Operating System and System Programming
T2- Examination (Solution)
Odd-2022

Q.1. [**5 Marks**] Suppose there are 3 parallel running processes. They all share a variable D. Read-write operations are performed on D.

P1	**P2**	**P2**
------ ------ D=D +20 ------ ------	------ ------ D=D - 50 ------ ------	------ ------ D=D +10 ------ ------

a) The processes are executed on a uniprocessor system running a time-shared operating system. If the minimum and maximum possible values of D after the three processes have completed execution are X and Y respectively, then the value of Y - X is __________. (Initialize D =100)

b) Let the processing environment is multiprocessing then write a semaphore solution (pseudo code) to ensure the execution order P2, P1, and P3. Take the appropriate variable and initialized.

Solution:

a) [**Strict Marking, No PARTIAL MARKING Award 2.5 mark for correct value of X & Y only else 0**]

Minimum value of D is X=50, Maximum value of D is Y =130, **Y-X=130-50=80**

b) [**Strict Marking, No PARTIAL MARKING Award 2.5, Variable Initialization is a must**]

Semaphore initialization: S1=1, S2=0, S3=0		
P1	**P2**	**P2**

While(true)
{
Wait(S3)
D=D +20
Signal(S2)
}
	While(true)
	{
	Wait(S1)
	D=D -50
	Signal(S3)
	}
		While(true)
		{
		Wait(S2)
		D=D + 10
		Signal(S1)
		}

Q.2. [**5 Marks**] Consider the following snapshot of a system in which four resources *A*, *B*, *C*, and *D* are available. The system contains a total of 1 instance of *A*, 5 of resource *B*, 2 of resource *C*, and 2 of resource *D.*

	Allocation				**Max**				**Available**			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P_0	0	0	1	1	0	0	1	1	1	5	2	2
P_1	1	0	0	1	1	7	5	1				
P_2	1	3	5	1	2	3	5	2				
P_3	0	5	3	1	1	6	5	2				
P_4	0	0	1	1	5	6	5	1				

Answer the following question using Banker's algorithm.

1. What is the content of the need matrix? [**1- Mark for correct Need Matrix]**
2. Is the system in a safe state? If the system is safe, show how all the process could complete them execution successfully. If the system is unsafe, show how deadlock might occur. Explain. [**2- Mark All calculation is required]**
3. If a request from P_1 arrives for (0,3,2,1), can the request be granted immediately? If yes or no write the sequence of step. [**2- Mark , Proper justification is required]**

SOLUTION:

1.

	Allocation				**Max**				**Need**				**Available**			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P_0	0	0	1	1	0	0	1	1	0	0	0	0	1	5	2	2
P_1	1	0	0	1	1	7	5	1	0	7	5	0				
P_2	1	3	5	1	2	3	5	2	1	0	0	1				
P_3	0	5	3	1	1	6	5	2	1	1	2	1				
P_4	0	0	1	1	5	6	5	1	5	6	4	0				

2.

a. P0 need (0 0 0 0) less than available (1 5 2 2)
   New available = (1 5 2 2) +(0 0 1 1) = (1 5 3 3)
b. P1 need (0 7 5 0) greater than available (1 5 3 3)
   New available = (1 5 3 3)
c. P2 need (1 0 0 1) < available (1 5 3 3)
   New available = (1 5 3 3) +(1 3 5 1) = (2 8 8 4)
d. Need of P3 (1 1 2 1) < available (2 8 8 4)
   New available = (2 8 8 4) +(0 5 3 1) = (2 13 11 5)
e. Need of P4 (5 6 4 0) > available (2 13 11 5)
   New available = (2 13 11 5)
   The system is in an unsafe mode, request of P4 is greater than the resource available with the system. So the system is in deadlock with process P1 and P4.

3. request from process P1 is (0 3 2 1)

If **request of P1 (0 3 2 1) <= need of P1 (0 7 5 0)** and **request of P1(0 3 2 1) <= available (1 5 2 2)**

Since the condition request of P1 <= need of P1 is false, the request from process P1 is not granted.

