Chapter 7: Deadlock:

2070 Bhadra:

Q. What is deadlock? State the necessary conditions for deadlock to occur. Give reasons, why all conditions are necessary. [10]

Ans:

Deadlock is the state where none of the processes can:

- Run
- Release resources
- Be awakened

Deadlock occurs when the process in a set are in simultaneous wait state for the release of resource held exclusively by one of the waiting process in the set.

The necessary conditions for deadlock are:

i) Mutual Exclusion:

At least one resource is held in a non-sharable mode that is only one process at a time can use the resource. If another process request that resource, the requesting process must be delayed until the resource has been released.

ii) Hold & Wait:

There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other resources.

iii) No Preemption.

Resources cannot be preempted, that is, a resource can only be released voluntarily by the process holding it, after the process has completed the task.

Circular Wait.

There must exist a set $\{P_0, P_1, \ldots P_n\}$ of waiting process such that P_0 is waiting for a resource which is held by P_1, P_1 is waiting for a resource which is held by P_2 ... P_{n-1} is waiting for a resource which is held by P_n and P_n is waiting for a resource which is held by P_0

All the conditions are necessary for deadlock to occur. If any of the condition is not satisfied then, the resource may be free of any process i.e. the process release the resource and new ~~resource~~ process can be allocated that resource and deadlock won't occur.

2070 Magh.

Q. Consider a system consisting of m resource of the same type, being shared by n process. Resource can be requested and released by process only one at a time. Show that the system is deadlock free if the following two conditions hold:

a) The maximum need of each process is between 1 and m resources.

b) The sum of all maximum needs is less than m+n.

Ans: Let:

N = Summation of all Need$_i$; P_i = Process i

A = Summation of Allocation$_i$

M = Summation of all Max$_i$

Now,

DATE :

Given: Condition:

(a) The maximum need of each process is between 1 and m resources.

If the system is assumed to not be deadlock free and there exists a deadlock state, then $A \geq m$ because there is only one resource which can be requested/released one at a time.

(b) The sum of all maximum need is less than m+n.

In this condition, $M < m+n = N+A$

So, we get $N+m < m+n$

i.e. $N < n$.

It shows that at least one process i that $Need_i = 0$.

From condition (a), this process can release at least one resource, so there are n-1 process sharing n resources at this point, and both condition (a) and (b) still hold true. No process will wait permanently so there is no deadlock.

2071 Bhadra:

Q. Consider a system with 5 processes P_0 through P_4 and three resources types A, B, C. Resources type A has 7 instances, B has 2 and C has 6 instances. Suppose at to time we have the following state:

a. Is the given system in deadlock state?

b. Suppose P2 makes an additional request (0, 0, 1), what'll be the effect of this request to the system.

Process	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0			
P1	2	0	0	2	0	2	0	0	0
P2	3	0	3	0	0	0			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	2			

a.

Solution,

The need matrix is :- [Need : Max.Req - Allocation]

	A	B	C
P0	0	-1	0
P1	0	0	2
P2	-3	0	-3
P3	-1	-1	-1
P4	0	0	0

Finish [n]

(1)	(5)	(2)	(3)	(4)
P0	P1	P2	P3	P4
F	F	F	F	F
T	T	T	T	T

Work matrix is: (Available Initially)

0	0	0

1) Finish P0 = f and Need P0 < Work. So, P0 executes. When P0 completes execution, update work.

Work matrix: | 0 | 1 | 0 | [Work = Work + Allocation]

DATE:

2.) For P1, Finish P1 = F but Need P1 ≤ Available is not satisfied. So, resource is not given to P1.

3.) For P2, Finish P2 = F and Need P2 ≤ Available. So, P2 executes. When P2 completes execution, update work.

Work is:

3	1	3

4.) For P3, Finish P3 = F and Need P3 ≤ Available. So, P3 executes. When P3 completes execution, update work.

Work is:

5	2	4

5.) For P4, Finish P4 = F and Need P4 ≤ Available. So, P4 executes. When P4 completes execution, update work. Work is:

5	2	6.

