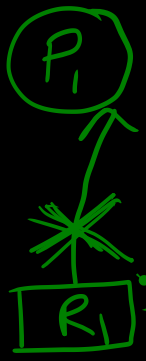
Deadlock Prevention: ✗

- → ~~multi~~ mutual exclusion ✗ ✗
- → Hold and wait ✗ ✗
- → no preemption ✗ ✗
- → Circular wait ✗ ✓

} necessary conditions.

if any one of these conditions fail, then DL is not possible



By forcefully pulling away a resource. 'CPU' was resource.
→ not possible because work will be lost

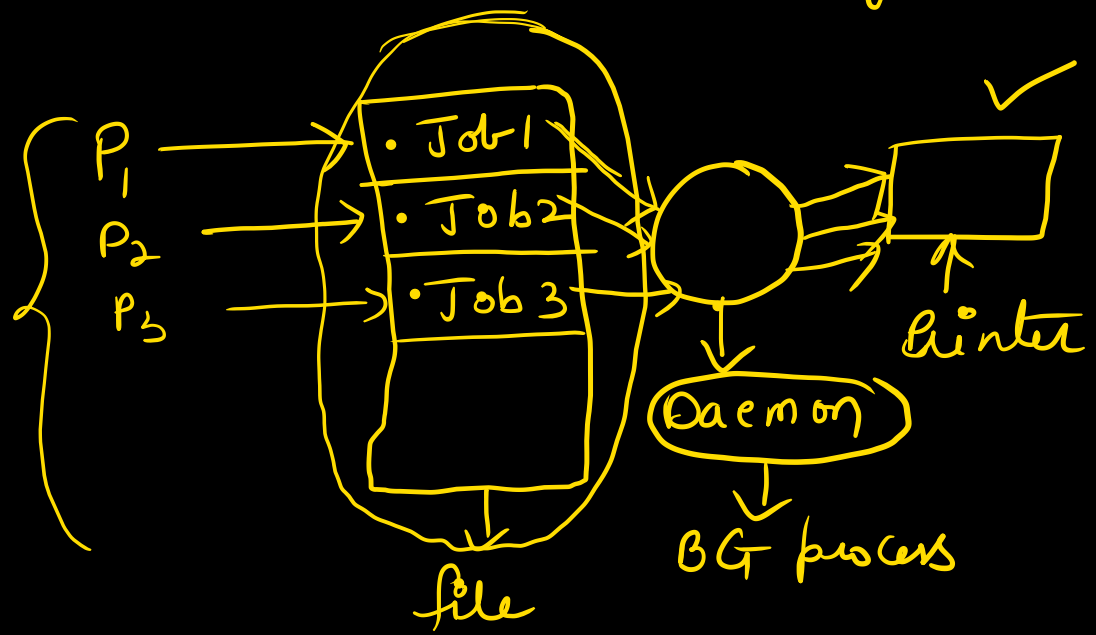
mutual exclusion: only one process can use a resource at a time.

fail → multiple processes can access a resource at the same time.

It is not possible. Ex: CPU, register, ...

Ex: Printer → multiple processes cant access at the same time.

But we can use spooling.



MEX → NO

Hold and wait:

!(Hold and wait) = (!Hold) or (!wait).

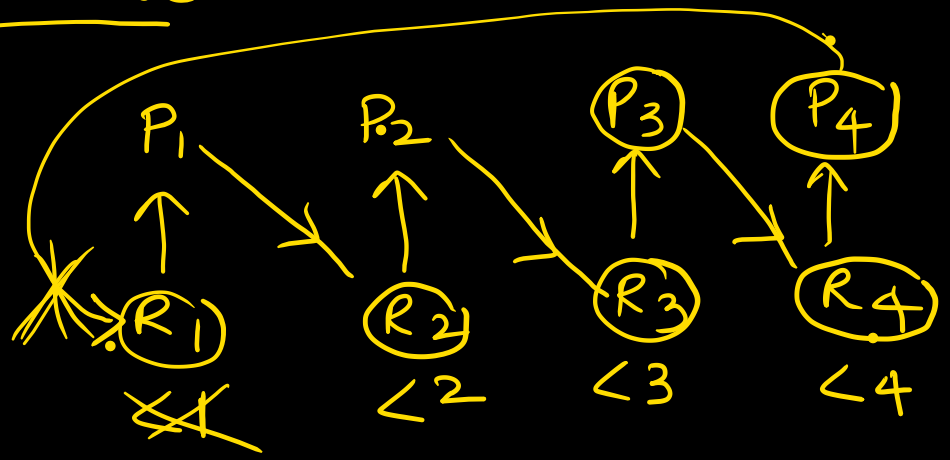
!Hold: Before requesting a new resource, release all previously held resources.

