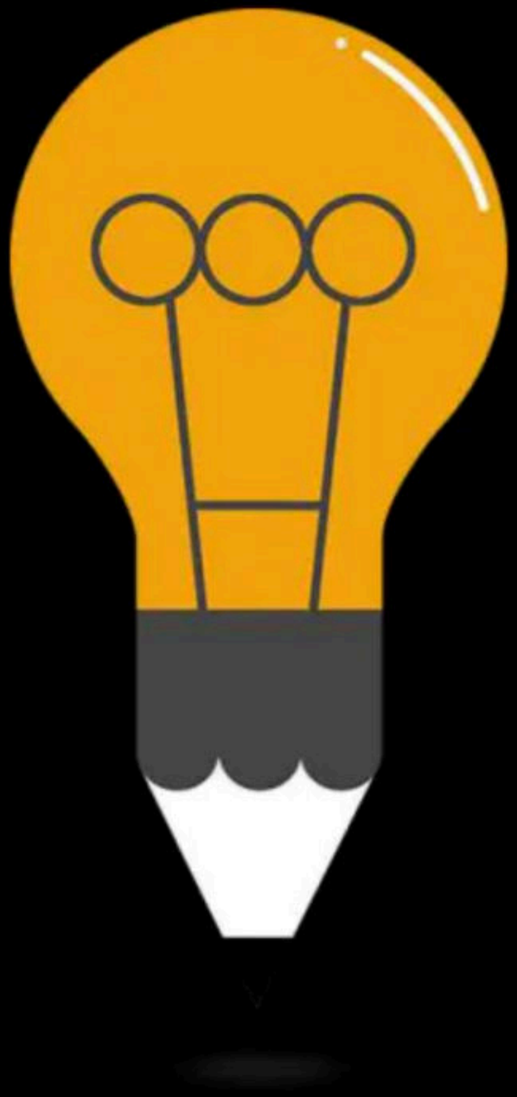
unacademy

# Semaphore Questions & Classical Synchronization Problems

Comprehensive Course on Operating System for GATE - 2024/25

**Vishvadeep Gothi** • Lesson 17 • May 8, 2023

unacademy



# Operating System
# Semaphore

By: Vishvadeep Gothi

unacademy

# Solution 1

```
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

```
int turn=0;
```

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

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

# Solution 3: 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;
}
```

unacademy

# Synchronization Hardware

1. TestAndSet()
2. Swap()

# TestAndSet()

```
Boolean Lock=False;

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

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

unacademy

# Swap()

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

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

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

# Synchronization Tools

1. Semaphore
2. Monitor

# Semaphore

- Integer value which can be accessed using following functions only
  - wait() / P() / Degrade()
  - signal() / V() / Upgrade()

# wait() & signal()

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

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

# Types of Semaphore

0 , 1

0, 1, 2, 3, .....

Binary Semaphore

Counting Semaphore

# 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 |

# Critical Section Solution

```
S = 1

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

# Question

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?

$= 10 - (6 * 1) + (8 * 1)$

$= 12$

# Question

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

a) Execution of 22 P() and 15 V()

b) Execution of 25 P()

c) Execution of 33 P() and 8 V()

d) Execution of 31 P() and 6 V()

$37 - x = 12$

$x = 25$

Consider a binary semaphore S = 1.

What will be the value of S, after following operations?

| | S |
|---|---|
| wait(S) | 0 |
| signal(S) | 1 |
| wait(S) | 0 |
| signal(S) | 1 |
| signal(S) | 1 |

Notes :-

If binary semaphore value is 1. Then signal(S) will run successfully and value will remain same.

if $S = 0$

wait(s) cannot run successfully. Process will be stuck in while loop inside wait().

Ques) Consider 4 processes P1, P2, P3, P4 as follows. A binary semaphore S is initialized with value S = 1

| P1 | P2 | P3 | P4 |
|---|---|---|---|
| wait(S) | wait(S) | Signal(S) | Signal(S) |
| Print(1) | Print(1) | Print(1) | Print(1) |

when all processes Run successfully, then min. no. of times 1 printed?

a) 1 b) 2 c) 3 d) 4

1,1 S=0

Is it possible that all processes may not complete?

P3, P4, P1, P2

yes max 1 process either P1 or P2

In previous questh if S is initialized with zero, then max. how many processes may stuck in wait() ?

