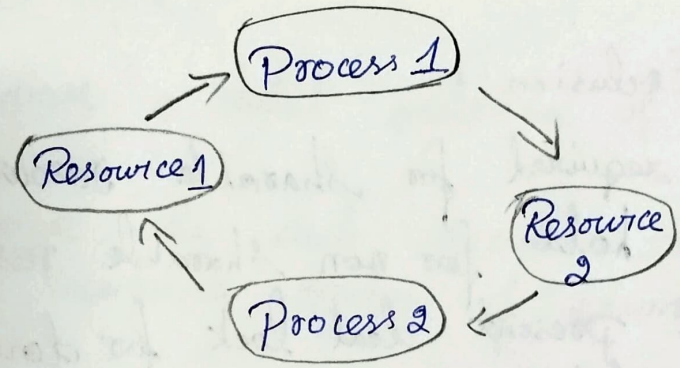
M. Sudharsan

DEADLOCKS...

A deadlock happens in Operating System when two or more process need some resource to complete their execution that is held by the process.



Deadlock characterization :-

In a deadlock, process never finish executing, and System resources are tied up, preventing others job from starting.

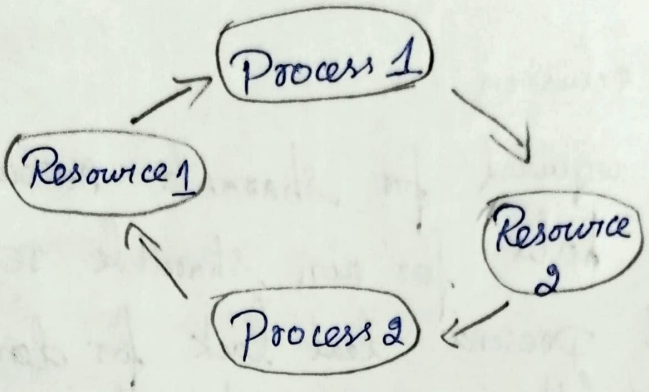
Deadlock Methods :-

generally Speaking, we can deal with the deadlock problem in one of 3 ways.

* We can use a protocal to prevent or avoid deadlock ensuring that the system will never enter a dead state.
* We can allow the system to enter a deadlock state, detect it, and recover.
* We can ignore the problem together and pretend that deadlocks never occurs in the system.

DEADLOCKS

A deadlock happens in Operating System when two or more process need some resource to complete their execution that is held by the Process.



Deadlock characterization :-

In a deadlock, process never finish executing, and System resources are tied up, preventing others job from starting.

Deadlock Methods :-

generally Speaking, we can deal with the deadlock problem in one of 3 ways.

* We can use a protocal to prevent or avoid deadlock ensuring that the System will never enter a dead state.
* We can allow the system to enter a deadlock state, detect it, and recover.
* We can ignore the problem together and pretend that deadlocks never occurs in the system.

Safe state:-

A safe is a state if it has a of Process and these are enough resources for the 1st process to be finished and after it releases it's resources there.

Banker's Algorithm:-

This algorithm is used on a banking System to ensure that the bank never allocate its available cash in such a way that is could no longer satisfy the needs of all its customer.

Resource - Request algorithm:-

1) If Request$_i$ ≤ need$_i$, go to step 2.
2) If Request$_i$ ≤ Available, go to step3.
3) Have the system pretend to have allocated the resources.

Available = Available + Request$_i$;
Need$_i$ = Need$_i$ - Request$_i$;

Eg: Consider a system with five process of through P_4

Resource type	Instance
A	10
B	5
C	7.

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0	0	0	0	0	
P_1	2	0	0	2	0	2			
P_2	3	0	3	0	0	0			
P_3	2	1	1	1	0	0			
P_4	0	0	2	0	0	2			

Suppose, that at time to the following.

	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 - Max - Allocation :-

Need.

