Operating System

Additional Question

(Part-A)

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

Question 1: The following program consists of 3 concurrent processes and 3 binary semaphores.

The semaphores are initialized as, S0 = 1, S1 = 1, and S2 = 0.

Process P0	Process P1	Process P2

While (true) {
Wait (S0);
Print '0'
Release (S1);
Release (S2);
}
	Wait (S1);
	Release (S0);
		Wait (S2);
		Release (S0);

How many times will process P0 print '0'?

1. At least twice
2. Exactly twice
3. Exactly thrice
4. Exactly once

Solution (Question 1)

The semaphores are initialized as S0=1, S1=0, S2=0.

Because S0 =1 then P0 enter into the critical section and other processes will wait until either S1=1 or S2 =1

The minimum number of times 0 printed:

- S0 =1 then P0 enter into the critical section
- **print '0'**
- then release S1 and S2 means S1 =1 and s2 =1
- now either P1 or P2 can enter into the critical section
- if P1 enter into the critical section
- release S0
- then P2 enter into the critical section
- release S0
- P1 enter into the critical section
- **print '0'**

The minimum number of time **0 printed** is **twice** when executing in this order **(p0 -> p1 -> p2 -> p0)**

The Maximum number of times 0 printed:

- S0 =1 then P0 enter into the critical section
- **print '0'**
- Then release S1 and S2 means S1 =1 and s2 =1
- Now either P1 or P2 can enter into the critical section
- If P1 enter into the critical section
- Release S0 means S0 =1
- S0 =1 then P0 enter into the critical section
- **print '0'**
- Then P2 enter into the critical section
- Release S0 means S0 =1
- S0 =1 then P0 enter into the critical section
- **print '0'**

Maximum no. of time **0 printed** is **thrice** when execute in this order **(p0 -> p1 -> p0 -> p2 -> p0)**

So, **At least twice** will process P0 **print '0'**

Question 2: Each Process P_i, i= 1.....9 is coded as follows

```
repeat
   P(mutex)
   {Critical section}
   V(mutex)
forever
```

The code for P_{10} is identical except it uses V(mutex) in place of P(mutex). What is the largest number of processes that can be inside the critical section at any moment?

A) 1

B) 2

C) 3

D) None of above

Solution(Question 2):

```
repeat
    P(mutex)
    {Critical section}
    V(mutex)
forever
```

Now, let me say P_1 is in Critical Section (CS)
then P_{10} comes executes the CS (up on mutex)
now P_2 comes (down on mutex)
now P_{10} moves out of CS (again binary semaphore will be 1)
now P_3 comes (down on mutex)
now P_{10} come (up on mutex)
now P_4 comes (down on mutex)
So, if we take P_{10} out of CS recursively all 10 process can be in CS at same time using Binary semaphore only.

Question 3: 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.

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. $S_1 = 0; S_2 = 1; S_3 = 1$

Question 3: 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.

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. $S_1 = 0; S_2 = 1; S_3 = 1$

Solution (Question 3):

- Given threads are T1, T2, and T3, and three binary semaphore variable is used for synchronization S1, S2, and S3.
- In order to get the required output, only semaphore S1 should be initialized to 1, other semaphores should be initialized to 0.
- If we initialize S3 or S2 has 1, then it may start with T3 or T2 So those S2 and S3 are must be zero only.
- The first element in this sequence is 'B'. It means thread T2 should execute first.
- Thus, at this moment. S1 = 1, S2 = 0, S3 = 0.

- Given sequence need to print, BCABCABCA...

Hence the correct answer is $S_1 = 1; S_2 = 0; S_3 = 0$.

Question 4: Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S1 and S2 are randomly assigned.

Method used by P1	Method used by P2
While (S1 == S2); Critical Section S1 = S2;	While (S1 != S2); Critical Section S2 = not (S1);

Which one of the following statements describes the properties achieved?

A Mutual exclusion but not progress

B Progress but not mutual exclusion

C Neither mutual exclusion nor progress

D Both mutual exclusion and progress

Solution (Question 4):

- It can be easily observed that the Mutual Exclusion requirement is satisfied by the above solution, P1 can enter critical section only if S1 is not equal to S2, and P2 can enter critical section only if S1 is equal to S2.

- But here Progress Requirement is not satisfied. Suppose when s1=1 and s2=0 and process p1 is not interested to enter into critical section but p2 want to enter critical section. P2 is not able to enter critical section in this as only when p1 finishes execution, then only p2 can enter (then only s1 = s2 condition be satisfied). Progress will not be satisfied when any process which is not interested to enter into the critical section will not allow other interested process to enter into the critical section.

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition (Pearson Publications), 2014.

3. https://www.geeksforgeeks.org/

4. https://www.javatpoint.com/

5. https://www.tutorialspoint.com/

6. https://www.nesoacademy.org/

7. https://www.baeldung.com/

8. https://www.educative.io/

9. https://testbook.com