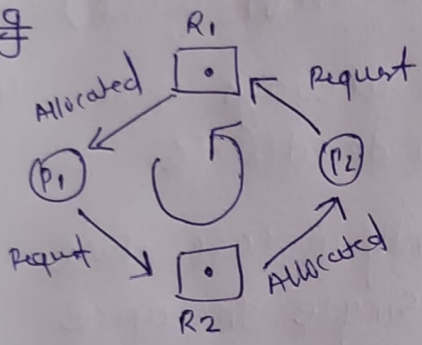
Deadlock concept :

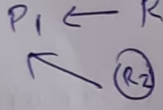
Two or more process are waiting on some event which never happened then this process are said to be involve in deadlock

Process → P1 P2

Resource → R R

e.g





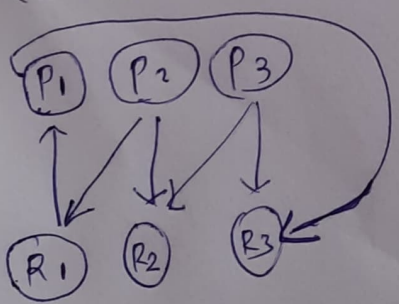
For deadlock there are four necessary conditions :-

(1) Mutual Exclusion (one at a time)

(2) NO preemption

(3) Hold & Wait (holding one res. waiting for other)

(4) Circular Wait



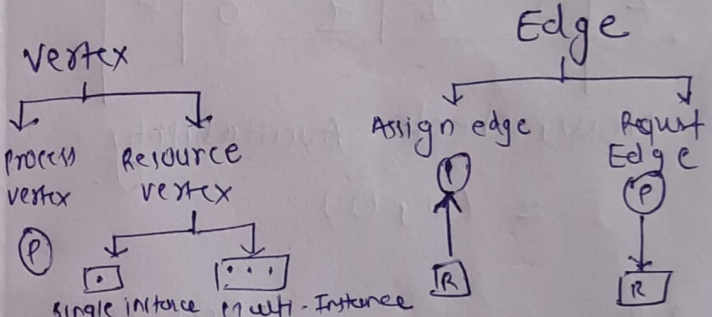
(1) Mutual exclusion :-
The resource are allocated to only to one process at a time (NO sharing)

(2) Hold & Wait :-
The process holding some resources and at same time waiting some other resources.

(3) NO - preemption
NO sharing of CPU

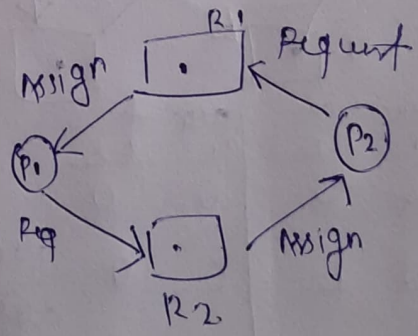
(4) Circular wait / cycle

Topics
- Resource allocation Graph (RAG)



RAG is the most effient and convenient way to represent the state of the System.

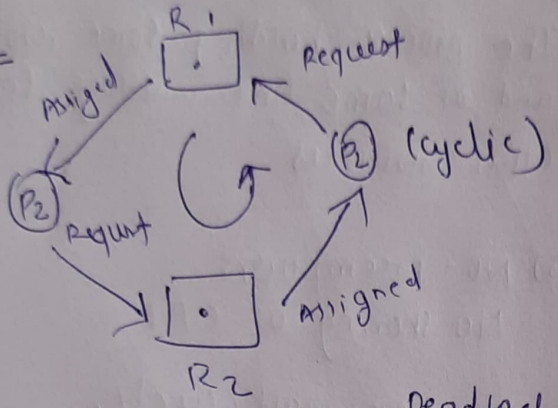
e.g



Why we represent in the form?

We have to check whether deadlock occurs or not

e.g



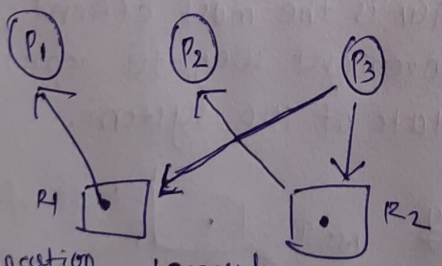
Circular wait — Deadlock situation

	Allocation R1	Allocation R2	Request R1	Request R2	
P_1	1	0	0	1	x
P_2	0	1	1	0	x

Now, check Availability

$= (0, 0)$ (R_1 R_2)

e.g (Acyclic)



	Allocation R1	Allocation R2	Request R1	Request R2	
x P_1	1	0	0	0	
x P_2	0	1	0	0	Deadlock doesn't occurs
x P_3	0	0	1	1	

Availability (0, 0)

(1, 0)

(1, 1)

A process is waiting for a resource

↳ if waiting is finite (starvation)

↳ if the waiting is infinite (deadlock)

If there is a circular wait in RAG (cycle)

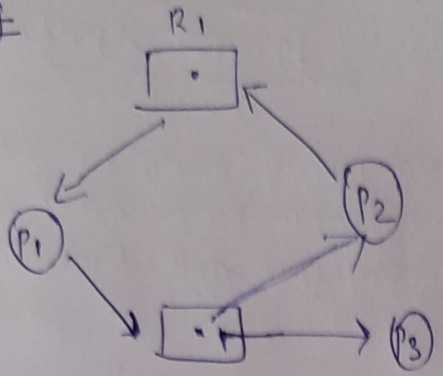
↳ than always there will be a deadlock

↳ but only in case there is a single Instance Resource

- If RAG has no cycle than there is no deadlock.

