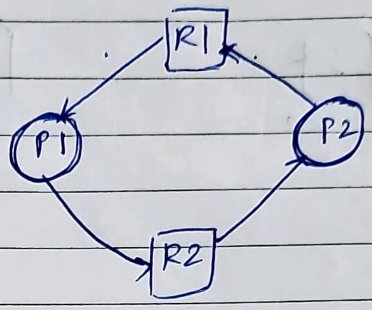
# UNIT - 3

**Deadlock.**

eg { P1 P2
Printer }
Scanner }

* Both require printer and scanner.
* P1 wants printer and later wants scanner
* P2 wants scanner first and later wants printer

* After printer, P1 goes to ~~printer~~ waiting state since scanner is not available.
* Similar to P2 also.



* Unsafe situation - when we don't know when the resource is not available.

> To use Resource:
> * Process makes req.
> * Use the resource to max limit
> * Return back the resource.

* Deadlock is a unsafe situation but All unsafe situation don't lead to deadlock.

## Necessary conditions for Deadlock

1. Mutual exclusion — no two process will use same resource at same time
2. Hold and wait — * hold ~~for~~ one resource and wait for resource.
3. Circular wait — one ~~leas~~ process has to wait for other to complete
4. No pre-emption — • wait for one process to complete execution; * forcibly you can't take a resource from a process.

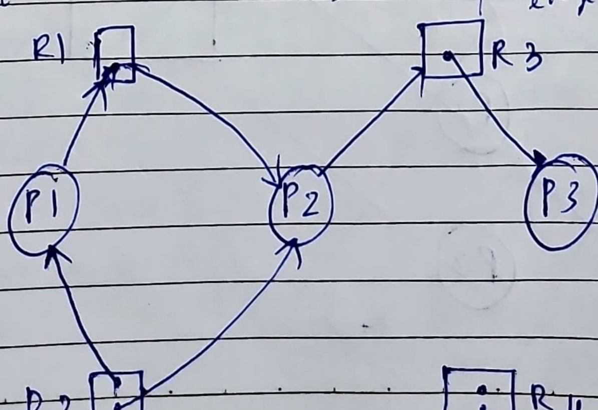
## Deadlock representation using

* resource allocation graph.

$P = \{P1, P2, ..., Pn\}$
$R = \{R1, R2, ..., Rm\}$
$E = \{Pi \rightarrow Rj, R_k \rightarrow Pm\}$.

eg: $P = \{P1, P2, P3\}$
$R = \{R1, R2, R3, R4\}$
$E = \{$ ~~R1→P1, R2→P2~~,
assignment edge: $R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$.
request edge: $P1 \rightarrow R1, P2 \rightarrow R3$

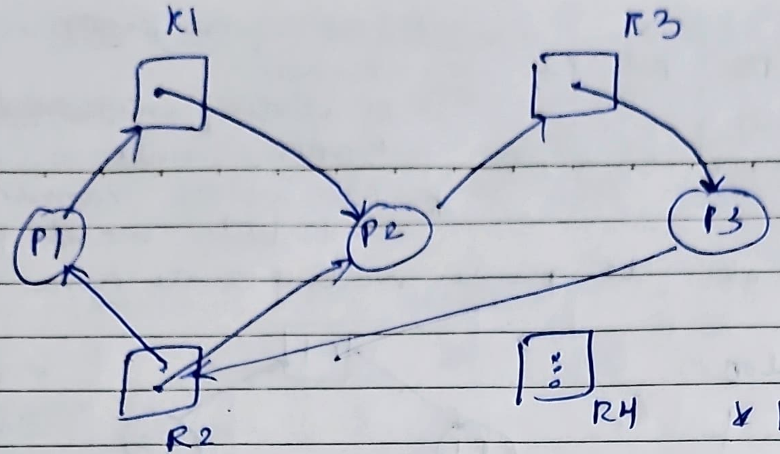
note:
$R \rightarrow P$ assignment edge.
$P \rightarrow R$ request edge



* P1 is holding R2 and waiting for R1
* P2 also.
* P3 is holding R3 but waiting for none.
* Hence it is in safe state

order for execution ⟨P3, P2, P1⟩.

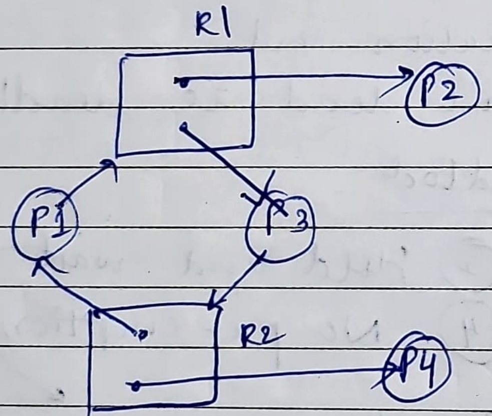
eg ②




* ~~all processes are in hold and waiting and circular wait~~

* Hence deadlock.

