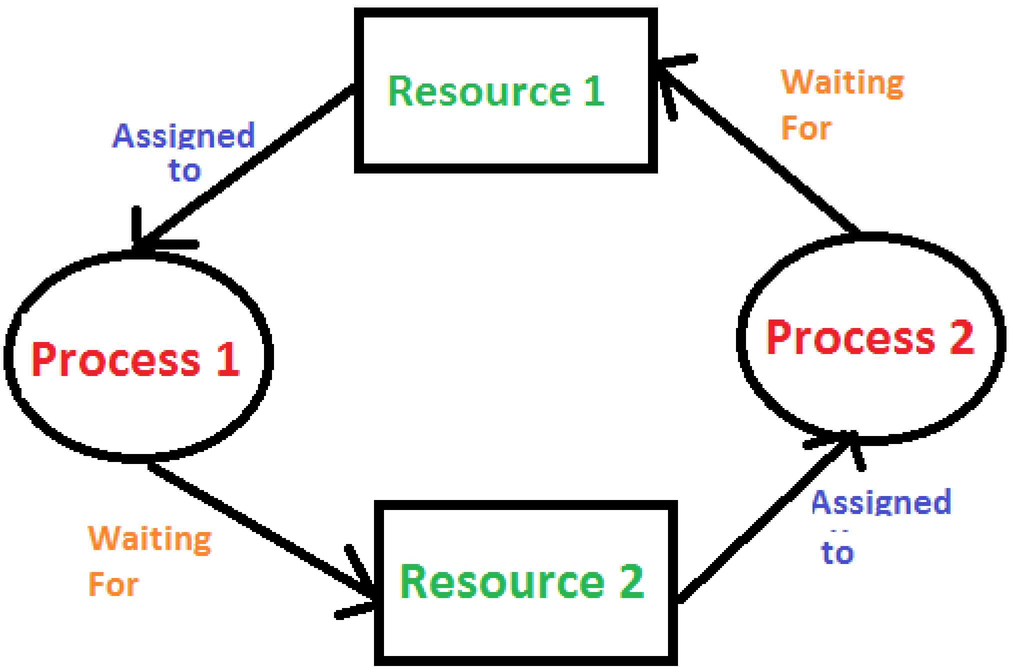
Sina Ziaee 97521387

Question 1

Processes in operating systems use resources in this manner:
1) requesting for resources
2) using those requested resources after assignment
3) releasing those used resources
Deadlock is a situation in which a set of processes are blocked because each process is holding some resources and is waiting for other resources that is held by other process.(in other words none of the processes is running as of this matter that can not access all of it’s required resources because they are held by others)



for example in the above example as we can see we have a cycle between 2 processes and 2 resources and none of them is going to be finished because of the dependency on a resources that is not currently available.

Question 2)

- Mutual Exclusion:
only one process at a time can access a resource
- Hold and wait:
processes holding resources are waiting for other resources held by other processes
- No preemption:
a resource is only released when a process holding it is completed and then it's going to release it's held resources.
- circular wait:
as mentioned in above Picture there are resources and processes inside a cycle.

Question 3)

1- Assuring that a deadlock is never going to happen
a) prevention
The strategy of deadlock prevention is to design the system in such a way that the possibility of deadlock is excluded
b) avoidance
This approach allows most of the necessary conditions of deadlocks but with the help of a prior knowledge of future process requests, tells if a resource assignment is granted or denied
2- Allowing deadlocks to occur and then detecting and removing them
Deadlock detection is used by employing and algorithm that tracks the circular waiting and killing one or more processes so that deadlock is removed
3- Ignoring the fact that they actually happen and do nothing about them

Question 4)

System is in safe state if there exists a safe sequence of all processes.

