Unit - III

Deadlocks : System model, deadlock characteri
Method for handling deadlocks
Deadlock prevention.
Deadlock avoidance.
Deadlock detection, Recovery form deadlocks

Deadlocks :- two or more process uses the same resaurce.

* A process in os uses the resources for request resource, use the resoure, release the resource.

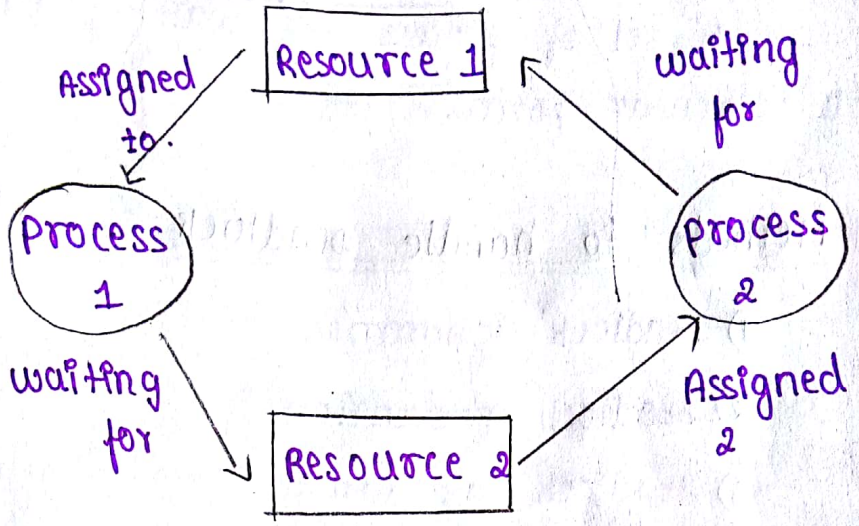
* A Deadlock is a situation where a set of process are blocked because each process is holding a resource and waiting for another resources acquried by someother process.

* A situation occurs whe in os when th are two or more process that hold a some other resource and wait for resource held by other.

For example :-

Process 1 is holding resources 1 and waiting for Resource 2 which is acquired by the process 2 and process 2 is waiting for Resource 1.

Scanned by CamScanner



Deadlocks :-

Deadlock can arrays If the following conditions occurs.

* Mutual Exclusion (Non-sharable)
* Hold & wait
* No preemption
* circular wait.

Mutual Exclusion :-

Mutual Exclusion two or more resource are non-sharable (only one process can use at a time).

Hold & wait :-

A process is holding atleast one resoure and waiting for resource.

No preemption :-

A resource can not be taken from a process unless the process release the resource.

Scanned by CamScanner

circular wait :-

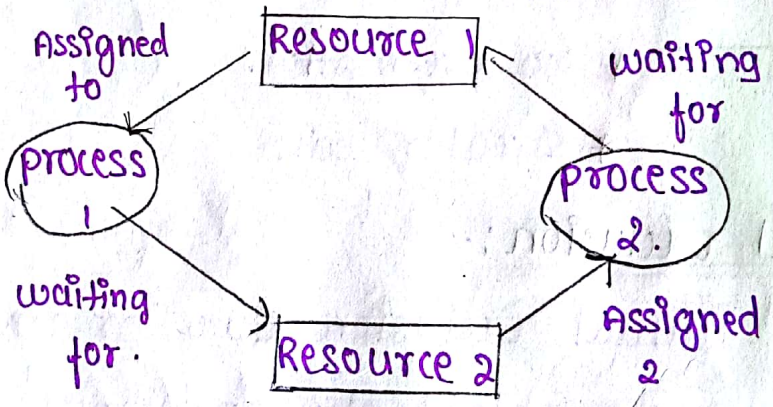
A set of process are waiting each other in a circular form.

Methods to handle deadlock.

1) Deadlock ignorance
2) Deadlock prevention (allocating time)
3) Deadlock avoidance (allocating resource)
4) Deadlock detection & recovery.

Deadlock ignorance :- (Just ignore).

