CS & IT ENGINEERING

Operating System

Process Synchronization

Lecture -4

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Semaphore

Topic Questions on Semaphore

Topics to be Covered

- **Topic** Samaphore
- **Topic** Producer-Consumer Problem
- **Topic** Reader-Writer Problem
- **Topic** Dining Philosopher Problem

[NAT]

Ans = 7

#Q. Consider a non-negative counting semaphore S. The operation P(S) decrements S, and V(S) increments S. During an execution, 20 P(S) operations and 12 V(S) operations are issued in some order. The largest initial value of S for which at least one P(S) operation will remain blocked is _______.?

[2016]

Successful $P(S) = 20 - 1 = 19$

$S - 19 + 12 = 0$

$\boxed{S = 7}$

[NAT]

#Q. Consider a non-negative counting semaphore S. The operation P(S) decrements S, and V(S) increments S. During an execution, 43 P(S) operations and 27 V(S) operations are issued in some order. The largest initial value of S for which at least 5 P(S) operation will remain blocked is __11__.?

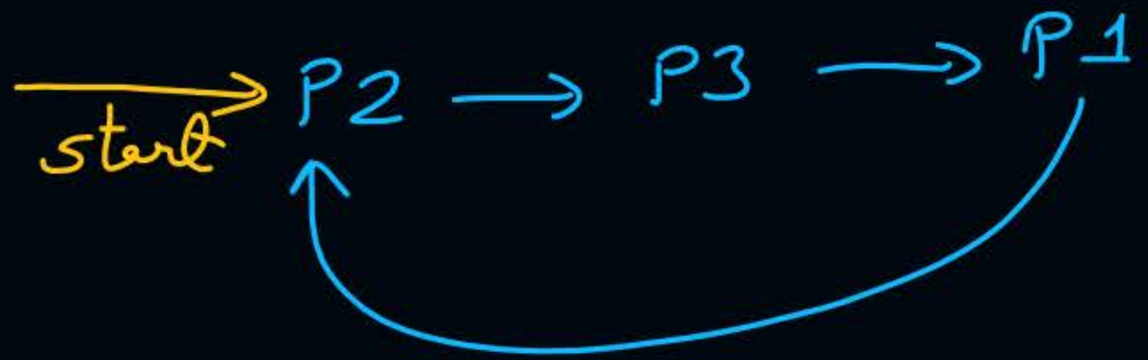
successful $P(S) = 43 - 5 = 38$

$S - 38 + 27 = 0$

$S = 11$

Ques) 3 Processes P1, P2, P3

$S1 = 0, \quad S2 = 1, \quad S3 = 0$

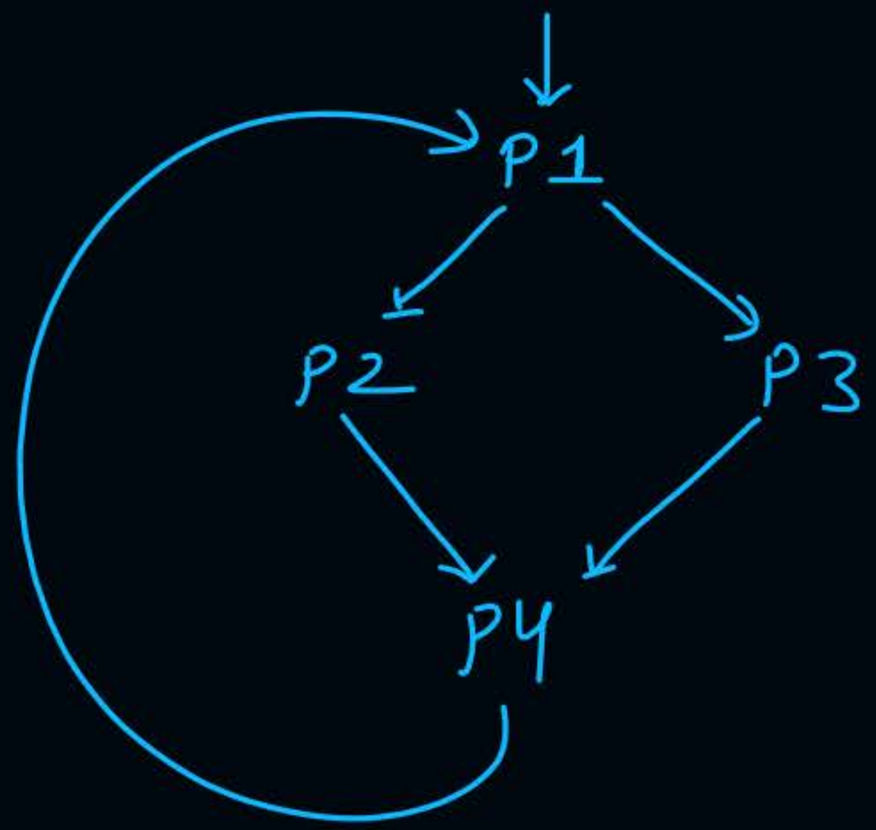


Precedence graph

P1	P2	P3
wait(S1)	wait(S2)	wait(S3)
signal(S2)	signal(S3)	signal(S1)