6.) For P2 P1, Finish P1 = F and Need P1 ≤ Available. So, P1 executes. When P1 completes execution, update Work. Work is:

7	2	6.

(a) No, the system is not in deadlock state since all the process executes.

(b) Since, the request (0, 0, 1) is more than available (0, 0, 0). It can't be granted immediately.

2071 Magh.

Q. Explain the necessary condition of deadlock? How can a system detect deadlock and what does it do after detection?

Ans: 1st question: 2070 Bhadra.

The system can detect a deadlock as:

i.) Deadlocks detection for single instance of each resource type:

For single instance of each resource type, deadlock can be detected by construction wait-for graph from resource allocation graph.

If cycle exist in wait-for-graph then deadlock exist otherwise deadlock doesn't exist.

E.g:

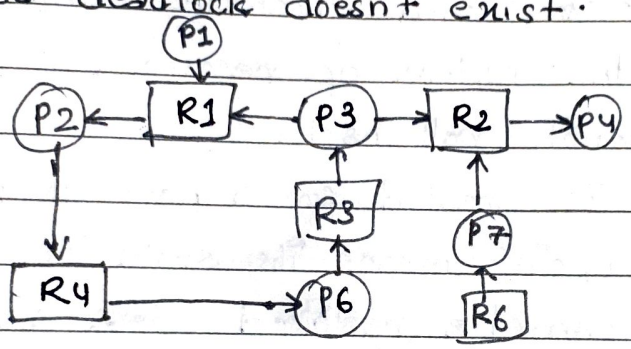

Fig: Resource allocation graph.

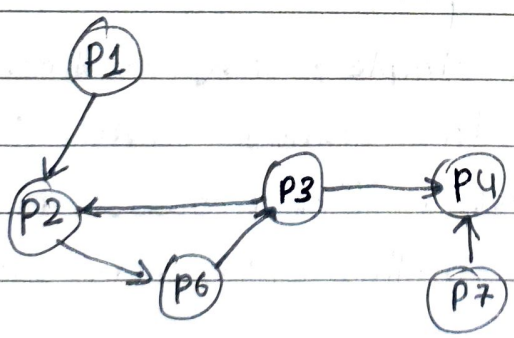

Fig: Wait-for graph.

Here, since cycle exists in wait-for graph. Deadlock exist.

ii.) Deadlock detection for multiple instance of a resource type:

DATE:

If there are multiple instances of a resource type, the wait-for graph cannot detect deadlock. So, for this we use algorithm similar to Banker's algorithm to detect deadlock.

Once the deadlock has been detected in the system, different methods are applied to recover the system from deadlock and continue with processing. The methods are:

i.) Recovery through preemption:
The resource is temporarily taken away from its current owner and given to other process. It depends on the nature of resource

ii.) Recovery through roll back:
- A checkpoint is created for a process periodically.
- Creating checkpoint means its state is written to a file so that it can be ~~restored later~~. restarted later.
- If a deadlock is found, the process is restarted

iii.) Recovery through killing process.
It is the crudest and simplest way to break deadlock. A process is killed in the deadlock state. The other process gets its resources. The process that will yield up no ill effect to entire system is chosen.

2072 Ashwin

Q. What is deadlock? Explain the essential condition of deadlock. How do you detect the deadlock. Explain with examples.

Ans: 1st + 2nd question: 2070 Bhadra.
3rd question: 2071 Magh

PAGE NO.:
DATE:

2072 Magh:

Q. What is deadlock avoidance and detection? Explain all possible deadlock prevention techniques.

Ans. Deadlock avoidance can be achieved by never allowing allocation of resources to a process if it will lead to a deadlock. It is done by being careful at the time of resource allocation. The system must be able to decide whether granting a resource is safe or not, any only make allocation when it is safe.

Deadlock detection is the process of detecting the deadlock in the system if it is not prevented or avoided.

The possible deadlock preventing techniques are:

1. Denying the mutual exclusion condition.
2. Denying the Hold and Wait condition.
   It can be done by allocating all the required resources to a process before the start of its execution this way hold and wait condition is eliminated but it will lead to low device utilization.
3. Denying the No-Preemption condition:
   Preempt resources from the process when resources are required by other high priority process.
4. Denying the circular wait condition:
   A process is entitled to only one resource at a time or at a moment. If it needs a second one it must release a second one.