Deadlock is when two or more process ask the same resource



for example :-

* when the printer (Resource) printer tries to print the page 1 (process1) and page 2 (process 2) at the same time os and then the os will hang.

* Try to ignore the deadlock simply by restarting.

* on restarting the computer or system

Scanned by CamScanner

then the problem will rectify. This method is called as deadlock ignorance.

* The os consist of a soure code or resource code as default. so when we add deadlock code to the souree code in the os then the burden will increase. which result in the low performance of the system.

* on ignoring deadlock the system capability will increase the speed will get increase and also the system performance will also increase.

Deadlock avprevention :- (வருமுன்)

Try to find solution before the deadlock occurs

Necessary conditions of deadlocks.

i) Mutual exclusion
ii) No preemption
iii) Hold & wait
iv) circular wait.

The above four conditions leads to deadlocks.

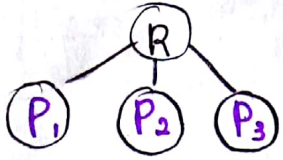
Deadlock prevention says try to remove (or) make false all the four conditions or atleast try to remove (or) make false any one of the conditi then deadlock is prevented.

Make mutual exclusion false.

* To make the mutual exclusion false just sh the resources.

Scanned by CamScanner

But



* But, printer is not sharable.

* Some cases we can make mutual exclusi false, in some cases we can not make it false

NO preemption :-

* we have to make the no preemption (false to preemption is true.

* By using the time quantum method

P_1 t_1

P_2 t_2

* TO access resources we are allocating time to process.

* whenever time is over it has to release the resources.

For example :-

* If the process 1 is allocated with the time t1 it must release the resource when the time is over. once it release then the process P_2

Hold & wait :-

* Try to do no hold and wait then the deadlock can be prevented.

* Try to give all resources to the process before its starts.

Scanned by CamScanner

circular wait :-

* Try to make circular wait false to prevent Deadlock

* To remove circular wait just give the numbering to all resources. like cpu, cpu1, printer 2, scanner 3 so that the process can request resources in the increasing order.

Deadlock avoidance.

The simplest and most usefull model requires that each process declares a maximum number of resource that it may need.

Deadlock avoidance algorithm dynamically examines the resource allocation can never be a circular way condition.

Basic facts :-

If a system is in safe state then there is no deadlock.

If a system is in unsafe state then there is a possibility of deadlock.

Avoidance :-

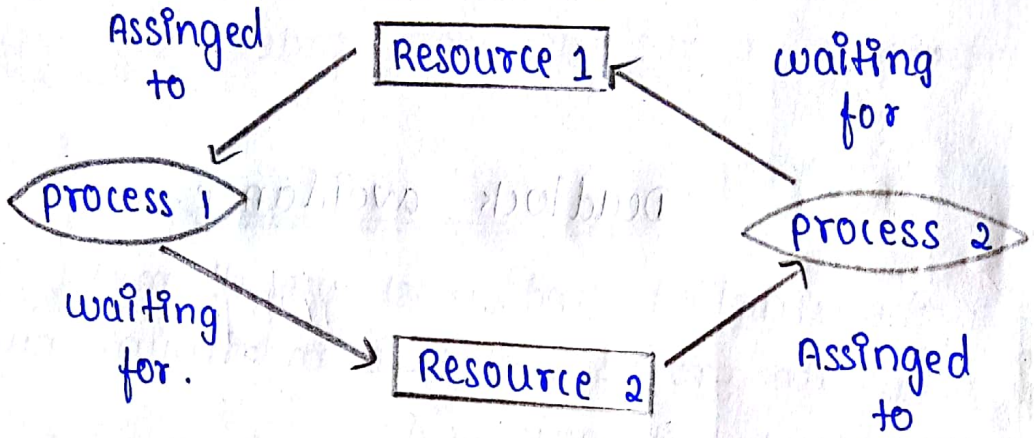
Ensure that a system will never enter a unsafe state

Deadlock avoidance can be explain using Banker's Algorithm.