Multi-Instance RAG

e.g



	Allocation		Request		
	R_1	R_2	R_1	R_2	
P_1	1	0	0	1	x
P_2	0	1	1	0	x
P_3	0	1	0	0	x

Availability

(0,0)

(0,1)

0 1

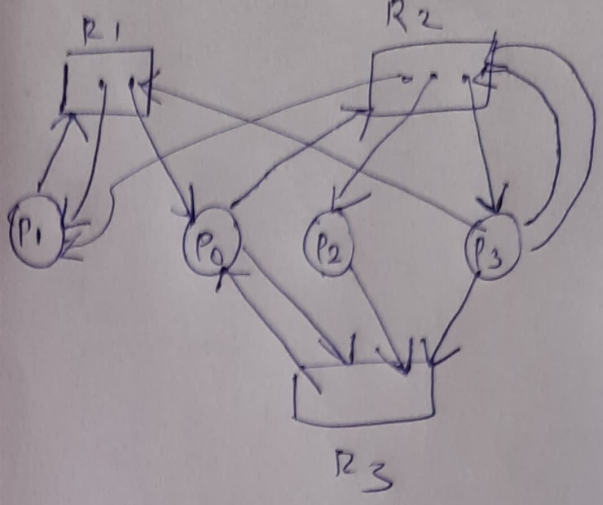
0 1

1 1

∴ there is no deadlock

GATE

e.g



	Resource Allocation			Request			
	R_1	R_2	R_3	R_1	R_2	R_3	
P_0	1	0	1	0	1	1	x
P_1	1	1	0	1	0	0	x
P_2	0	1	0	0	0	1	x
P_3	0	1	0	1	2	0	x

Current Availability

(0,0,1)

0 1 0

0 1 1

1 0 1

1 1 2

2,3,2

∴ NO deadlock occurs

The current Availability is 2,3,2

Various Methods to Prevent / handle deadlocks

~~* preventing 4~~

**

(1) Deadlock ignorance (Ostrich Method)

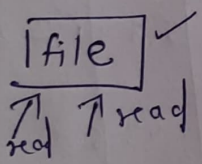
↳ We ignore because we don't want to Affect performance (speed)

(2) Deadlock prevention

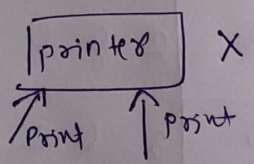
→ preventing 4 necessary conditions will prevent deadlock

→ At least try to discard one of the condition to prevent deadlock

(1) Mutual exclusion
(2) Hold & Wait
(3) NO preemption
(4) Circular Wait



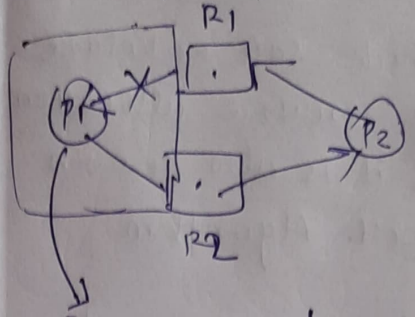
We can open one file in multiple read mode.



We can't print 2 files at same time

∴ mutual exculsion can't be prevented everytime

(2) NO preemption

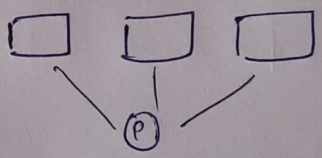


P1 get premated

NOW P1 is free
P2 can be Successfully executed

How we can do this?
By using time quatum or time stamp.

(3) Hold & Wait

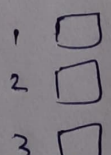


(1) Allocate the resource req. by process before start of execution

(2) The process should release all existing resources before making new requst

(4) Circular Wait

Just give the numbering order all the resources



Increasing order

(3) Deadlock Avoidance

We will check safe or unsafe for every resource allocation we check it is also known as Banker's Algorithm.

(4) Deadlock detection and Recovery

- first it we detect and than recover
- recovery simply
  - kill the processes or process
  - Resource preemption

"Bankers' Algo"
Also known as deadlock Avoidance Algorithm.
Deadlock detection Algo

Steps

[1] find out remaining need.

Remaining need = max need - current allocation

[2] you have to check current available and current need of each process

e.g

Given

Total A = 10, B = 5, C = 7

Process	Allocation A	B	C	Max Need A	B	C	Avg A	B	C	Rem Need A	B	C
P_1	0	1	0	7	5	3	3	3	2	7	4	3
P_2	2	0	0	3	2	2	5	3	2	1	2	2
P_3	3	0	2	9	0	2	7	4	3	6	0	0
P_4	2	1	1	4	2	2	7	5	3	2	1	1
P_5	0	0	2	5	3	3	10	5	5	5	3	1
	7	2	5				10	5	7			

Safe / Unsafe