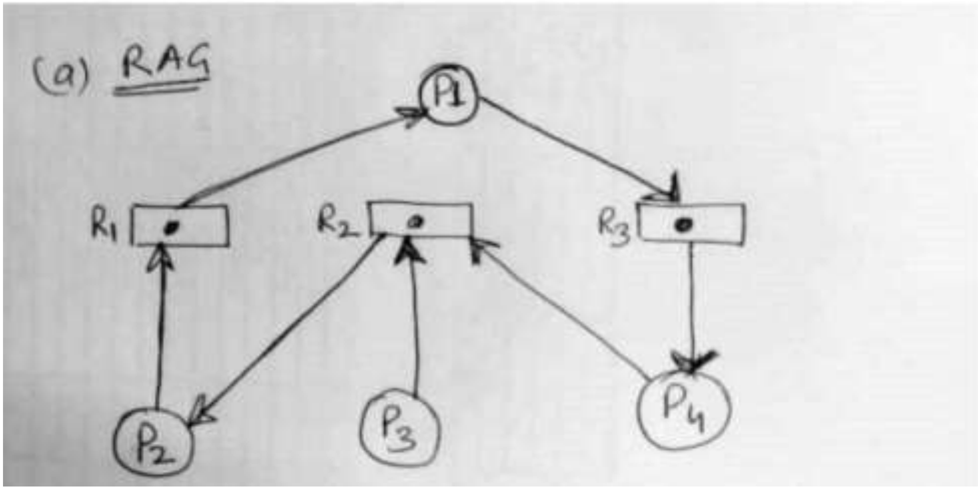
Q.3 Consider the following scenario: **[2 Marks**] [**CO-2**]

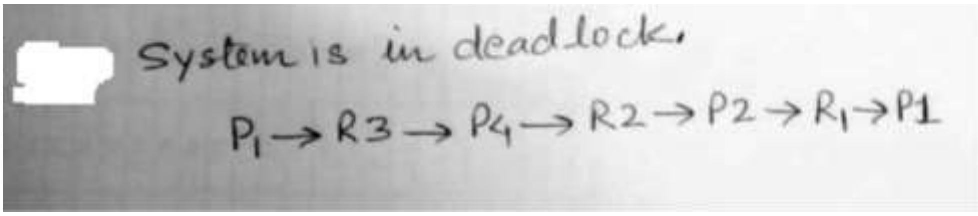
Suppose there are four processes, P1, P2, P3 and P4 and three resources R1, R2 and R3.

- P1 is requesting R3 and holding R1.
- P2 is requesting R1 and holding R2.
- P3 is requesting R2 and holding none.
- P4 is requesting R2 and holding R3

a. Construct a Resource allocation graph (RAG) for the above scenario. [**1- Mark]**
b. Find out whether the given scenario is in deadlock or not? **[1- Mark if part 'a' is correct else zero]**

Solution





Q.4. [**4 Marks**] A computer system uses 42-bit physical address space and pages that are 16 KB each to store data. Every entry in the page table has a page number along with valid or invalid (1 bit), dirty (2 bits), and read/write (1 bit). Answer the following questions:

1. What is the length of the virtual address space (in bytes) that the system may handle if the maximum size of the process page table is 64 GB? [**2- Mark]**
2. What is the size of an inverted page table? (Assuming 30 bits for a process ID). [**1- Mark]**
3. What is the effective memory access time if TLB access time is 35 milliseconds and memory access time is 150 milliseconds (Assuming all pages are in TLB)? [**1- Mark]**

Solution:

[Strict Marking-Complete calculation must be written.]