Scanned by CamScanner

Deadlock system model.

A Deadlock occurs when a set of process is stalled because each process is holding a resource and waiting for another process to acquire another resource.



Deadlock in os

System model :

For the purpose of deadlock discussion a system can be modeled as a collection of limited resources that can be divided into different categories and allocated to a variety of process, each with different requirements.

Memory, printers, cpu's, open files, Tape drive, CD-ROM's and other resources are the examples of resource categories.

By definition all resources within a category are equivalents and any of the resource within that category can equaly satisfy a request from that category.

Some categories may only have one resource.

Scanned by CamScanner

The kernal keeps track of which resources are free and which are allocated. to which process they are allocated and a queue of process waiting for this resource to become available, for all kernal managed resources. Mutexes or wait() & signal() calls can be used to control application manage resource.

when every process in a set is waiting for a resource that is currently assigned to another process in the set, the set is said to be deadlocks.

Operations :-

1) Request :

If the request cannot be granted immediately then the process must wait until the resources required to become available.

2) use :

The process makes use of the resource such as printing to a printer or reading from a file.

3) Release :

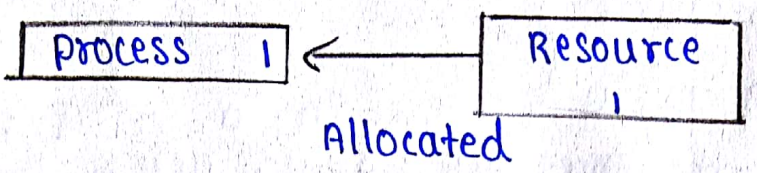
The process Releases the Resource allowing it to be used other processes.

Scanned by CamScanner

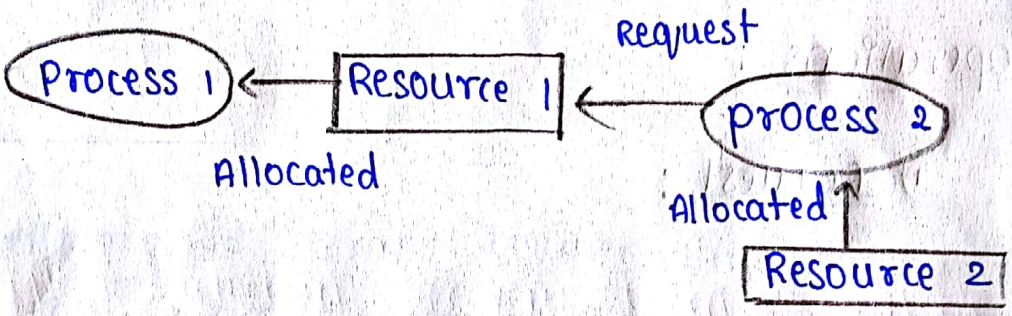
Deadlock characterization

[write about mutual exclusion, Hold and wait no preemption, circular wait]