Sequence <P1, P2, Pn> is safe if for each Pi, the resources that Pi can still request can be satisfied by currently available resources + resources held by all the Pj

For example imagine

5 processes P_0 through P_4;

3 resource types:

A (10 instances), *B* (5 instances), and *C* (7 instances)

Snapshot at time T_0:

	Allocation	*Max*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

	Need
	A B C
P_0	7 4 3
P_1	1 2 2
P_2	6 0 0
P_3	0 1 1
P_4	4 3 1

The system is in a safe state since the sequence $< P_1, P_3, P_4, P_2, P_0>$ satisfies safety criteria

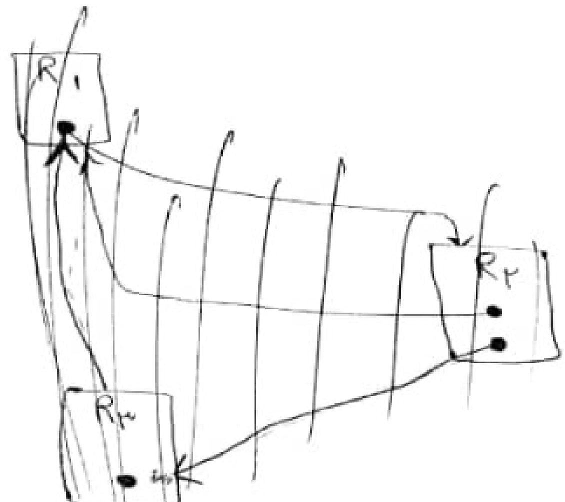
So As we see in above example if we processed the above sequence we can reach a safe state still.

Question 5)

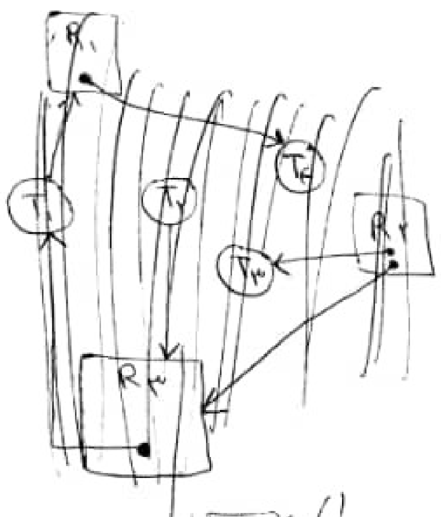
Processes: T_1, T_2, T_3, T_4

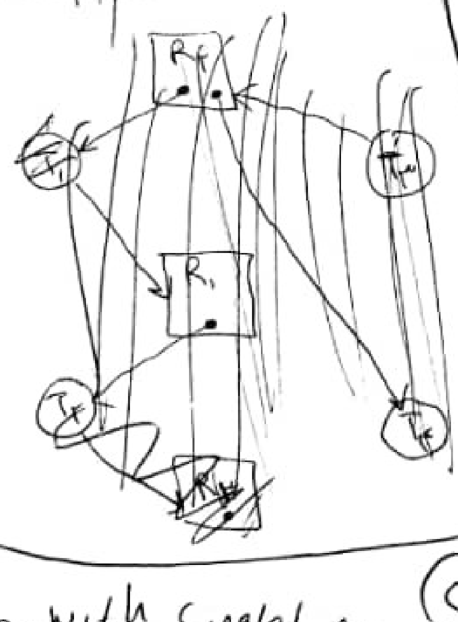
Resources: $\underbrace{R_1}_{1}, \underbrace{R_2}_{2}, \underbrace{R_3}_{1}$

Question 5

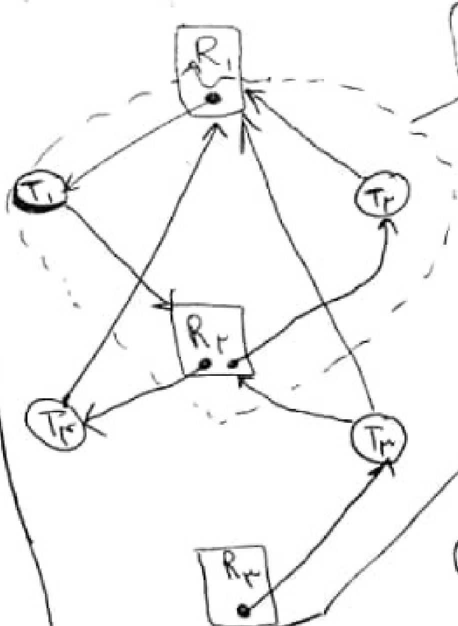


Dead Lock Occurs



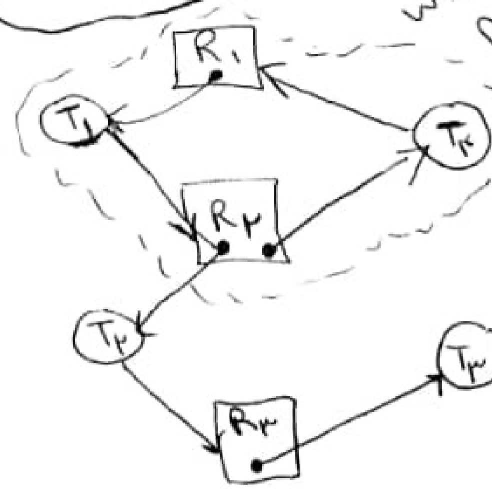


Cycle and deadlock



(a)

