CS & IT ENGINEERING

Operating System

Process Synchronization

/Coordination

Lecture No. 5

By- Dr. Khaleel Khan Sir

- Semaphore
- Counting & Binary Semaphores
- Problem Solving

SEMAPHORES :

→ Proposed by E. Dijkstra

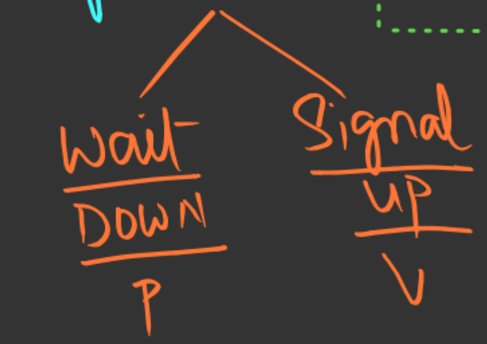
→ It is an O S Resource

→ It is a general purpose utility ;

→ It is implemented as an A.D.T

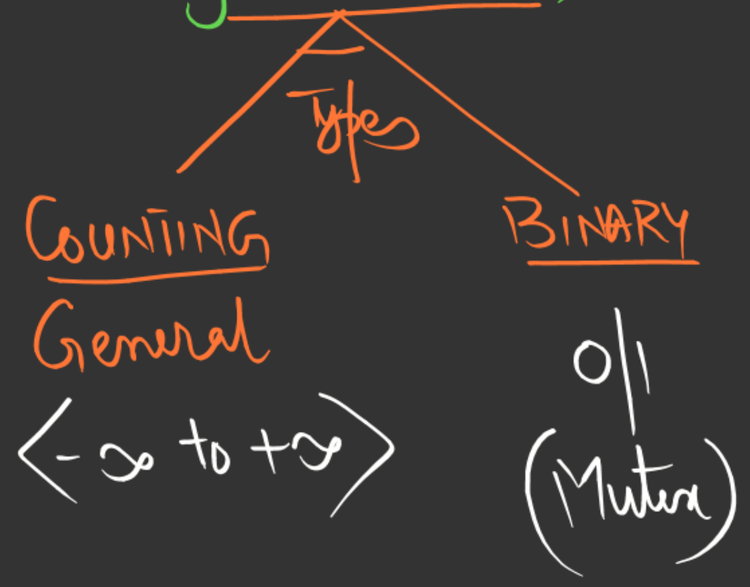
→ Semaphore Support 2-operations

→ Semaphore op'rs are atomic ;



→ Semaphore operations are used in the applications like System Calls ;

→ Defn : Semaphore is a variable (SEM : ADT), that takes only Integer value ;



1. COUNTING SEMAPHORE

K.m

```
struct
{
    int value;
    QueueType L;
} CSEM;
```

→ List of PCB's of those Processes, that gets Blocked while performing "DOWN" op'n unsuccessfully

