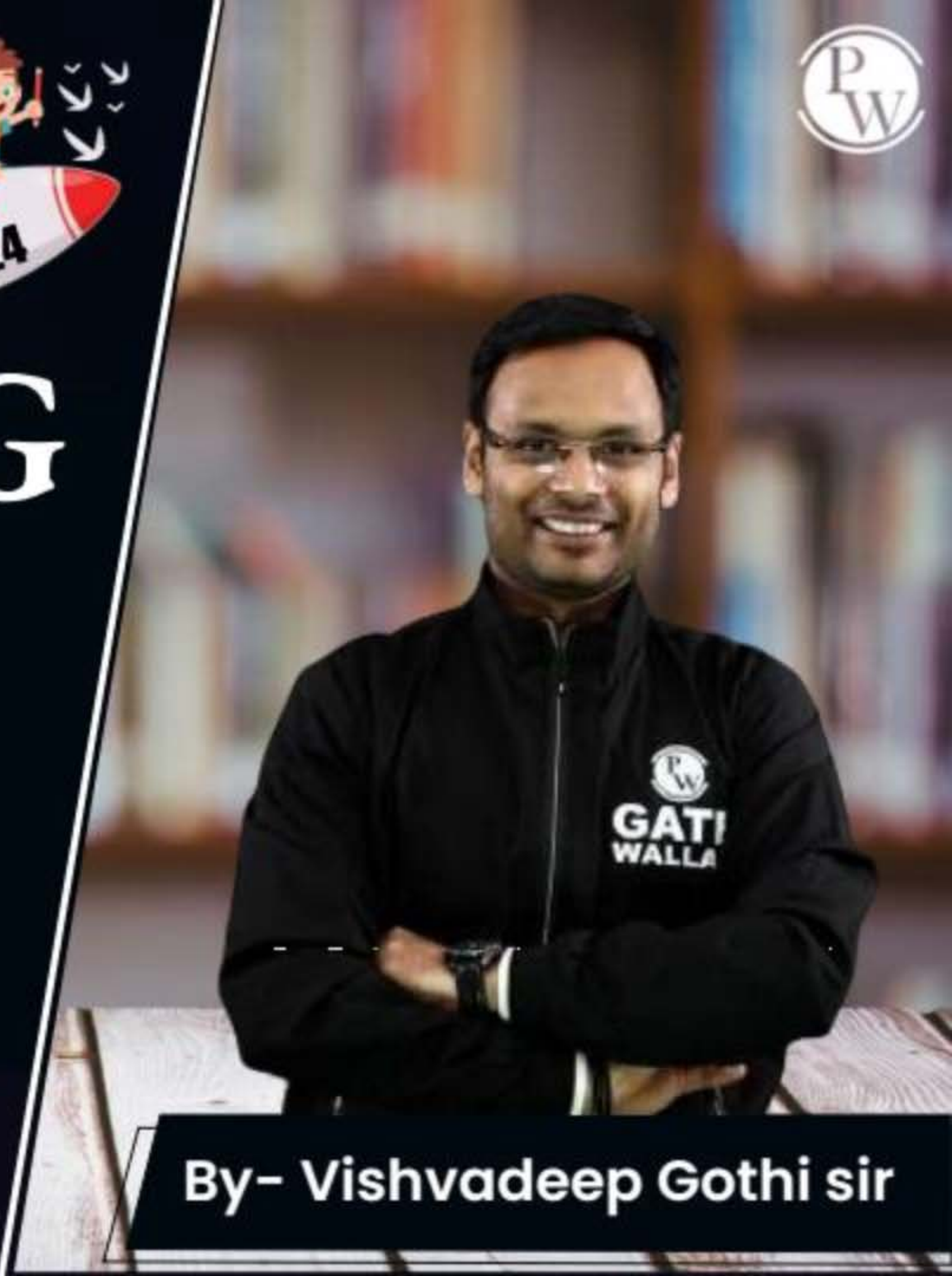
CS & IT ENGINEERING

Operating System

Process Synchronization

Lecture - 05



By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Mutual Exclusion

Topic Progress

Topic Bounded Waiting

Topic Two-Process Solution for Critical Section

Topics to be Covered

- **Topic** Two-Process Solution for Critical Section
- **Topic** Synchronization Hardware
- **Topic** Test-And-Set(), Swap()
- **Topic** Semaphore

Solution 1

× M.E.
× B.W
✓ Progress

✓ Starvatn

```
Boolean lock=false;

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

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

Solution 2

✓ M.E.
✓ B.W.
✗ Progress

✓ starvation

```
int turn=0;

while(true)
{
    while(turn!=0);
    CS
    turn=1;
    RS;
}
```

```
while(true)
{
    while(turn!=1);
    CS
    turn=0;
    RS;
}
```

— M. E.
— Progress
— B. W.

× starvation

Peterson's Solution

```
Boolean Flag[2];
int turn;

while(true) {
    Flag[0]=true;
    turn=1;
    while(Flag[1] && turn==1);
        CS
    Flag[0]=False;
        RS;
}

while(true){
    Flag[1]=true;
    turn=0;
    while(Flag[0] && turn==0);
        CS
    Flag[1]=False;
        RS;
}
```

Topic : Synchronization Hardware

provid instructions in CPU architecture
to support synchronization

1. TestAndSet()
2. Swap()

Topic : TestAndSet()

Returns the current value flag and sets it to true.

Flag = False ⇒ returns false
& sets flag = True

Flag = True ⇒ returns true
& sets flag = True

Topic : TestAndSet()