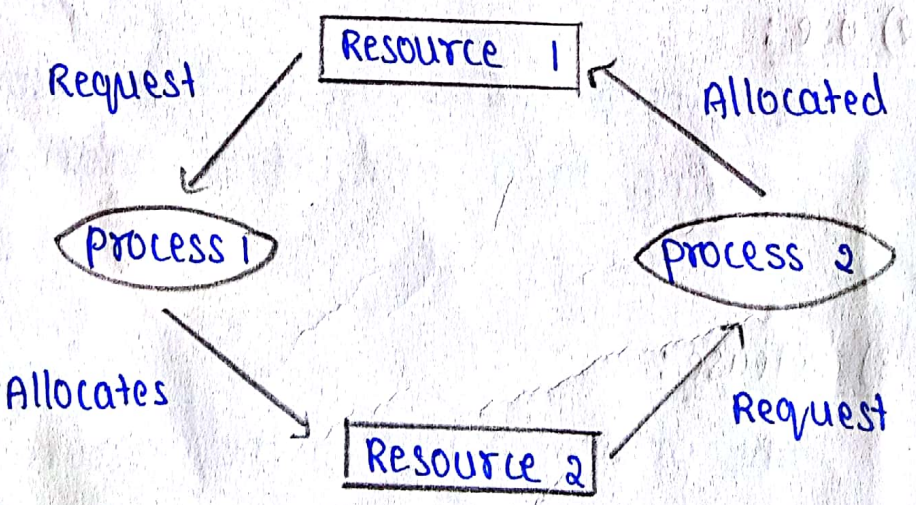
Mutual exclusion :-



No preemption :-



circular wait :-



Scanned by CamScanner

Banker's Algorithm.

process	Allocation			maximum			current work			Remaining need (Max - allocation)		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2	7	4	3
P_1	2	0	0	3	2	2	5	3	2	1	2	2
P_2	3	0	2	9	0	2	7	4	3	6	0	0
P_3	2	1	1	4	2	2	7	4	5	2	1	1
							7	5	5			
P_4	0	0	2	5	3	3	10	5	7	5	3	1

1) Remaining need $\leq$ current work

2) work = current work + Allocation.

P_0 = 7 4 3 $\leq$ 3 3 2

P_1 = 1 2 2 $\leq$ 3 3 2

work = 3 3 2 + 2 0 0

= 5 3 2

P_2 = 6 0 0 $\leq$ 5 3 2

P_3 = 2 1 1 $\leq$ 5 3 2

work = 5 3 2 + 2 1 1

= 7 4 3

P_4 = 5 3 1 $\leq$ 7 4 3

work = 7 4 3 + 0 0 2

Scanned by CamScanner

$= 7\,4\,5$

$P_0 = 7\,4\,3 \leq 7\,4\,5$

work $= 7\,4\,5 + 0\,1\,0$

$= 7\,5\,5$

$P_2 = 6\,0\,0 \leq 7\,5\,5$

work $= 7\,5\,5 + 3\,0\,2$

$= 10\,5\,7$

$\langle P_1, P_3, P_4, P_0, P_2 \rangle$ is a safe state.

If the process executes in the sequence deadlock will not occurs.

Deadlock detection Recovery from Deadlock :-

In the system does not employe deadlock prevention or deadlock avoidance then the deadlock may occur in the system.

In that case that environment (i.e) the system has to ensure two things.

1) To examine the state of the system to determine whether a deadlock has occur.

2) An Algorithm to recover from the deadlock.

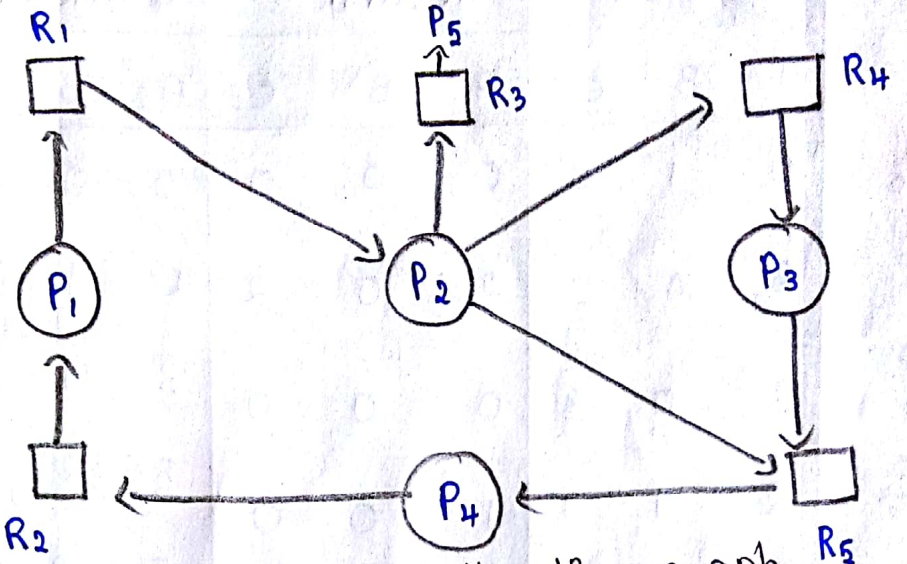
To dectect the deadlock there are two ways.