U.m

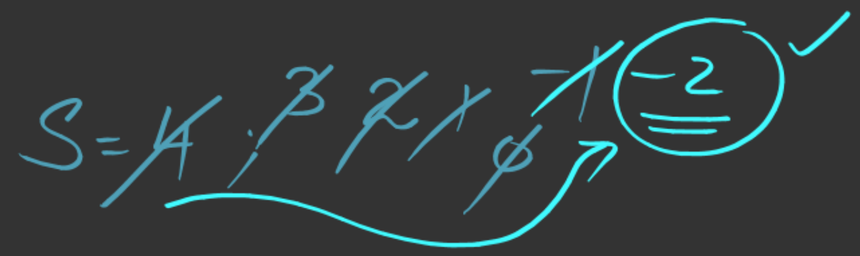
```
CSEM S;
S.value = 4;
:
DOWN(S);
```

Sys-Call

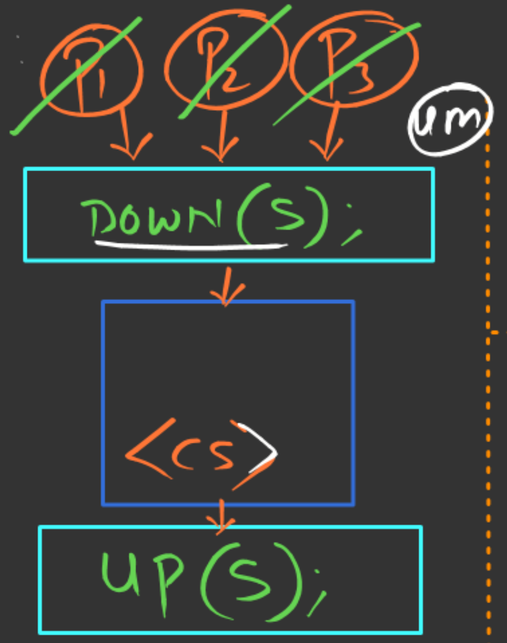
< Stmnt >

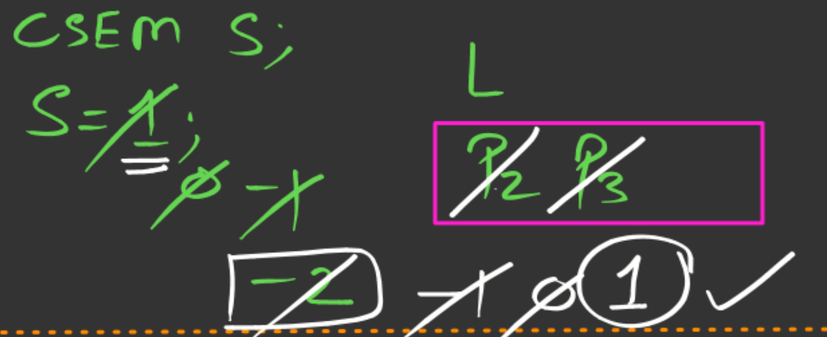
K.m

```
DOWN(CSEM S)
{
1. S.value = S.value - 1;
2. if (S.value < 0) [unsuccess]
   {
      put this process (PCB)
      in S.L (Queue)
      & Block_it (Sleep)
   }
   else (≥0) [Success]
      return;
}
```

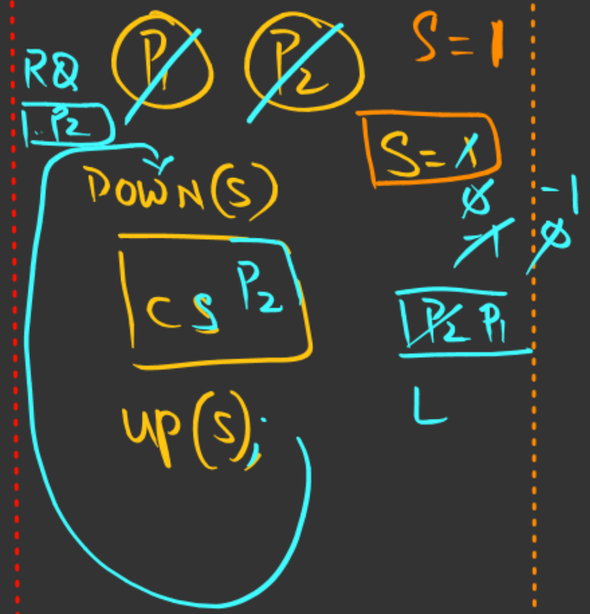


Down: 1,1,1,1
: 1,1





UP(CSEM S)
{
1. S.value = S.value + 1;
2. if(S.value ≤ 0)
{
Select a Process from S.L(0) & wakeup();
}
else
return;
}



Note 1: "+ve value indicates that those many down op's can be carried out successfully"

Note 2: -ve value indicates, No. of Blocked Processes in the Q

Q1) CSEM S;

S = ~~16~~ ;

Opns : ~~6P~~ ; ~~3V~~ ; ~~15P~~ ; ~~4V~~ ; ~~12P~~ ; ~~3V~~ ;

4 7 -8 -4 -16 -13 ✓

S = ?