- may not be possible because previous resources may be still needed

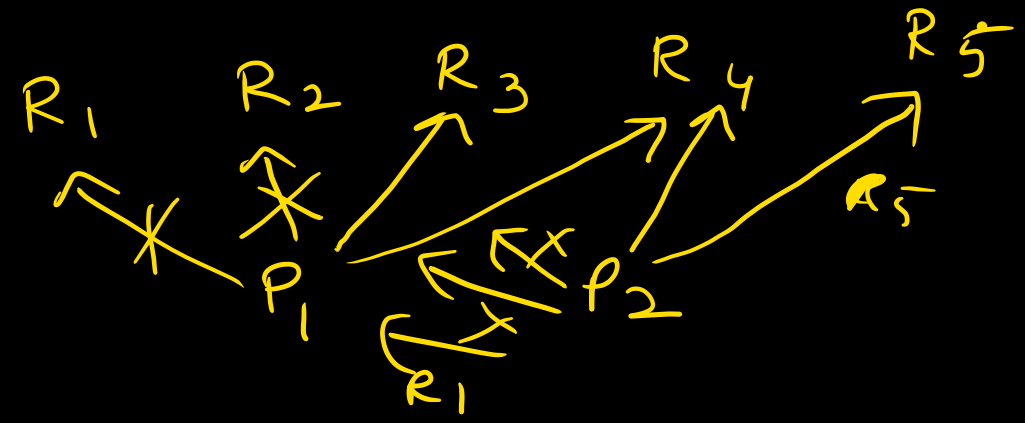
!wait: Request and obtain all resources before we start

- not possible because we cant predict what resources we need.

Circular wait:



Circular wait.

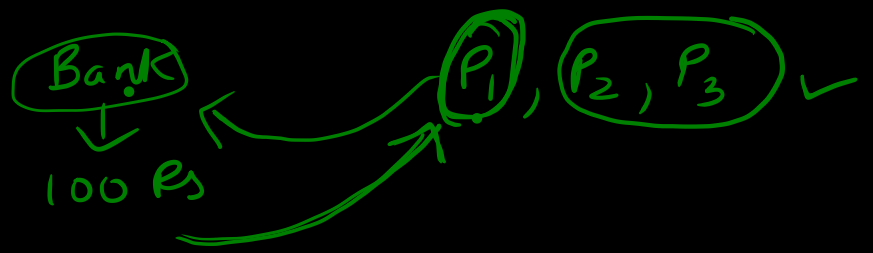


Order all resources numerically and processes can ask for resources always in ascending order.

If every process ask for resources in ascending order then no deadlock will occur.

This is the only feasible solution for dead lock prevention.

Deadlock handling:



→ .OL ignorance ✓

→ .OL prevention ✓

→ OL avoidance ⟹ Gate questions

→ ✓ OL detection and recovery.

OL avoidance:

Bankers algo ✓ ∂ Safety algo ∂ Safe and unsafe OL avoidance algo.

Process	R_1	R_2	R_3	R_4
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0
	5	3	2	2

