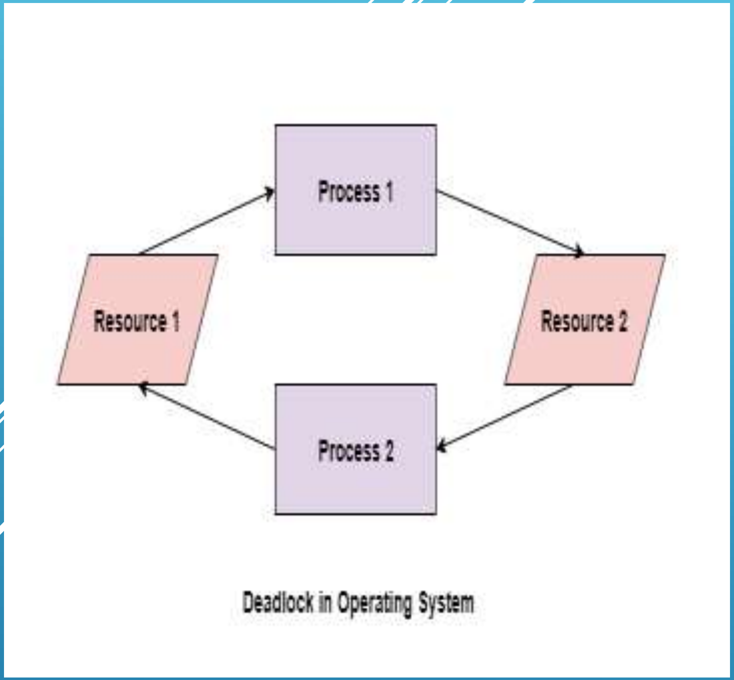
DEADLOCK

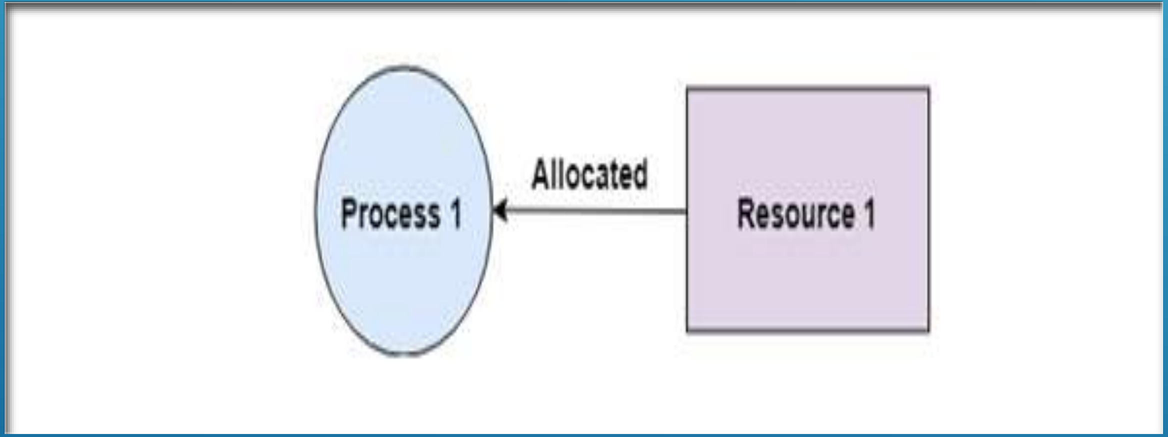
A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.