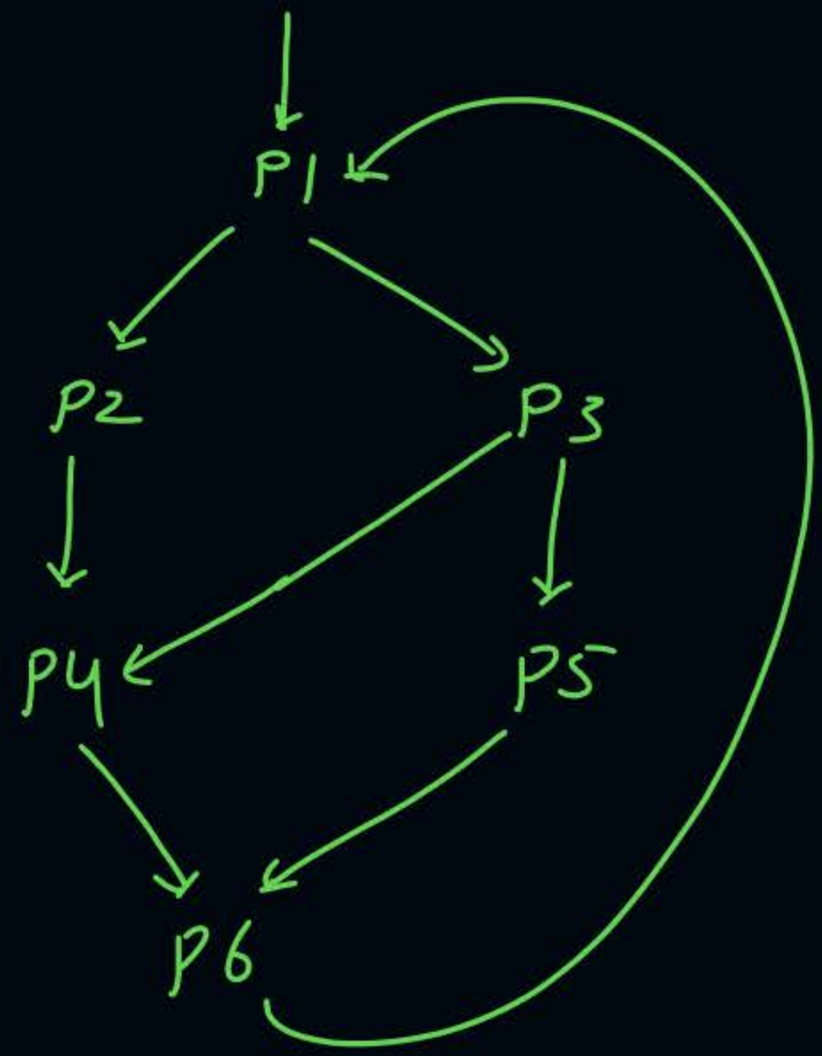
Ques) 4 Processes P1, P2, P3, P4

S1 = 1, S2 = 0, S3 = 0, S41 = 0, S42 = 0



P1	P2	P3	P4
wait(S1)	wait(S2)	wait(S3)	wait(S41) wait(S42)
≡	≡	≡	≡
signal(S2) signal(S3)	signal(S41)	signal(S42)	signal(S1)

Ques)



S1 = 1, S2 = 0, S3 = 0, S41 = 0, S42 = 0, S5 = 0, S61 = 0, S62 = 0

P1	P2	P3	P4
wait (S1)	wait (S2)	wait (S3)	wait (S41) wait (S42)
≡	≡	≡	≡
signal (S2) signal (S3)	signal (S41)	signal (S42) signal (S5)	signal (S61)

P5	P6
wait (S5)	wait (S61) wait (S62)
≡	≡
signal (S62)	signal (S1)

[NAT]

$x = 0$

Ans = 2

#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 one, stores it to memory and then terminates. Each of the processes Y and Z reads x from memory , decrements by two, 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 is the maximum possible value of x after all processes complete execution? **[2013]**

W	X	Y	Z
wait(S)	wait(S)	wait(S)	wait(S)
$x = x + 1$	$x = x + 1$	$x = x - 2$	$x = x - 2$
signal(S)	signal(S)	signal(S)	signal(S)

Ques) In prev Ques.

1. min. possible value of x = $\underline{-4}$?

2. no. of distinct possible values of x = $\underline{7}$?

$0+1+1-2-2 \Rightarrow 1, -2, 2, -4, -1, 0, -3$

Ques) In prev. Ques if S is initialized by 1.

max value of x = $\underline{-2}$?

Sol^n mutual exclusion satisfied, all processes can run one-by-one.

[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 ?

$0+2+2-3-3$

$\underbrace{2, -3, 4, -6, -1, 3, -4, -2}_{8 \text{ values}}$

[MCQ]

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

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

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

$V_1 = 10+1+1+1+1+1-1-1-1$
$= 12$

$V_2 = 7$

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

spin lock (busy waiting) :-

if a process uses CPU but still waits for C.S. by running a forever loop.