Resources assigned

	R_1	R_2	R_3	R_4
✓A	1	1	0	0
✓B	0	1	1	2
✓C	3	1	0	0
✓D	0	0	1	0
✓E	2	1	1	0

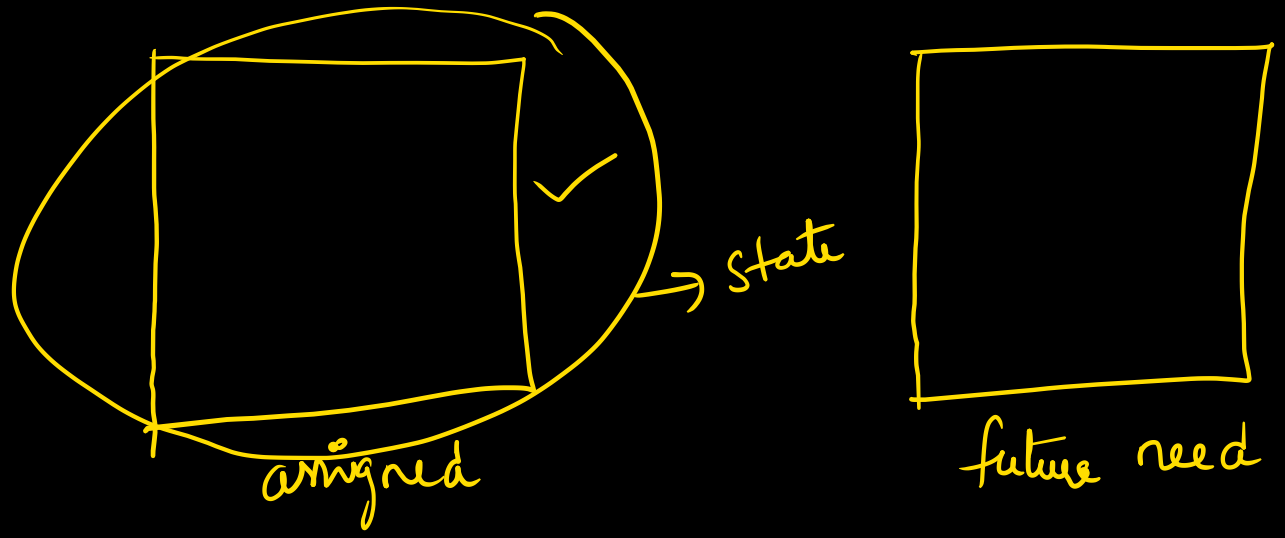
Resources still needed

OS got req from E = (0010)
will OS grant it?

$E = (6\ 3\ 4\ 2)$ → Total resources available
$P = (5\ 3\ 2\ 2)$ → resources assigned
$A = (1\ 0\ 2\ 0)$ → resources available

Safe.

Ans: Granted



Bankers algo: will grant a request only if it safe.

all processes can Complete? in ~~any~~ some order.

Safe State: A state is said to be safe if there exists a sequence in which you could satisfy the need of all processes in some order with out getting into a.

Bankers algo: will always go from one safe state to another safe state.

	X	Y	Z
P_0	1	2	1
P_1	2	0	1
P_2	2	2	1
	5	4	3

alloc

	X	Y	Z
P_0	1	0	3
P_1	0	1	2
P_2	1	2	0

request

P_1	0	1	2
	2	0	1
	2	1	3
P_0	1	2	1
	3	3	4
P_2	2	2	1
	5	5	5

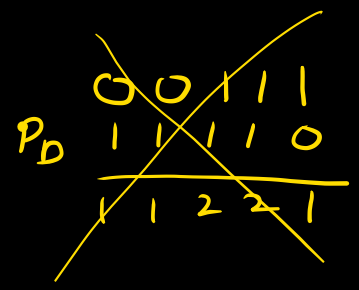
There are 5 units of each resource type. Safe or not?

Safe

	X	Y	Z
Total	5	5	5
alloc	5	4	3
avail	0	1	2

Gate 16)

	Allocated					Maximum					Future need				
P_A	1	0	2	1	1	1	1	2	1	3	0	1	0	0	2
P_B	2	0	1	1	0	2	2	2	1	0	0	2	1	0	0
P_C	1	1	0	1	1	2	1	3	1	1	1	0	3	0	0
P_D	1	1	1	1	0	1	1	2	2	0	0	0	1	1	0



available = 0 0 X 1 1 . Q. what is the smallest value of 'X' for which this is safe state?

X=1

X=2

0 0 2 1 1

P_D: 1 1 1 1 0

1 1 3 2 1

P_C: 1 1 0 1 1

2 2 3 3 2

P_A: 1 0 2 1 1

3 2 5 4 3

P_B:

Gate-14 - Set 1

	Alloc			max			future need		
	X	Y	Z	X	Y	Z	X	Y	Z
P_0	0	0	1	8	4	3	8	4	2
P_1	3	2	0	6	2	0	3	0	0
P_2	2	1	1	3	3	3	1	2	2

AV: 3 2 2

Req 1: P_0 (0 0 2) ✓

Req 2: P_1 (2 0 0)

Which one is granted.

safe to grant req 2.

	1	2	2
P_1:	5	2	0
	6	4	2
P_2:	2	1	1
	8	5	3

P_0