1. PA= 42 bit, PS= 16 KB= 214 , Page table size= 64 GB
   Page table entry = Frame number (?) + valid/invalid (1) + dirty bit (2) +read/write (1)
   No of frames= $2^{42}/2^{14} = 2^{28}$
   Page table entry size= 28+1+2+1= 32 bit= 4 B
   No of page table entries= 64 GB / 4 B =16 GB
   Size of virtual address space = No of pages * page size = 16 GB *16 KB = 2^{48}
   Inverted page table entry size= Process id + page number = 30+34 = 64 bit = 8 B
   Size of inverted page table= Number of frames * inverted page table entry size
   = 2^{28} * 8 B = 231 = 2 GB
2. EMAT= 35+150= 185 ms (100% TLB hit)

Q.5. [4 Marks] Consider the following scenario. Counselling is being conducted in LT 2 of JIIT Campus for CSE branch (assume that enough seats are available). LT 2 can hold N candidates. The number of candidates who wish to get admission are waiting in LT 3 having the seating capacity larger than N. The counselling being conducted in following manner. Whenever JIIT authority is ready for counselling, it opens front door of LT 2 and waits for the candidates to fills completely. Once candidate finishes with counselling, the backdoor of LT 2 is opened and the candidates are let out. Once all the candidate residing in LT 2 have exited, another batch of N candidate is admitted again through the front door. This process continues indefinitely.

We model the available branch seats and the JIIT as threads in a multithreaded program. The threads must be synchronized as follows. A candidates cannot allow for the counselling until the JIIT authority has opened its front door of LT 2. The JIIT cannot start the counselling service until N candidates have come in., The candidates cannot exit until the back door is open. The JIIT cannot close the backdoor and prepare for the next batch until all the candidates of the previous batch have left.

Note: You can only use the following variables and function in your solution.
semaphore variables: mutex_enter, mutex_exit, enter_candidate, exit_candidate, enter_LT2, exit_LT2.

Counter variables: count_enter, count_exit.
Unsynchronized code for LT2:
OpenFrontDoor()
CloseFrontDoor()
Conduct_counselling()
OpenBackDoor()
CloseBackDoor()

Unsynchronized code for LT2:
Computer_Allocation()
Choice_Filling()
Computer_Deallocation()

Solution

1. write down the complete synchronized code for LT2.

```
OpenFrontDoor()
        do N times: V(enter_candidate)      //do N times
        P(enter_LT2)
CloseFrontDoor()
        Conduct_counselling()
OpenBackDoor()
        do N times: V(exit_candidate)
        P(exit_LT2)
CloseBackDoor()
```

Award **2- Mark for correct code**
Check alternative solution also

2. write down the complete synchronized code for choice filling.

```
P(enter_candiate)
        Computer_Allocation()
P(mutex_enter)
        count_enter ++
        if(count_enter == N)
        {
          V(enter_LT2)
          count_enter  = 0
        }
V(mutex_enter)
        choice_filling()
P(exit_candidate)
        Computer_Deallocation()
P(mutex_exit)
        count_exit++
        if(count_exit == N)
        {
        V(exit_LT2)
        count_exit = 0;
        }
V(mutex_exit)
```

Award **2- Mark for correct code**
Check alternative solution also

Alternative solution for you reference:

1. write down the complete synchronized code for LT2.

```
OpenFrontDoor()
        V(enter_candidate)
        P(enter_LT2)
CloseFrontDoor()
        Conduct_counselling()
OpenBackDoor()
        V(exit_candidate)
        P(exit_LT2)
CloseBackDoor()
```

2. write down the complete synchronized code for choice filling.

```
P(enter_candiate)
        Computer_Allocation()
P(mutex_enter)
        count_enter ++
        if(count_enter < N)
        {
          V(enter_candidate)
        } else if(count_enter == N)
        {
        V(enter_LT2)
         count_enter  = 0
        }
V(mutex_enter)
        choice_filling()
P(exit_candidate)
        Computer_Deallocation()
P(mutex_exit)
        count_exit++
        if(count_exit < N)
        {
        V(exit_LT2)
        }
        else if(count_exit == N)
        {
        V(exit_LT2)
        count_exit = 0;
        }
V(mutex_exit)
```