CS & IT ENGINEERING

Operating System

Process Synchronization

Lecture - 07

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Semaphore

Topic Questions on Semaphore

Topics to be Covered

- **Topic** Classical Problems on Synchronization
- **Topic** Bounded Buffer Problem
- **Topic** Reader-Writer Problem
- **Topic** Dining Philosopher Problem

[NAT]

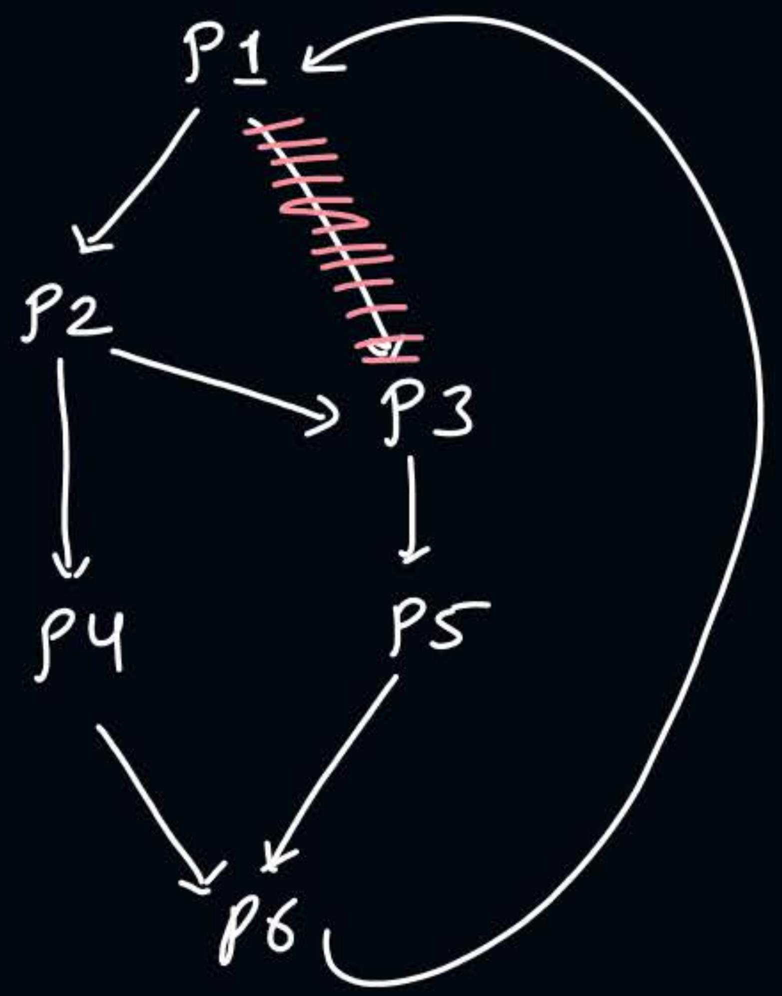
Ans = 8

#Q. A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the process W and X reads x from memory, increments by 2, stores it to memory and then terminates. Each of the processes Y and Z reads x from memory , decrements by 3, stores it to memory and then terminates. Each processes before reading x invokes the P operation (i.e., wait) on a counting semaphore S and invokes the V operation (i.e., signal) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What are the total distinct possible values of x after all processes complete execution ?

$0 + 2 + 2 - 3 - 3$

$2, -3, 4, -6, -1, 1, -4, -2$

H.W. Quest^n



$S1 = 1$

$S2, S3, S4, S5, S61, S62 = 0$

P1

```
wait(S1)

signal(S2)
```

P2

```
wait(S2)

signal(S3)
signal(S4)
```

P3

```
wait(S3)

signal(S5)
```

P4

```
wait(S4)

signal(S61)
```

P5

```
wait(S5)

signal(S62)
```

P6

```
wait(S61)
wait(S62)

signal(S1)
```

[MCQ]

$X = 10$

#Q. Consider the two functions incr and decr shown below.