Dead lock detection :-

* Single Instance of each Resource type
* Several Instance of a resource type
* Detection - Algorithm usage.

Starvation:-

Same process may always be picked as victim, include number of rollback in cost factor. Dead locks exist if and only if the wait-for-graph contains a cycle.

Several Instance of a Resource Type:-

Available:- A vector of length m indicates the number of available resources of each type.

Detection algorithm:-

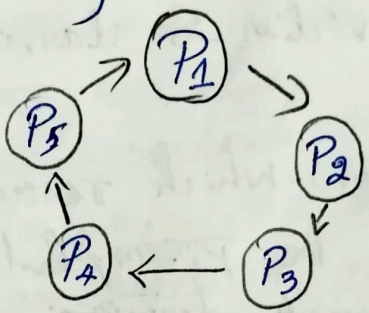
1) Let work and finish be vectors of length m and n, resp initialize.
   (a) Work = Available.
   (b) for i = 1, 2, ..., n if Allocation i ≠ 0.
2) Find an index i such that both:-
   (a) Finish [i] == false
   (b) Request i ≤ work.
   If no such i exist, go to step 4
3) Work = work + Allocation i;
   finish [i] = true
   go to step 3.
4) If finish[i] == false, for some i, 1 ≤ i ≤ n.

Eg:- If 5 process P_0 through P_4; 3 resource type A (7 instance) B (a instance), C (6 instances).

-> If a process requests resources, check if resources are available.
-> If available, then allocate
-> If not available, check if allocated to some other process that is waiting.

Circular Wait :-

-> Impose a total ordering of all resources types
-> Require the each process requests resources is an increasing order of enumeration.



Deadlock Avoidance :-

Deadlock is a state in which a process is waiting for the resources that is alread used by another process and that another process is writing for another resources.

Deadlock Recovery :-

* Process Termination
* Resource preemption

Process Termination :-

* About all deadlock process.

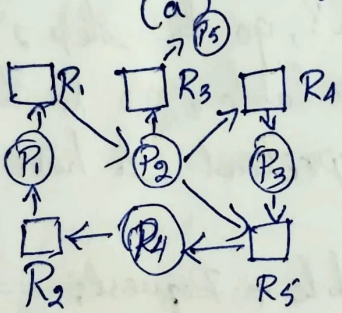
Single Instance of each Resource type:

If all resources have only a single Instance, then we can define a deadlock-detection algorithm, called wait-for graph.

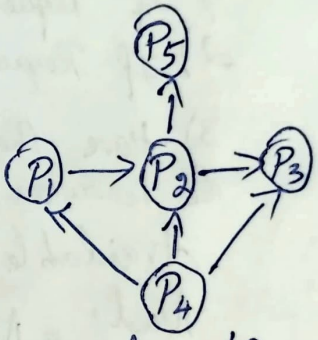
Edge from P_i to P_j in a wait-for graph implies that process P_i is waiting for process P_j to release a resource that P_i needs. An edge $P_i \rightarrow P_j$ exists in a wait-for graph if and only if the corresponding resource allocation graph.

Resource allocation graph: Corresponding wait for gra

(a)



(b)



The System then only grants the request that will lead to Safe states.

Safe state

Resource-allocation graph algorithm.

Banker's algorithm

- Safety algorithm
- Resource
- Example.

* About one process at a time until the deadlock cycle is eliminated.
* priority of the process.
* How long process has computed, and how much longer.
* Resources process needs to complete.
* How many process will need to be terminated.

Resource preemption :- If preemption is required to deal with dead locks, then three issues used to be addressed.

→ Selecting a victim → starvation
→ Roll back.

Selecting a victim :- Which resources and which Process are to be preempted? As in process termination, we must determine the order of preemption to minimize cost.

Cost factor :-

1) Number of resources a dead locked process is holding.
2) Amt of time the process consumed during its execution.

Roll back :- We must rollback the process to some safe state and restart it from that state.

Roll back - About the process is restart it.