RADHIKA PATWARI

18CS10062

PAGE NO.
DATE: 17/3/2021

Class test 2 [OS]

Question 1:

1.1 (v) ab → first child process (fork()==0) executes then parent process executes

(vi) ba → first parent process executes, then child process executes.

1.2 Asynchronous threading is used. process does not wait for threads to complete. Both main process and threads share same file descriptor pointers. So changes of both will be visible on out.txt

Possible contents

(v) ab , (vi) ba

as only one thread is created and order of execution of thread and main process may vary.

1.3 (v) ab (vi) ba

as thread and main process may execute in any order and they are asynchronous.

1.4 buffer is a global variable shareable by both main process and thread. Process does not wait for thread to complete (asynchronous) and both change the value of 'buffer'

Possible outputs :-

(V) ab → if threads executes first, then process executes.
thread writes 'a' then process writes 'b'.

(vi) ba → process writes 'b' then thread writes 'a'

(iv) aa → buffer = 'b' is run, then suddenly thread executes & changes buffer = 'a'. thread then writes 'a' and process also writes 'a' as buffer is global variable

(vii) bb → buffer = 'a' is run by thread, then process executes and changes buffer to 'b'. Process writes 'b', then thread writes 'b'. So 'bb' is possible as buffer is global variable.

Question 2.

2.1 S0 = 1, S1 = 0, S2 = 0

3 times P0 can print '0'

First, P0 executes S0 = 0, S1 = 0, S2 = 0
P0 prints 0
P0 signals S1 S0 = 0, S1 = 1, S2 = 0
P0 signals S2 S0 = 0, S1 = 1, S2 = 1
then P1 executes S0 = 1, S1 = 0, S2 = 1

As P0 is in while loop, prints '0'. ; $S0=0, S1=0, S2=1$

Then P2 executes $S0=1, S1=0, S2=0$

Again P0 executes $S0=0, S1=1, S2=1$
and prints '0'.

But P1 & P2 has completed execution. So at max, 0 can be printed 3 times.

2.2

No. Deadlock may occur between consumer and producer.

Let consumer() run,
it executes wait(s),
New values $n=0, s=0$,
So now both semaphores are 0 and consumer stops at wait(n) and busy waits/get blocked

Now producer() runs and it get blocked at wait(s) as $s=0$.

So consumer is waiting at semaphore n
producer is waiting at semaphore s.

Only producer can set $n=1$ but it is already blocked.

→ Hence deadlock occurs

Question 2

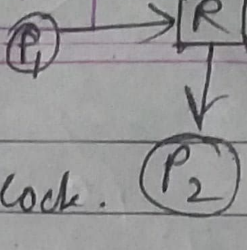
2.3

```
P1 : while(true) do {
        wait (SX);
        wait (SY);
        X = X + 10 ;
        Y = Y - 20;
        signal (SX) ;
        signal (SY);

     }
P2 : while (true) do {
        wait (SX);
        wait (SY);
        Y = Y + 20;
        X = X - 10 ;
        signal (SX);
        signal (SY);

     }
```

Question 3.



3.1 For single instance of resources, presence of a cycle denotes deadlock.

Check for any request of resource R_j by process P_i, if

- the allocation results in the formation of cycle, does not do this allocation and let P_i wait.

- If the resource R_j is not available, then let P_i wait.

3.2

	Allocated	Max	Need.	Available
A :	10 211	11213	01002	0 0 x 1 1
B :	20 110	22210	02100	
C :	11010	21310	10300	
D :	11110	11221	00111	

Whatever maybe value of X, process A can never be scheduled because 5th resource type available is 1 and only A has hold 1

So ~~total~~ 5th resource type ~~8~~ = 2.

But A requires 3 in maximum.

So no value of X is possible for a safe state of 4 processes

3.3

Given $\sum_{I=1}^{N} need_i < M+N$

Now let P_1 have need of M.

So allocate all M resources to P_1 and let P_1 complete. Then M resources are released. Again allocate M to P_2.. and so on.

So it is always possible to obtain a safe sequence

$\langle P_1, P_2 .. P_N \rangle$ for which need of $P_i \leq M$ and can ~~f~~ be fulfilled by available resources as one process can hold only 1 resource at a time.

available = M

Only one process can be reserved at any point.

Question 4

4.1 Max value of var = 390 for order P_2, P_3, P_1

Min value of var = 310 for order P_1, P_3, P_2

4.2 a) S1 = 0
S2 = 1
S3 = 0 } Yes, for this value, it is possible

At any time only one semaphore is 1, while others are 0.

when L3 executes, start → S1=0, S2=1, S3=0
then end → S1=1, S2=0, S3=0

when L2 executes, start → S1=1, S2=0, S3=0
then end → S1=0, S2=0, S3=1

when L1 executes, start → S1=0, S2=0, S3=1
then end → S1=0, S2=1, S3=0

So at any time, only semaphore is available.

b) Let sequence of execution be L3, L3, L1

So for 1st execution of L3,
we have output = 1, S1=3, S2=5, S3=1

for 2nd execution of L3,
we have output = 11, S1=4, S2=4, S3=1

for 3rd execution of L1,
we have output = 112, S1=4, S2=5, S3=0

Hence output = 112

Now for another 2 to be followed, L1 should execute.
But S3 = 0, only L2 can signal S3.

But if L3 executes, output becomes 1126...

Hence it is not possible to obtain output with starting sequence 1122622...