```
Boolean Lock=False;
```

shared variable

```
boolean TestAndSet(Boolean *trg){
boolean rv = *trg;
*trg = True;
Return  rv;
}
```

solution using Test and set ( )

P1 P2

```
while(true)
{
 while(TestAndSet(&Lock));
        CS
 Lock=False;
}
```

✓ Mutual Exclusion

✓ Progress

X Bounded waiting

✓ starvation

Topic : Swap()

one for each process

```
Boolean Key;                    //Local
Boolean Lock=False;             //Shared
```

↳ indicates that c.s. is free

```
void Swap(Boolean *a, Boolean *b)
{
boolean temp = *a;
*a=*b;
*b=temp;
}
```

P1   Lock = ~~F~~ ~~T~~ ~~T~~ ~~F~~ T   P2

key = ~~T~~ F   key = ~~T~~ ~~T~~ F

P1

```
while(true){
 Key = True;
 while (key==True)
       Swap(&Lock, &Key);
    CS
 Lock=False;
    RS
}
```

✓ M.E.

✓ Progress

× B.W.

✓ starvation

Topic : Synchronization Tool

os levels

application levels

1. Semaphore
2. Monitor

Topic : Synchronization Tool

Semaphore:-

- Integer value which can be accessed using following functions only

  1. wait() / P() / Degrade() $\Rightarrow$
  2. signal() / V() / Upgrade() $\Rightarrow$

  } functions are atomic

semaphore $\Rightarrow$ always non-negative integer (until otherwise given)

Topic : wait() & signal()

```
wait(S)
{
   while(S<=0);
   S--;
}
```

```
signal(S)
{
   S++;
}
```

wait (S) successful only when S>0.

if any binary semaphore has value S = 1 and signal(S) successfully runs with S remains 1.

Topic : Types of Semaphore

Binary Semaphore	Counting Semaphore
It takes only 2 values 0 or 1.	It takes any non-negative integer 0, 1, 2, 3, 4, 5, ...

Topic : Types of Semaphore

Binary Semaphore	Counting Semaphore
It is used to implement the solution of critical section problems with multiple processes	It is used to control access to a resource that has multiple instances

mutual Exclusion

Topic : Critical Section Solution

S = 1 ← binary semaphore

```
while(True)
{
 wait(S)
     C.S.
  signal(s)
}
```

If counting semaphore $s = 2$

P1, P2, P3, P4 4 processes

wait(s)

C.S.

signal(s)

Max No. of processes can be in C.S. section together —— ?

$S = \cancel{2}\ \cancel{1}\ 0$

Ans = 2

How many processes can execute C.S. —— ?

Ans = 4

[NAT]

#Q. Consider a counting semaphore S, initialized with value 10. What should be the value of S after executing 6 times P() and 8 time V() function on S?

$$10 - 6 + 8$$

$$= 12$$

[MSQ]

#Q. Consider a semaphore S, initialized with value 37. Which of the following options gives the final value of S=12?

$37 - 25 = 12$

A. Execution of 22 P() and 15 V() $-22+15=-7$

B. Execution of 25 P() -25

C. Execution of 33 P() and 8 V() $-33+8=-25$

D. Execution of 31 P() and 6 V() $-31+6=-25$

[NAT]

#Q. Consider a binary semaphore S, initialized with value 1. Consider 10 processes P1, P2 .... P10. All processes have same code as given below but, one process P10 has signal(S) in place of wait(S). If all processes to be executed only once, then maximum number of processes which can be in critical section together ?

```
process
{
 wait(S)
      C.S.
  signal(s)
}
```

Ans = 3

Topic : Solution

P1, P2, ..., P9

```
process
{
  wait(S)
     C.S.
  signal(s)
}
```

P10

```
process
{
  signal(S)
     C.S.
  signal(s)
}
```

$S = \cancel{1}\,\cancel{0}\,\cancel{1}\,0$

$S=1$

P1 wait(S) ⇒ in CS

$S=0$

P10 signal(S) ⇒ in CS

$S=1$

P2 wait(S) ⇒ in CS

$S=0$

[MCQ]

#Q. Consider a binary semaphore S, initialized with value 1. Consider 10 processes P1, P2 .... P10. All processes have same code as given below but, one process P10 has signal(S) in place of wait(S). If all processes to be executed only once, then maximum number of processes which can be in critical section together ?

```
while(True)
{
 wait(S)
     C.S.
  signal(s)
}
```

Ans = 10

Topic : Solution

$S = \cancel{1}\,\cancel{0}\,\cancel{1}\,\cancel{0}\,1$

P1, P2, ..., P9

```
while(True)
{
 wait(S)
    C.S.
 signal(s)
}
```

P10

```
while(True)
{
 signal(S)
    C.S.
 signal(s)
}
```

P1 wait(S) ⇒ in CS

P10 signal(s)

P2 wait(S) ⇒ in CS

P10 signal(s)

P3 wait(S) ⇒ in CS

P10 signal(s)

2 mins Summary

Topic Mutual Exclusion

Topic Progress

Topic Bounded Waiting

Topic Two-Process Solution for Critical Section

Happy Learning

THANK - YOU