---

Ans = 1

run

S = 0 P3, P4 ⇒ S = 1 ⇒ P1, P2

↓

one will stuck in wait()

# Question

Ans = 3

binary

Consider a 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)
}
```

# Solution

**P1, P2, ....., P9**

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

**P10**

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

① P1 runs wait(s) & enters into C.S.

S = 0

Preemption

② P10 runs signal(s) & enters into C.S.

S = 1

Preemption

③ P2 runs wait(s) & enters into C.S.

S = 0

In C.S.

P1 , P10, P2

# Question

binary

Consider a 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 ?

Ans = 10

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

# Solution

$S = 1$

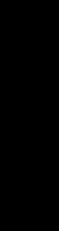
**P1, P2, ....., P9**

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

**P10**

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

1. P1 in C.S. $S = 0$
2. P10 in C.S. $S = 1$
3. P2 in C.S $S = 0$
4. P16 out of C.S. $S = 1$
5. P3 in C.S. $S = 0$
6. P10 in C.S. $S = \underline{1}$
7. P4 in C.S. $S = 0$



2 process strict alternation execution and blocking from P2.

Soln :- 2 binary semaphores. S1 = 0, S2 = 1

P1

wait(S1)

C.S.

Signal(S2)

P2

wait(S2)

C.S.

Signal(S1)

S2 = 0 ~~1~~

S1 = ~~0~~ 1

# Question

Consider a scenario where 3 processes are available: P0, P1 and P2. The processes must be executed in order P1, P0 and P2 only. Write a piece of code to control the sequence of execution with minimum number of semaphores?

semaphores → binary

3 semaphores | S0 = 0, S1 = 1, S2 = 0

| P0 | P1 | P2 |
|---|---|---|
| wait(S0) | wait(S1) | wait(S2) |
| ⋮ | ⋮ | ⋮ |
| signal(S2) | signal(S0) | signal(S1) |

# Question GATE-2022

Consider the following threads, $T_1$, $T_2$, and $T_3$ executing on a single processor, synchronized using three binary semaphore variables, $S_1$, $S_2$, and $S_3$, operated upon using standard **wait()** and **signal()**. The threads can be context switched in any order and at any time.

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| 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($S_1$); } |

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

A. $S_1 = 1; S_2 = 1; S_3 = 1$
B. $S_1 = 1; S_2 = 1; S_3 = 0$
C. $S_1 = 1; S_2 = 0; S_3 = 0$
D. $S_1 = 0; S_2 = 1; S_3 = 1$

unacademy

# GATE - 2010

The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S0=1, S1=0 and S2=0.

| Process P0 | Process P1 | Process P2 |
|---|---|---|
| while (true) {<br>wait (S0);<br>print '0';<br>release (S1);<br>release (S2);<br>} | wait (S1);<br>release (S0); | wait (S2);<br>release (S0); |

S0 = 1 + 0 + 0

S1 = 0 + 0 | 1

S2 = 0 + 0 | 1

How many times will process P0 print '0'?

A. At least twice
B. Exactly twice
C. Exactly thrice
D. Exactly once

00

Prints 0 2 times or 3 times

# Question GATE-2013

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?

Ans = 2

a) −4
b) −2
c) 0
d) 2
e) none

x = 0 S = 2

**W**

wait (S)
R1 = x 0
R1 = R1 + 1 1
x = R1
signal (S)

**X**

wait (S)
R2 = x
R2 = R2 + 1
x = R2
signal (S)

**Y**

wait (S)
R3 = x
R3 = R3 - 2
x = R3
signal (S)

**Z**

wait (S)
R4 = x
R4 = R4 - 2
x = R4
signal (S)

Sol) W, Y read x = 0

S = ~~2 1 0~~ ~~1~~ 2

x = ~~2~~ 1

X, Z read x = ~~1~~ ~~1~~ 2

S = ~~2 1 0~~ ~~1~~ 2

min possible value of $xc = -4$

---

no. of distinct values of $x$ are possible?

$\Rightarrow x + 1 + 1 - 2 - 2$ —

$-3, 0, -4, -1, 2, 1, -2$

# Question GATE-2023 ⇒ DPP (Homework)

Consider the two functions **incr** and **decr** shown below.

```
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 $\mathrm{X}$. The initial value of $\mathrm{X}$ is $10$.

Suppose there are two implementations of the semaphore $\mathbf{s}$, as follows:

**I-1:** s is a binary semaphore initialized to $1$.
**I-2:** s is a counting semaphore initialized to $2$.

Let $\mathrm{V1}$, $\mathrm{V2}$ be the values of $\mathrm{X}$ at the end of execution of all the threads with implementations **I-1**, **I-2**, respectively.

Which one of the following choices corresponds to the minimum possible values of $\mathrm{V1}$, $\mathrm{V2}$, respectively?

