Deadlocks

4 Conditions of a deadlock

1. Mutual Exclusion.
2. Hold and wait
3. NO preemption
4. Circular Wait

How to deal with deadlocks?

1. Prevention.
2. Avoidance.
3. Detection and resolution

---

1. **Prevention**:

Prevent 1 of the 4 conditions.

1. Mutual Exclusion. → Cannot be prevented.

2. **Hold and Wait**

Request all the resources at once

disadv

Might wait a long time to get all the resources (starvation).

Low resource utilization

---

3 No preemption

A process releases resources it holds if it is denied a request.

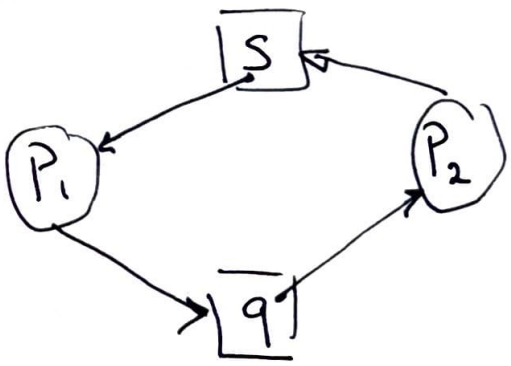
---

4 Circular Wait

Define a linear order for resource and require processes to request resources in that order

⇒ No Circular Wait

P_1	P_2	
wait(s)	wait(q)	wait(s)
wait(q)	wait(s)	wait(q)
:		
Signal(q)	Signal(s)	
Signal(s)	Signal(q)	



2 Avoidance

Processes must declare their maximum need of resources of each type.

State

1. Current allocation of resources.
2. Available resources.

Safe: The system can allocate resources to each process (up to its maximum need) in some order (Safe Sequence) and still avoids a deadlock.

Safe Sequence: P_1 P_2 P_3 P_5 P_4

P_1 can Finish by what is available.

P_2 can finish →

Unsafe: no safe sequence exists.

May lead to a deadlock.

Ex

12 tapes.

Unsafe (Deadlock)

Safe

	Current	Maximum.	Need
P_0	5	10	5
P_1	2	~~4~~ 6	2
P_2	2	9	7
	9		3 available.

=

$P_1 \rightarrow P_0 \rightarrow P_2$ Safe

P_1 → avaible =5

P_0 → avaible 10

P_2 requests one more tape

	Curren	max	Need
P_0	5	10	5
P_1	2	4	2
P_2	3	9	6
	10		avaible = 2

pretending P_1 → availabk 4

Unsafe

Avoidance ~ remaining in a safe state

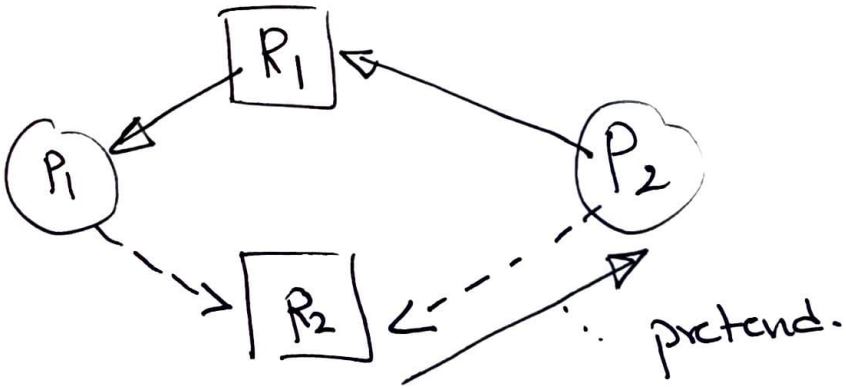
Grant requests only if they result in a safe state.

Resource Allocation graphs

Works only with single instances of resources.

Claim edge : request.

$P_i \dashrightarrow R_j$ process may request R_j



P_2 requests R_2 : will create a cycle

Should be denied.