= -13 ✓

Q2) CSEM S;

S = x

Opns : 10P ; 2V ; 18P ; 6V ;

[x - 10 + 2 - 18 + 6] = -5

S = -5 ;

x - 20 = -5

x = 15 ✓

Q) Consider a Counting Semaphore S; During an execution, 20 'P' ops & 12 'V' ops are issued in some order; The largest initial value of 'S' for which atleast one P(S) operation will remain blocked is ——; (NAT)

$S = x$

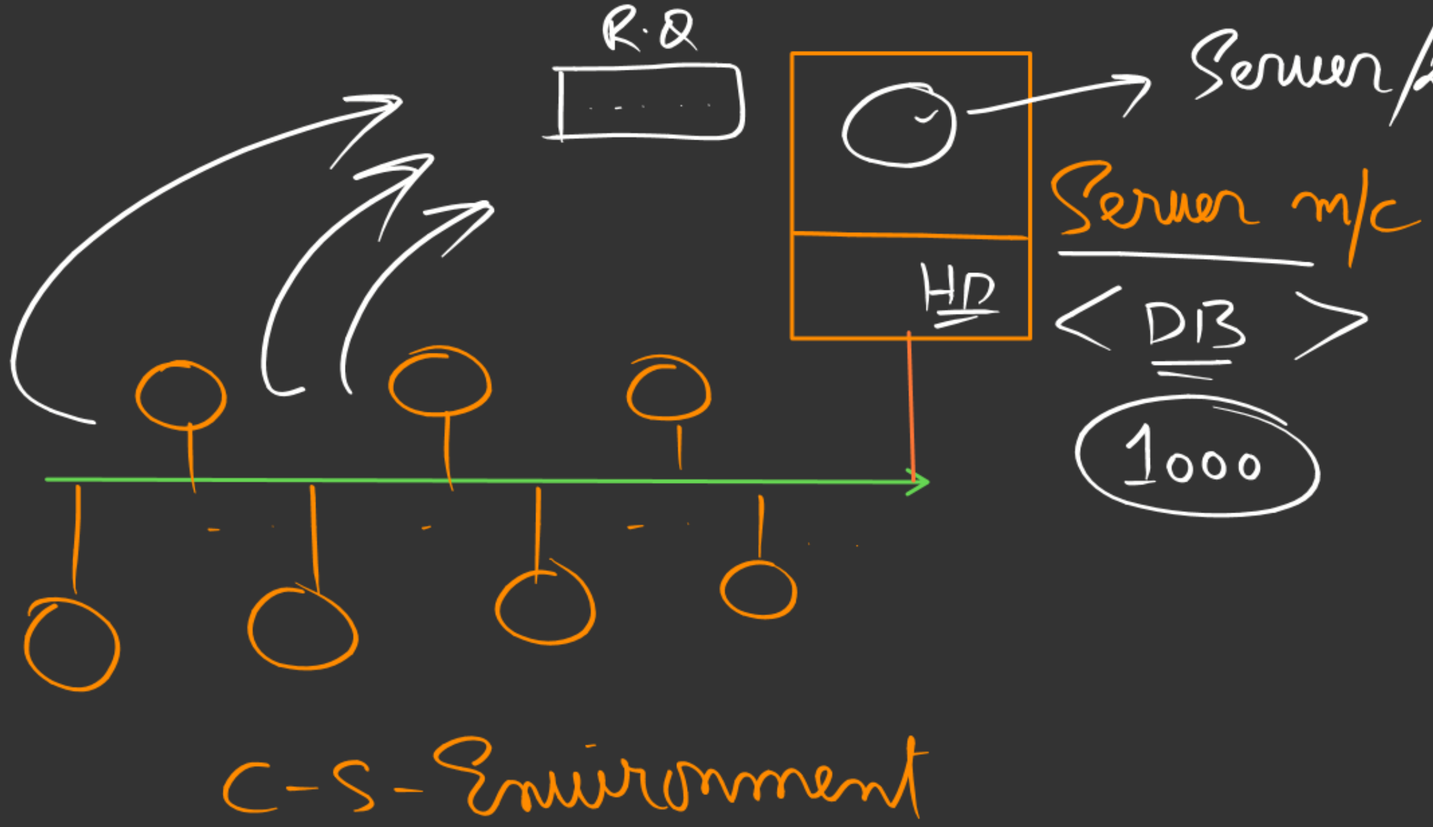
$[x - 20 + 12] = -1$

$x - 8 = -1$

$\boxed{x = 7}$

$S = 7$

$[7 - 20 + 12] = -1$




```
CSEM S;
S=1000
Process(i) i=1,∞
{
Down(S)
< DB >
Up(S);
}
```