5 threads ⇒ incr()
3 threads ⇒ decr()

```
incr() {                    decr(){
    wait(s);                    wait(s);
    X = X+1;                    X = X-1;
    signal(s);                  signal(s);
}                           }
```

There are 5 threads each invoking incr once, and 3 threads each invoking decr once, on the same shared variable X. The initial value of X is 10.

Suppose there are two implementations of the semaphore s, as follows:

I1: s is a binary semaphore initialized to 1.

I2: s is a counting semaphore initialized to 2.

Let V1, V2 be the values of X at the end of execution of all the threads with implementations I1, I2, respectively.

Which one of the following choices corresponds to the minimum possible values of V1, V2, respectively? [2023]

A) 15, 7

B) 7, 7

C) ✓ 12, 7

D) 12, 8

I1 :- Binary
$s = 1$

$x \Rightarrow V1$

$$10 + \underbrace{1+1+1+1+1}_{5} \underbrace{-1-1-1}_{3}$$

$= 12$

I2 :- Counting $s = 2$

$x \Rightarrow V2$

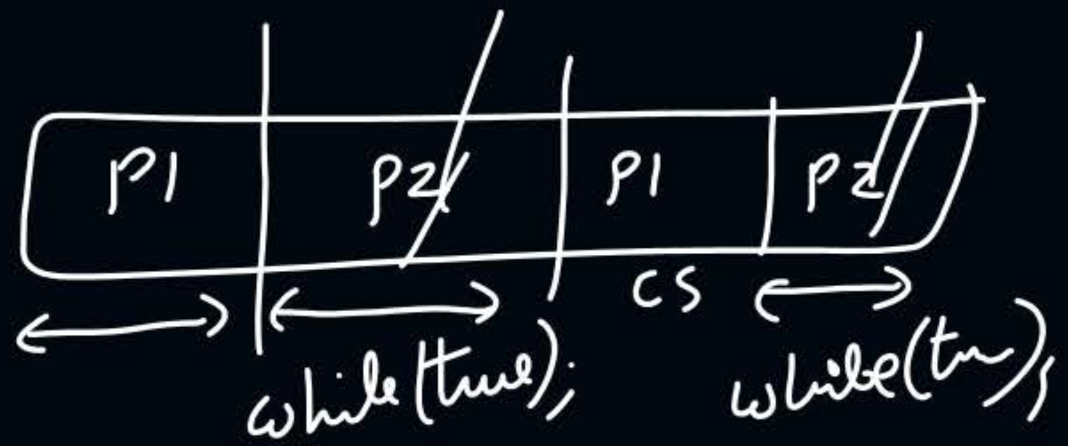
$$10 + \underbrace{1+1+1+1+1}_{\text{nullified}} \underbrace{-1-1-1}$$

$10 - 3 = 7$

Busy waiting :-

a process is busy in running on CPU but still waiting for critical section. {wastage of CPU time}

while (true); ⟸ when process executes this then it uses CPU but still waiting for critical section.



```
while(true);
```