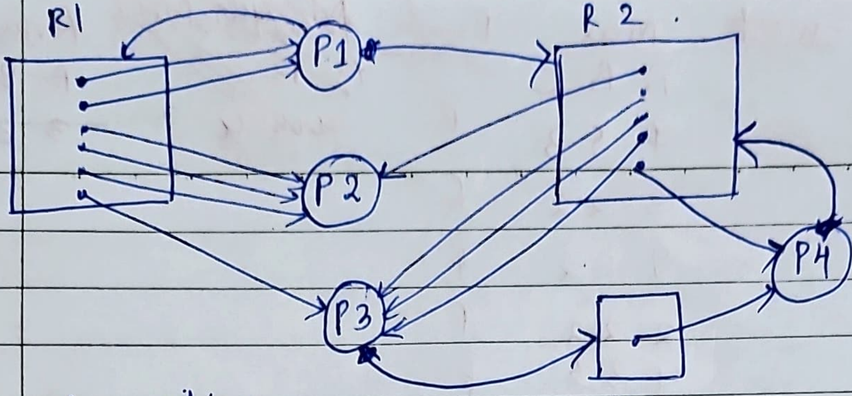
eg ③




* P2 and P4 are not waiting for any resource.

* ~~Hence safe situation~~

* Hence not deadlock.




$P2 \to P1 \to P4 \to P3$

* P2 is not waiting for any resource
* not a deadlock

Date :

## Deadlock Handling Techniques

1) Deadlock prevention
2) Deadlock Avoidance
3) Deadlock Detection and Recovery

Saftety algo (check if result is unsafe or not)

Resource allocation algo (extra resource will be given to process)

12 mark

### Safety Algo

1. Let Work and Finish be vectors of length m and n,
   Work = Available
   Finish [i] = false. for i = 0, 1, .., n-1
2. Find index i such that
   Finish[i] == false.
   $Need_i \leq Work$.
   If no such i exists, go to step 4.
3. Work = Work + Allocation
   Finish [i] = true
   go to step 2.
4. if Finish [i] == true for all i, then system is in safe state.

### Resource Req. Algo

Let $Request_i$ be request vector for process $P_i$. If $Request_i[j] = k$, Then $P_i$ wants k instance of Resource $R_j$. When request for resource is made,

1. if $Request_i \leq Need$, go to (2). otherwise raise error since process exceeded max. claim.
2. If $Request_i \leq Available$, go to (3). Otherwise $P_i$ must wait since resource not available.
3. Modify
   $Available$ -= $Request_i$
   $Allocation_i$ += $Request_i$
   $Need_i$ -= $Request_i$

| | Allocation | Max | Finish | Available |
|---|---|---|---|---|
| | A B C | A B C | | A B C |
| $P_0$ | 0 1 0 | 7 5 3 | F | 3 3 2 |
| $P_1$ | 2 0 0 | 3 2 2 | F | |
| $P_2$ | 3 0 2 | 9 0 2 | F | |
| $P_3$ | 2 1 1 | 2 2 2 | F | |
| $P_4$ | 0 0 2 | 4 3 3 | F | |

note: $\boxed{\text{need} = \text{max} - \text{allocation}}$

| | Need → (2 marks) | Work | Finish |
|---|---|---|---|
| | A B C | A B C | |
| $P_0$ | 7 4 3 | 3 3 2 | F |
| $P_1$ | 1 2 2 | | F |
| $P_2$ | 6 0 0 | | F |
| $P_3$ | 0 1 1 | | F |
| $P_4$ | 4 3 1 | | F |

At time $t_0$, work = 3 3 2.

Here: ~~Cl~~ Need ($P_1$) ≤ work and Finish[1] = 1

1 2 2 ≤ 3 3 2.

∴ Work = work + allocation ($P_1$)

= 3 3 2 + 2 0 0

= 5 3 2

Finish [1] = T

At time $t_1$, work = 5 3 2.

Need ($P_3$) ≤ Work.

∴ Work = work + allocation ($P_3$)

= 5 3 2 + 2 1 1

= 7 4 3.

Finish [3] = T.

At time $t_2$, work = 7 4 3

Need ($P_4$) ≤ Work.

∴ Work = 7 4 3 + 0 0 2.

= 7 4 5.

At time $t_3$, work = 7 4 5 (Since $P_2$ has less resource need).

Need ($P_2$) ≤ Work

∴ Work = 7 4 5 + ~~3 0 2~~ (3 0 2)

= ~~7 5 5~~ ~~10 4 7~~ ~~7 5 5~~ (10 4 7)

At time $t_4$, work = ~~7 5 5~~ ~~10 4 7~~ ~~7 5 5~~ (10 4 7)

Need ($P_2$) ≤ Work

∴ Work = 10 4 7 + ~~3 0 2~~ 0 1 0.

= ~~10 5~~ (10 5 7)

Since Finish[i] = true for all processes; the system is in safe state.

Safe sequence ~~order~~: P1, P3, P4, P2, P0.

Q) If P1 requests for $\langle 1\ 0\ 2 \rangle$ then is it considered or rejected?

1. P1 request = (1 0 2)
   P1 Need = (1 2 2).
   ∴ Request (P1) ≤ Need (P1)

2. Available (P1) = 3 3 2.
   Request (P1) ≤ Available

3. Available = Available − Req.

$$\begin{array}{r} 3\ 3\ 2 \\ -\ 1\ 0\ 2 \\ \hline 2\ 3\ 0 \end{array}$$

Available = 2 3 0

Alloc = Alloc + Req

$$\begin{array}{r} 2\ 0\ 0 \\ +\ 1\ 0\ 2 \\ \hline 3\ 0\ 2 \end{array}$$

Allocation(P1) = 3 0 2

Need(P1) = Need − Request

$$\begin{array}{r} 1\ 2\ 2 \\ -\ 1\ 0\ 2 \\ \hline 0\ 2\ 0 \end{array}$$

Need(P1) 0 2 0.

Now perform safety algorithm to check if it is safe by changing above values.

$\langle$ P1, P3, P4