II. Binary Semaphore (BSEM) / Mutexes

```
Struct
{
  enum value (0,1);
  QueueType L;
} BSEM;
```

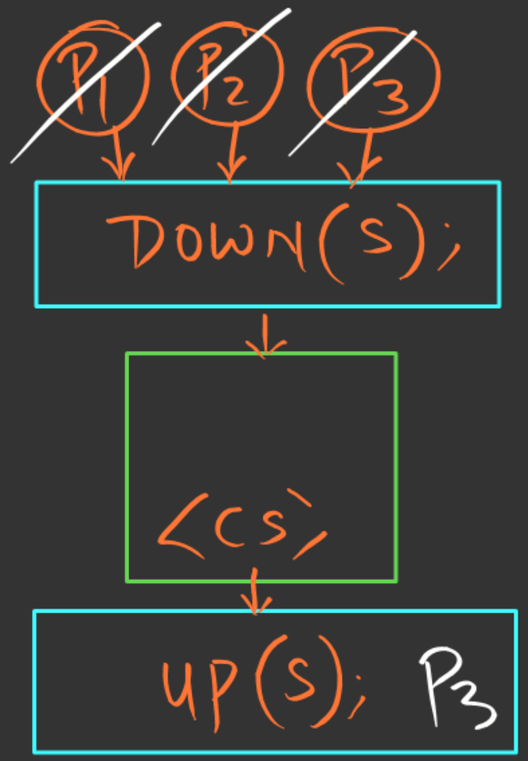
```
BSEM S;

S = 1;
:
:
:
DOWN(S);
<stmnt>;
```

```
DOWN(BSEM S)
{
  1. if (S.value = 1)
     {
       S.value = 0;
       return;
     }
     else
     {
       put this
         Process in
         S.L(Q) &
         Block(sleep)
     }
}
```

Success

Unsuccess



BSEM S;

S = ~~1~~ 0;

L

```
UP(BSEM S)
{
  if (S.L(Q) is NOT Empty)
  {
    Select a Process &
    Wakeup(); // Value remains
                 0 //
  }
  else
  {
    S.value = 1;
  }
}
```

BSEM S;

Q1)

Case I:

S = 1;
P(S);
S = 0
Status = Success

Case II:

S = 0;
P(S);
S = 0
Status = Unsuccess

Case III:

S = 0
V(S);
S = 0 – 'Q' is NOT Empty
S = 1 – 'Q' is Empty
Status = Success

Case IV:

S = 1;
V(S);
S = 1
Status = Success

Case V:

S = 0 (Initial)
V(S); (First op'n)
S = 1 [Q is Empty]
Status = Success

Q

Bsem S = 1, T = 1

```
Pr (i)
{
  while (1)
  {
    P(S);
    P(T);
    <CS>
    V(T);
    V(S);
  }
}
```

Same order

```
Pr (j)
{
  while (1)
  {
    P(S);
    P(T);
    <CS>
    V(S);
    V(T);
  }
}
```

S = X; 0

T = X; 0

Does it guarantee ME? How about Deadlock?

Q

PYQ

BSem S = 1, T = ~~0~~ 1

```
Pr (i)
{
  while (1)
  {
    P(T);
    Print ('1');
    Print ('1');
    V(S);
  }
}
```

```
Pr (j)
{
  while (1)
  {
    P(S);
    Print ('0');
    Print ('0');
    V(T);
  }
}
```

STRICT ALTERNATION

00110011001 1 . . .