$\Downarrow$

wastage of CPU time.

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);
    }
}
```

[NAT]

#Q. Given below is a program which when executed spawns two concurrent processes:

Semaphore X:=~~0~~;

/* Process now forks into concurrent processes P1 & P2 */

P1 : repeat forever	P2:repeat forever
V(X);	P(X);
Compute;	Compute;
P(X);	V(X);

Consider the following statements about processes P1 and P2:

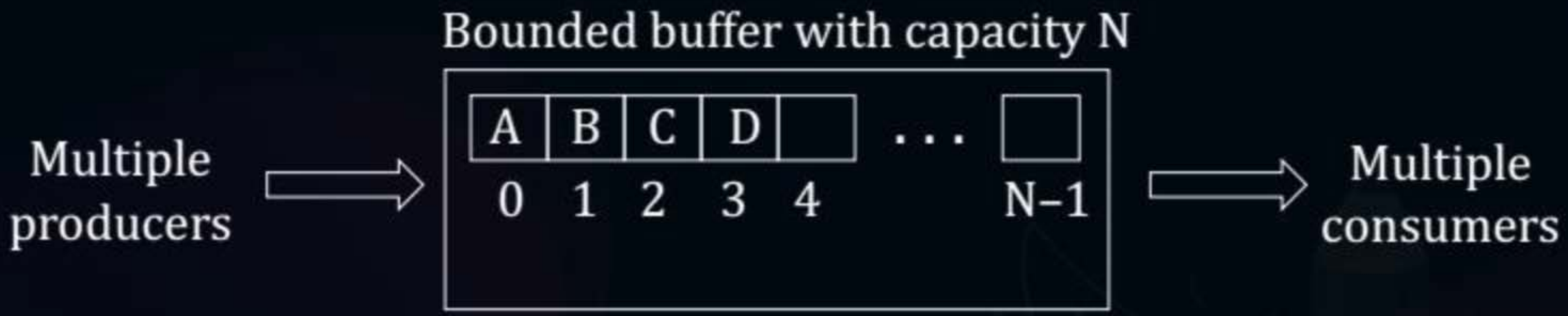
I: It is possible for process P1 to starve.

II. It is possible for process P2 to starve.

Topic : Classical Problems of Synchronization

1. Producer - Consumer (bounder - buffer)
2. Reader - writer
3. Dining - philosopher

Topic : Bounded Buffer Problem



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

Topic : Producer() & Consumer

Producer

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

Consumer

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

Topic : Producer() & Consumer

Producer

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

Consumer

wait(Full)
→ wait(mutex)
//remove item from buffer
signal(mutex)
signal(Empty)

if buffer is full ⇒ $\left.\begin{array}{l} Full = \cancel{N}\ N-1 \\ Empty = 0 \\ mutex = \cancel{1}\ 0 \end{array}\right\}$ deadlock

Topic : Producer() & Consumer

Producer

wait(Empty)

wait (mutex)

//add item on buffer

signal(mutex)

signal(Full)

Consumer

wait(mutex)

wait (full)

//remove item from buffer

signal(mutex)

signal (Empty)

When buffer is empty ⇒ Full = 0

Empty = ~~N~~ N-1

mutex = ~~1~~ 0

deadlock

Topic : Reader-Writer Problem

Consider a situation where we have a file shared between many people:

- If one of the people tries editing the file, no other person should be reading or writing at the same time, otherwise changes will not be visible to him/her
- However, if some person is reading the file, then others may read it at the same time

Topic : Reader-Writer Problem Solution

- If writer is accessing the file, then all other readers and writers will be blocked
- If any reader is reading, then other readers can read but writer will be blocked

	Reader	writer
Reader	✓	✗
writer	✗	✗

Topic : Reader-Writer Problem Solution

- Variables:
  - mutex: Binary Semaphore to provide Mutual Exclusion
  - wrt: Binary Semaphore to restrict readers and writers if writing is going on
  - Readcount: Integer variable, denotes number of active readers

- Initialization:
  - mutex: 1
  - w r t : 1
  - Read count: 0

Topic : Writer() & Reader() Processes

writer

```
wait(wrt)
//writing
signal(wrt)
```

wrt = ~~1~~ 0

readcount = ~~0~~ 1

mutex = ~~1~~ 0

Reader

```
wait(mutex)
Readcount++;
if(Readcount == 1)
    wait(wrt);
signal(mutex)

// Reading

wait(mutex)
Readcount--;
if(Readcount == 0)
    signal(wrt);
signal(mutex)
```

Topic : Writer() & Reader() Processes

writer

```
wait(wrt)
//writing
signal(wrt)
```

Reader

```
wait(mutex)
Readcount++;
if(Readcount==1)
    wait(wrt);

//Reading

Readcount--;
if(Readcount==0)
    signal(wrt);
signal(mutex)
```

only one reader will be allowed at a time for reading

2 mins Summary

- **Topic** Samaphore
- **Topic** Producer-Consumer Problem
- **Topic** Reader-Writer Problem
- **Topic** Dining Philosopher Problem

Happy Learning

THANK - YOU