A. $15, 7$
B. $7, 7$
C. $12, 7$
D. $12, 8$

# Counting Semaphore

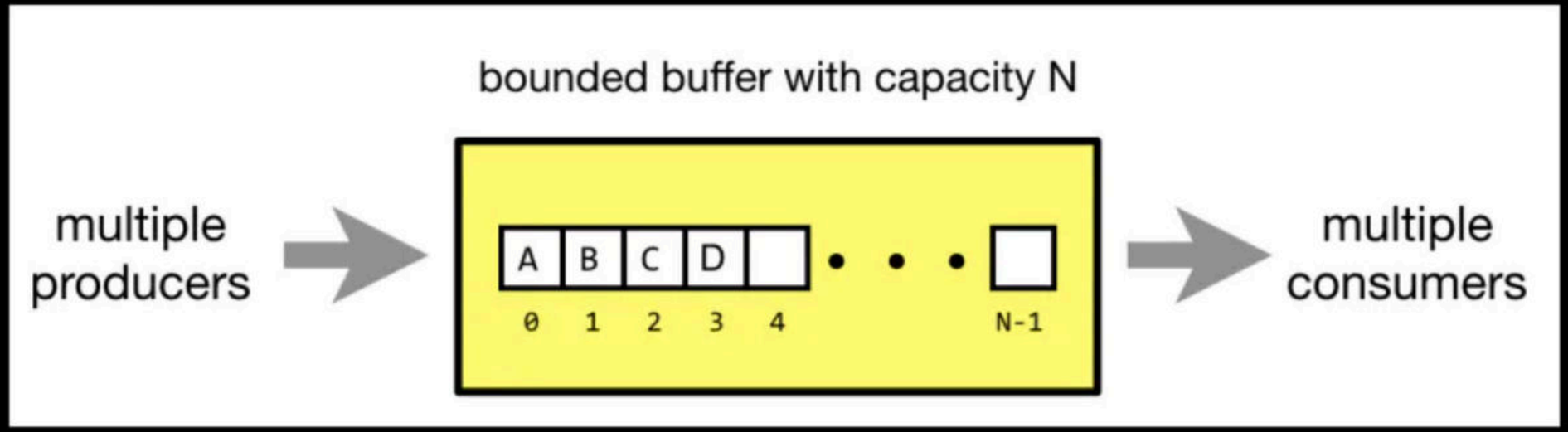
# Solutions Without Busy Waiting

```
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);
    }
}
```

# Classical Problems of Synchronization

# Bounded Buffer Problem



# Bounded Buffer Problem

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

# Bounded Buffer Problem: Solution

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

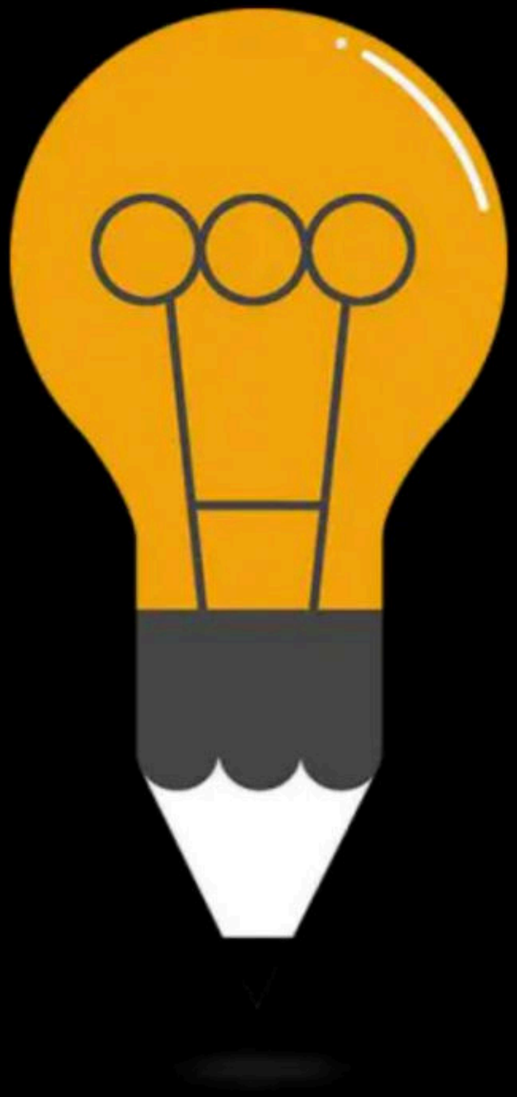
# Bounded Buffer Problem: Solution

- 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

# Producer()

# Consumer()

unacademy



# DPP

By: Vishvadeep Gothi

unacademy

# Question 1

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 minimum and maximum possible values of $x$ after all processes complete execution?

unacademy

# Question 2

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?

# Question GATE-2004

- 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.

unacademy

# Happy Learning.!