$(0011)^+$

does it guarantee ME ? ✓

Progress = ? ✗

Bounded wait = ✓

What is the Regular Expression that gets generated?

(O/P Sequence generated)

Q

Bsem m [0....4] = {1};

Assign

```
Pr (i)  i =0 , 4
{
      while (1)
      {
            P(m[i]);
            P(m[(i+1) %4]);
            <CS>
            V(m[i]);
            V(m[(i+1) %4]);
      }
}
```

What is the maximum no. of processes that can be in <CS>.

Q.

H/W

BSem S = 1, T = 0, Z = 0

```
Pr (i)
{
  While (1)
  {
    P(S);
    Print (*);
    V(T);
    V(Z);
  }
}
```

```
Pr (j)
{
    P(T);
    V(S);
}
```

```
Pr (k)
{
    P(Z);
    V(S);
}
```

What is the minimum and maximum no. of "*" that get printed.

H/W

Q

Consider the following threads, T1, T2 and T3 executing on a single processor, synchronized using three binary semaphore variables, S1, S2, and S3, operated upon using standard wait() and signal(). The threads can be context switched in any order and at any time.

Which initialization of the semaphores would print the Sequence BCABCABCA....?

T_1	T_2	T_3
While(true){	While(true){	While(true){
Wait(S_3);	Wait(S_1)	Wait(S_2)
Print("C");	Print("B");	Print("A");
Signal(S_2);}	Signal(S_3);}	Signal(S_1);}

A. S1 = 1; S2 = 1; S3 = 1

B. S1 = 1; S2 = 1;S3 = 0

C. S1 = 1; S2 = 0; S3 = 0

D. S1 = 0; S2 = 1; S3 = 1

H/W

Q

Consider a counting semaphore initialized to 2, there are 4 concurrent processes P_i, P_j, P_k & P_L. The Processes P_i & P_j desire to increment the current value of variable C by 1 whereas P_K & P_L desire to decrement the current value of C by 1. All Processes perform their update on C under semaphore control. What can be the minimum and maximum value of C after all Processes finish their update.

H/W

Q

Consider Three Processes using four Binary Semaphores a, b, c, d in the order shown below.

Which Sequence is a Deadlock Free sequence?

(I) X: P(a); P(b); P(c);
Y: P(b); P(c); P(d);
Z: P(c); P(d); P(a);

(II) X: P(b); P(a); P(c);
Y: P(b); P(c); P(d);
Z: P(a); P(c); P(d);

(III) X: P(b); P(a); P(c);
Y: P(c); P(b); P(d);
Z: P(a); P(c); P(d);

(IV) X: P(a); P(b); P(c);
Y: P(c); P(b); P(d);
Z: P(c); P(d); P(a);

Q

Each of a set of n processes executes the following code using two semaphores a and b initialized to 1 and 0, respectively. Assume that count is a shared variable initialized to 0 and not used in CODE SECTION P.

```
CODE SECTION P
wait(a); count=count+1;
if (count==n) signal (b);
signal (a): wait (b) ; signal (b);
CODE SECTION Q
```

+1/w

What does the code achieve ?

A. It ensures that no process executes CODE SECTION Q before every process has finished CODE SECTION P.

B. It ensures that two processes are in CODE SECTION Q at any time.

C. It ensures that all processes execute CODE SECTION P mutually exclusively.

D. It ensures that at most n−1 processes are in CODE SECTION P at any time.

Consider the two functions Incr( ) and Decr( )

G-2023

H/W

```
Incr( ){                Decr( ) {
  wait(s);                wait(s);
  x = x + 1;              x= x – 1;
  signal(s);              signal(s);
}                       }
```

Five Processes invoke Incr( ) and Three Processes invoke Decr( )

X is a shared variable initialized to 10.

I_1 : s value is 1 (Binary semaphore)

I_2 : s value is 2 (Counting semaphore)

Let V_1 and V_2 be the minimum possible values of the implementation of I1 and I2, then choose the values of x for V_1 and V_2.

(A) 12, 7

(B) 7, 7

(C) 15, 7

(D) 12, 8


THANK YOU
GW
SOLDIERS !