1) Single resource of each resource type.

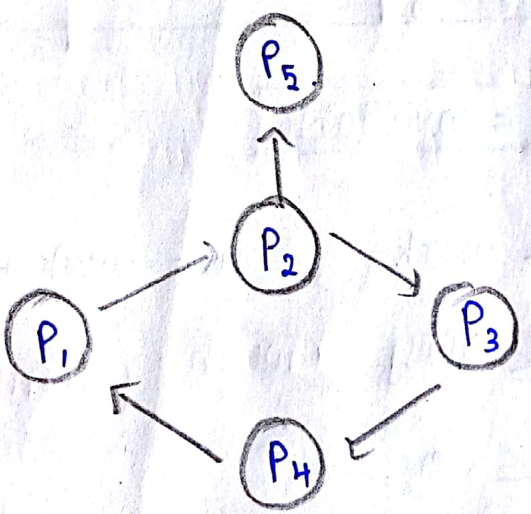
Scanned by CamScanner

2) several instance of a resource type.

single instance of each resource type :



Resource allocation graph



corresponding wait for graph.

If it forms a cycle then the deadlock will occur.

If it does'nt form a cycle then the deadlock never occur.

Scanned by CamScanner

Multiple Instance of Resource type :-

A = 7
B = 2
C = 6

Process	Allocated			Request			available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0	0	0	0	0	0
P_1	2	0	0	2	0	2	0	1	0
P_2	3	0	3	0	0	0	3	1	3
P_3	2	1	1	1	0	0	5	2	4
P_4	0	0	2	0	0	2	5	2	6
	7	2	6				7	2	6.

Work = available

Request ≤ work , W = work + allocation.

P_0 = Req ≤ work ✓

= 000 ≤ 000

W P_0 = 000 + 010

= 010

P_1 = Req ≤ work ✗

= 202 ≤ 010

P_2 = Req ≤ work ✓

= 000 ≤ 010

W = 010 + 303

Scanned by CamScanner

= 313

P_3 = Req ≤ work ✓

= 100 ≤ 313

w = 313 + 211

= 524

P_4 = Req ≤ work ✓

= 002 ≤ 524

w = 524 + 002

= 526

P_1 = Req ≤ work ✓

= 202 ≤ 526

w = 526 + 200

= 726.

∠ P_0, P_2, P_3, P_4, P_1,

If the current work is equal to the (i.e) match with the available values then there is no deadlock.

A = 7

B = 2

C = 6

which is equal to the current value then there is no deadlock.

Safe sequence ∠ P_0, P_2, P_3, P_4, P_1.

Scanned by CamScanner

Deadlock recovery :-

Deadlock Recovery performs when a deadlock is detected.

when deadlock detected, then our system stops working and after the recovery of the deadlock our system start working again. Therefore after the detection of deadlock a method must require to recover that deadlock to run the system again. this method or way is called as a deadlock recovery.

various ways of deadlock recovery.

* Deadlock recovery through ~~prepn~~ preemption
* Deadlock recovery through roll pack
* Deadlock recovery through killing processes.

Deadlock recovery through preemption :-

The ability to take a resource away from a process, have another process using and then give it back without the process noticing.

It is highly depended on the nature of the resource.

Deadlock reeovery through preemption is difficult are sometime itn possible.

Deadlock Recovery through Roll pack :-

In this case of deadlock recovery through roll back, whenever the deadlock is detected,

Scanned by CamScanner

It is easy to see which resources are needed.

to Do the recovery of deadlock, a process that owns a needed resources is rolled back to a point in time before it acquire some there resource just by starting one of its earlier. check points.

Deadlock recovery through killing process :-

This method of deadlock recovery through killing process is a simpliest way of deadlock recovery.

sometimes it is best to kill a process that can be return from the begining with no ill effects.

Scanned by CamScanner