Cycle without Dead lock



(b)

(c)

No, with singled resource instance and cycle, we will always have deadlocks

Question 6)

Available resources: 3, 2, 2

	Allocated			Max			Need		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	0	0	1	8	4	3	8	4	2
P2	3	2	0	6	2	0	3	0	0
P3	2	1	1	3	3	3	1	2	2

a) P1 → 0, 0, 2 , Available resources → 3, 2, 0

	Need		
	R1	R2	R3
P1	8	4	0
P2	3	0	0
P3	1	2	2

We cannot continue with P1 as we do not have enough resources but we can proceed with P2 (available resources >= Need(p2))
So that P2 is finished and releases it's allocated resources:

P1 → 3, 0, 0 , Available resources → 0, 2, 0 ,
after finished available resources are → 6, 4, 0

	Need		
	R1	R2	R3
P1	8	4	0
P2	0	0	0
P3	1	2	2

but we can not continue further more as we can't continue with either P1 or P3 as we do not have enough resources to assign them. So we can not reach a safe State with this approach.

(available resources < Need(p1)) & (available resources < Need(p3))

Available resources: 3, 2, 2

	Allocated			Max			Need		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	0	0	1	8	4	3	8	4	2
P2	3	2	0	6	2	0	3	0	0
P3	2	1	1	3	3	3	1	2	2

b) P2 → 2, 0, 0 , Available resources → 1, 2, 2

	Need		
	R1	R2	R3
P1	8	4	2
P2	1	0	0
P3	1	2	2

So now we have a safe state and we can choose both P2 and P3 to continue, let’s choose P2 (available resources >= Need(p2))

	Need		
	R1	R2	R3
P1	8	4	2
P2	0	0	0
P3	1	2	2

P2 → 1, 0, 0 , Available resources → 0, 2, 2 ,
after finished available resources are → 6, 4, 2

It’s a safe state as we can now continue with P3 So
(available resources >= Need(p3))

P3 → 1, 2, 2, Available resources → 5, 2, 0,
after finished available resources are → 8, 5, 3

	Need		
	R1	R2	R3
P1	8	4	2
P2	0	0	0
P3	0	0	0

It's a safe state as now we can choose P1 to continue
(available resources >= Need(p1))

P1 → 8, 4, 2 , Available resources → 0, 0, 1

	Need		
	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0

So both " a " is illegal and " b " is legal

Question 7)

Available resource → 0, 1, 2, 1

	Allocated				Max				Need			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	2	0	0	2	3	1	1	2	1	1	1	0
P2	3	0	1	0	3	0	1	0	0	0	0	0
P3	4	2	0	0	5	5	5	3	1	3	5	3
P4	1	3	2	0	6	4	4	3	5	1	2	3

a)
P2 is finished and releases it's resources
Available resource → 3, 1, 3, 1

Available resources >= Need (P1)

P1 → 1, 1, 1, 0 , Available resources → 2, 0, 2, 1
after finished available resources are: 5, 1, 3, 3

	Need			
	R1	R2	R3	R4
P1	0	0	0	0
P2	0	0	0	0
P3	1	3	5	3
P4	5	1	2	3

Safe state, we can continue with P4

P4 → 5, 1, 2, 3 , Available resources → 0, 0, 1, 0
after finished available resources are: 6, 4, 5, 3

	Need			
	R1	R2	R3	R4
P1	0	0	0	0
P2	0	0	0	0
P3	1	3	5	3
P4	0	0	0	0

Safe state, we can continue with P3

P3 → 1, 3, 5, 3 , Available resources → 5, 1, 0, 0
after finished available resources are: 10, 6, 5 3

	Need			
	R1	R2	R3	R4
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0

So We found a sequence of safe states like this:

<P2, P1, P4, P3 >

b)
R1 →10
R2 → 6
R3 → 5
R4 → 3