⇑

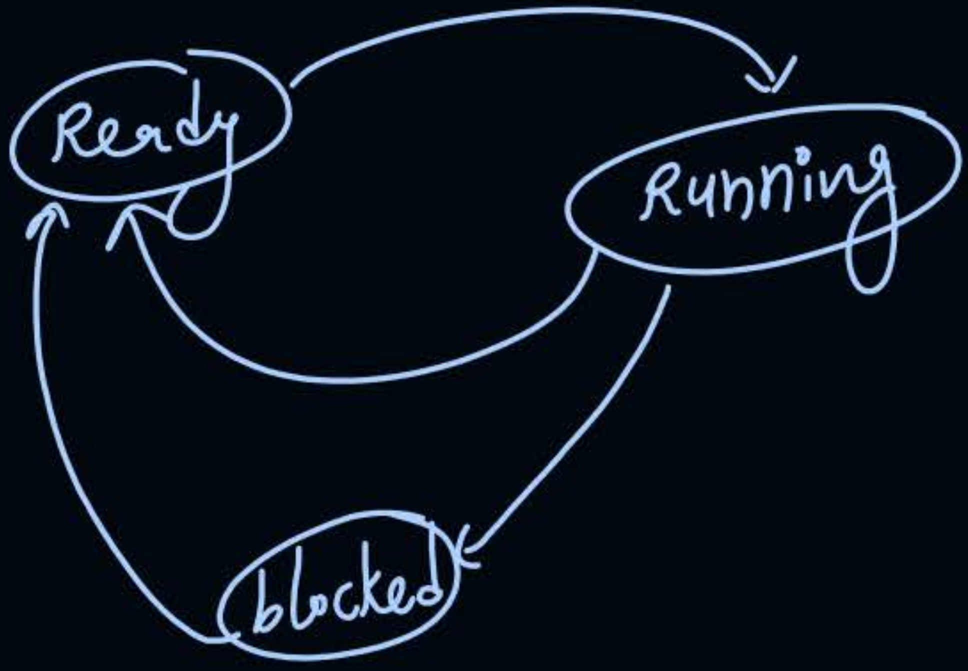
busy waiting

```
wait(s)

c.s.

signal(s)
```

$\Rightarrow$

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

busy waiting



Topic : Solutions Without Busy Waiting

```
wait(Semaphore s){
   s=s-1;
   if (s<0) {
      // add process to queue
      block();
   }
}
```

```
signal(Semaphore s){
   s=s+1;
   if (s<=0) {
      // remove
process p from queue
      wakeup(p);
   }
}
```

P1 ⇒ ~~in CS~~ out

P2 ⇒ ~~blocked~~ ~~in C.S.~~ out

P3 ⇒ ~~blocked~~ ~~CS~~ out

Queue :- ~~P2~~, ~~P3~~

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

✓ wait(s)

✓ C.S.

✓ signal(s)

Topic : Classical Problems of Synchronization

- → Bounded buffer (producer- consumer) problem
- → Reader- writer problem
- → Dining- philosopher problem

Topic : Bounded Buffer Problem

N

Bounded buffer with capacity N

Topic : Bounded Buffer Problem

- Known as producer-consumer problem also
- Buffer is the shared resource between producers and consumers

Topic : Bounded Buffer Problem : Solution

- Producers must block if the buffer is full
- Consumers must block if the buffer is empty

Topic : Bounded Buffer Problem : Solution

- **Variables:**
  - Mutex: Binary Semaphore to take lock on buffer (Mutual Exclusion)
  - Full: Counting Semaphore to denote the number of occupied slots in buffer
  - Empty: Counting Semaphore to denote the number of empty slots in buffer

- **Initialization:**
  - Mutex = 1
  - Full = 0
  - Empty = N

Buffer is completely empty initially

Topic : Producer()

wait(Empty)
wait(mutex)
// add produced item on buffer
signal(mutex)
signal(Full)

if first 2 wait() statements are swapped then there will be a deadlock when buffer is completely full.

Topic : Consumer()

```
wait(Full)
wait(Mutex)
// remove item from buffer & consume
signal(mutex)
signal(Empty)
```

if first 2 statements of consumer process are swapped then there can be deadlock when buffer is empty

Prod.

wait(empty)
→ wait(mutex)

———

signal(mutex)
signal(full)

Cons.

wait(mutex)
→ wait(Full)

———

signal(mutex)
signal(Empty) ✓

full = 0
empty = ~~n~~ n−1
mutex = ~~1~~ 0

2 mins Summary

Topic Classical Problems on Synchronization

Topic Bounded Buffer Problem

Topic Reader-Writer Problem

Topic Dining Philosopher Problem

Happy Learning

THANK - YOU