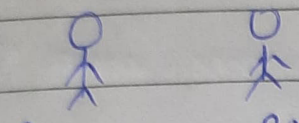
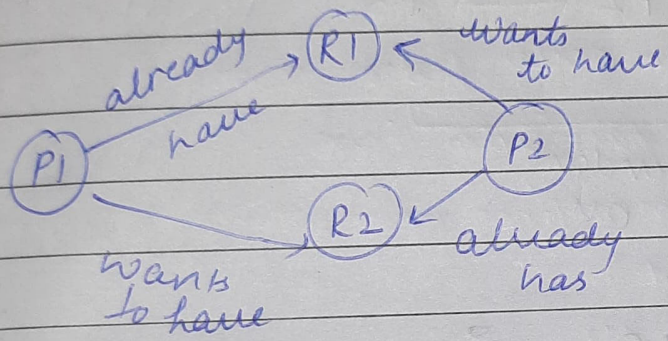
Deadlock Introduction



P1 says P2 leave resources after you!
P2 says P1 leave resources after you!



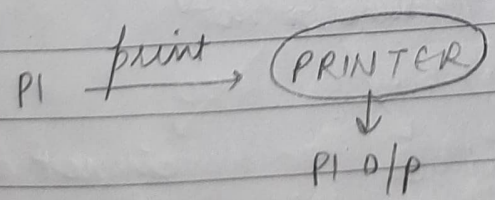
DEADLOCK!

Conditions of a Deadlock

* Mutual Exclusion
* No preemption
* Hold & Wait
* Circular wait

Mutual Exclusion

- access of Resources must be mutually exclusive
- Resources are in non-shareable mode



No preemption

- A resource can't be snatched from a process until released by the process itself.

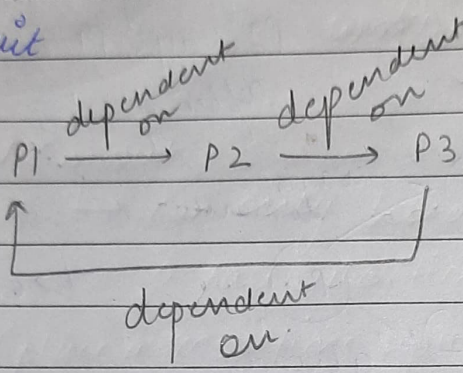
Hold and Wait

- If P1 needs 10 and has hold 3 resources

DATE: / /
PAGE:

↳ P1 should hold 3 resources while waiting for 7 other resources to come.

Circular wait



Deadlock Handling Methods

(#) **Prevention of deadlock**

- design system such that deadlock never occurs.
- Try to violate any of the four conditions

(#) **Avoidance of deadlock.**

- Whenever a process needs a resource
  ↳ check if giving the resource possible or not
- system takes decision of enter. a fresh req or not based on some data.
  - if yes → Continue → safe mode
  - if not → revert back → unsafe mode

Detection and Recovery of deadlock

– decision after deadlock occurs

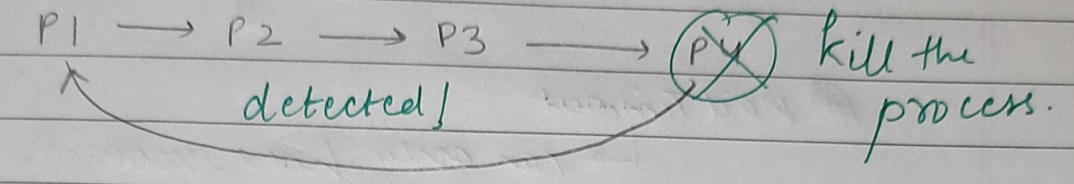
DATE: / /
PAGE:

↓

detect deadlock

↓

Recover from deadlock



Recovery → kill process (inefficient)

→ Resource pre-emption

Ignorance of deadlock

↳ Ignoring deadlock ↳ Dil mai ek bar ayega

Recovery: ← System restart

Deadlock Prevention

Condition ① Mutually exclusive

For deadlock, resources must be non-shareable and to prevent deadlock we can make resources shareable.

BUT we can't change it (property of resource)

Can't be violated

Condition (II) Hold and wait

DATE: / /
PAGE:

// conservative approach

When a process enters → provide all resources ← then only start

BUT inefficient ⇒ many process will keep waiting

Wait timeout
↳ can only hold resources for a particular time
(place a max. time upto which a process can wait)
↳ After which process must release all the resources.

Condition (III) Non preemption

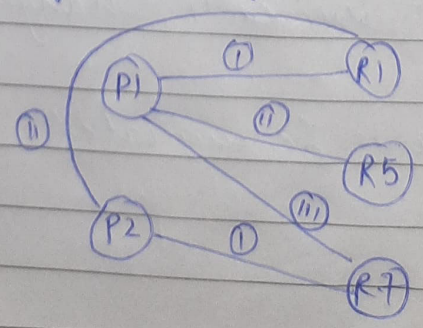
based on priority [we allow a process to forcefully preempt the resource held by other process
↳ preferably wait state process.

Condition (IV) Circular wait

P1 → P2 → P3 → P4 → P1

- Resources are numbered R1, R2, R3 ..... R50
- Process can ask for resources but in either rising or falling order.

Process	Need
P1	R1 R5 R7
P2	R7 R1



DATE: / /
PAGE:

Numerical on Bankers's Algorithm

Process	Allocated			Request		
	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

All resources X, Y and Z are available 5 units each. Which process is executed last?

First lets compute how many units of X, Y & Z are available after allocation.

Total units of X allocated = 1 + 2 + 2 = 5
Total units of Y allocated = 2 + 0 + 2 = 4
Total units of Z allocated = 1 + 1 + 1 = 3

Units of X available = 0
——— Y ——— = 1
——— Z ——— = 2

Now P0 requires 1 unit of X
0 units of Y
and 3 units of Z
⇒ Not available to us
⇓
P0 needs to wait.

On the other hand P1 req. are satisfied!
⇒ P1 will run
⇒ Release resources.

Units of X available now

⇒ P2 (X Units allocated + X units requested)

⇒ 2+0 ⇒ 2 units

Similarly

Units of Y available ⇒ 0+1 = 1

Units of Z available ⇒ 1+2 = 3

for P2 units of X needed = 1
—— Y —— = 2
—— Z —— = 0
⇒ Can't run as Y units are not comple.

for P0 units of X needed = 1
—— Y —— = 0
—— Z —— = 3
Satisfied!

P0 will run ✓

Release resources ✓

Units of X available ⇒ 1 + 1 + 1 = 3
—— Y —— ⇒ 3
—— Z —— ⇒ 4

Check for P2 ⇒ Req. satisfied!

⇒ P2 will run ✓

P2 releases resources ✓

Units of X available = 5
—— Y —— = 5
—— Z —— = 5

LAST PROCESS EXECUTED ⇒ (P2) Ans