PAGE NO.:
DATE:

2073 Bhadra

Q. What is the difference between deadlock and idefinite postponment? Consider a system with 5 concurrent process (P0, P1, P2, P3, P4) and 4 resource type (R0, R1, R2, R3). The no. of instances of each resource type in the system are (6, 4, 4, 2) respectively. Is the state safe? Show the execution of the process.

Allocation:

	R0	R1	R2	R3
P0	2	0	1	1
P1	1	1	0	0
P2	1	1	0	0
P3	1	0	1	0
P4	0	1	0	1

Maximum claim:

	R0	R1	R2	R3
P0	3	2	1	1
P1	1	2	0	2
P2	1	1	2	0
P3	3	2	1	0
P4	2	1	0	1

Ans:

Deadlock is a situation in which two or more competing actions are each waiting for the other to finish and thus neither ever does. If two processes are in deadlock, it is not possible for them to ever do any useful work - because they depend on one another, and neither will ever yield.

Indefinite postponement is to delay indefinitely the scheduling of a process while other process receive the system's attention. If a process is postponed indefinitely, it is at least theoretically possible for such process to continue and do some useful work at some time in the future.

Here,

Available = $(6-(2+1+1+1), 4-(1+1+1), 4-(1+1), 2-(1+1))$

= $(1, 1, 2, 0)$

The need matrix is:

	R0	R1	R2	R3
P0	1	2	0	0
P1	0	1	0	2
P2	0	0	2	0
P3	2	2	0	0
P4	2	0	0	0

Finish [n]

(4)	(5)	(1)	(2)	(3)
P0	P1	P2	P3	P4
F	F	F	F	F
T	T	T	T	T

Initially Work : Available.

1	1	2	0

1.) For P0, Finish P0 = F but Need ≤ Work is not satisfied. So, resource is not given to P0.

2.) For P1, Finish P1 = F but Need P1 ≤ Work is not satisfi So, resource is not given to P1.

3.) For P2, Finish P2 = F, and Need P2 ≤ Work. So, P2 executes and work is updated.

Work:

3	2	2	0

4.) For P3, Finish P3 = F, and Need P3 ≤ Work. So, P3 is executed and work is updated.

Work:

5	2	3	0

5.) For P4, Finish P4 = F and Need P4 ≤ Work. So, P4 is

executed and work is updated.

Work

5	2	3	0

3	3	3	1

6.) For P0, Finish P0 = F and Need P0 ≤ ~~Allocation~~ Work. So, P0 executes and work is updated

Work:

6	4	3	0

5	3	4	2

7) For P1, Finish P1 = F and Need P1 ≤ Work. So, P1 executes and work is updated.

Work:

6	4	4	2

∴ The system is in safe state because all the process executes.

Sequence of execution:

P2, P3, P4, P0, P1. //

~~2073 Magh~~:

Q. 2074 Bhadra

Q. A system has 2 process and 3 resources. Each process need maximum of two resources. Is, deadlock possible? Explain.

Ans. For deadlock to occur, the following four conditions must be satisfied.

i) Mutual Exclusion.

ii) Hold & Wait

iii) No preemption

iv) Circular wait

DATE :

Suppose, two resources are provided to a process and one to another.

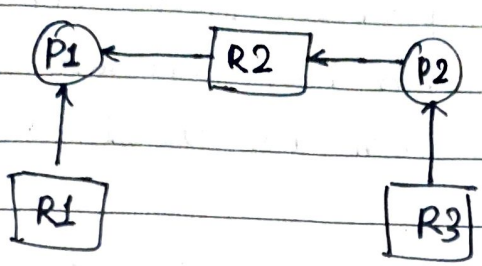

Fig: Resource Allocation Graph.

Now, since P1 has 2 resource as required, there is no hold and wait and also there is no circular wait. So, P1 can preempt resource R2 and P2 can use the resource. So, there is no deadlock possible.

2075 Bhadra

Q. Similar to 2073 Bhadra.