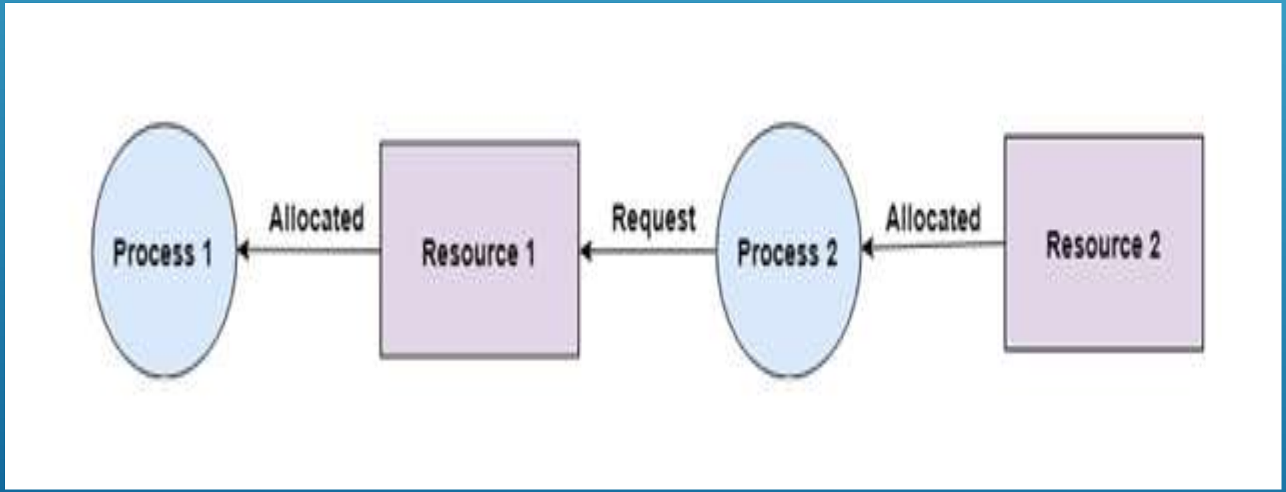
In the above diagram, the process 1 has resource 1 and needs to acquire resource 2. Similarly process 2 has resource 2 and needs to acquire resource 1. Process 1 and process 2 are in deadlock as each of them needs the other's resource to complete their execution but neither of them is willing to relinquish their resources.

CONDITIONS FOR DEADLOCK

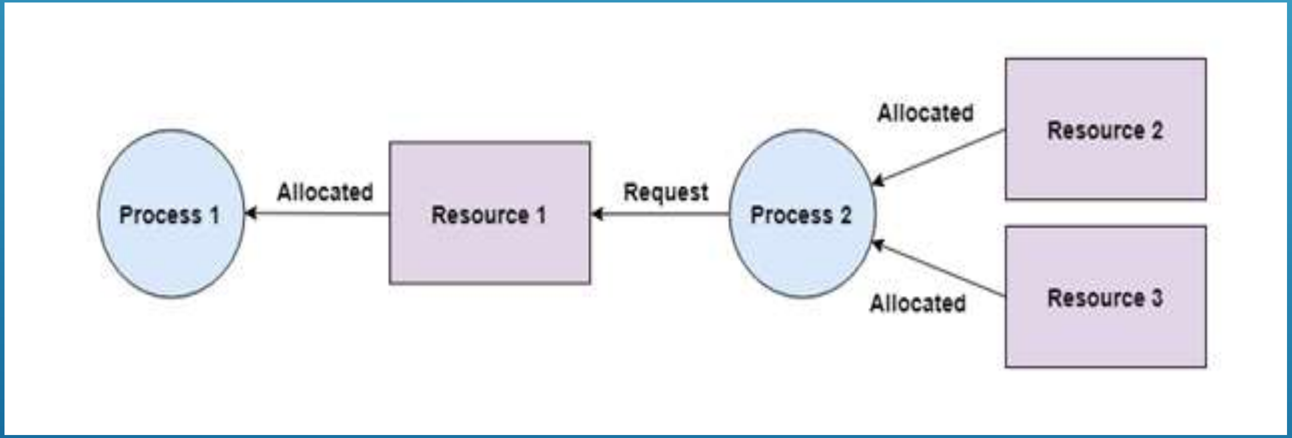
- **Mutual Exclusion:** There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



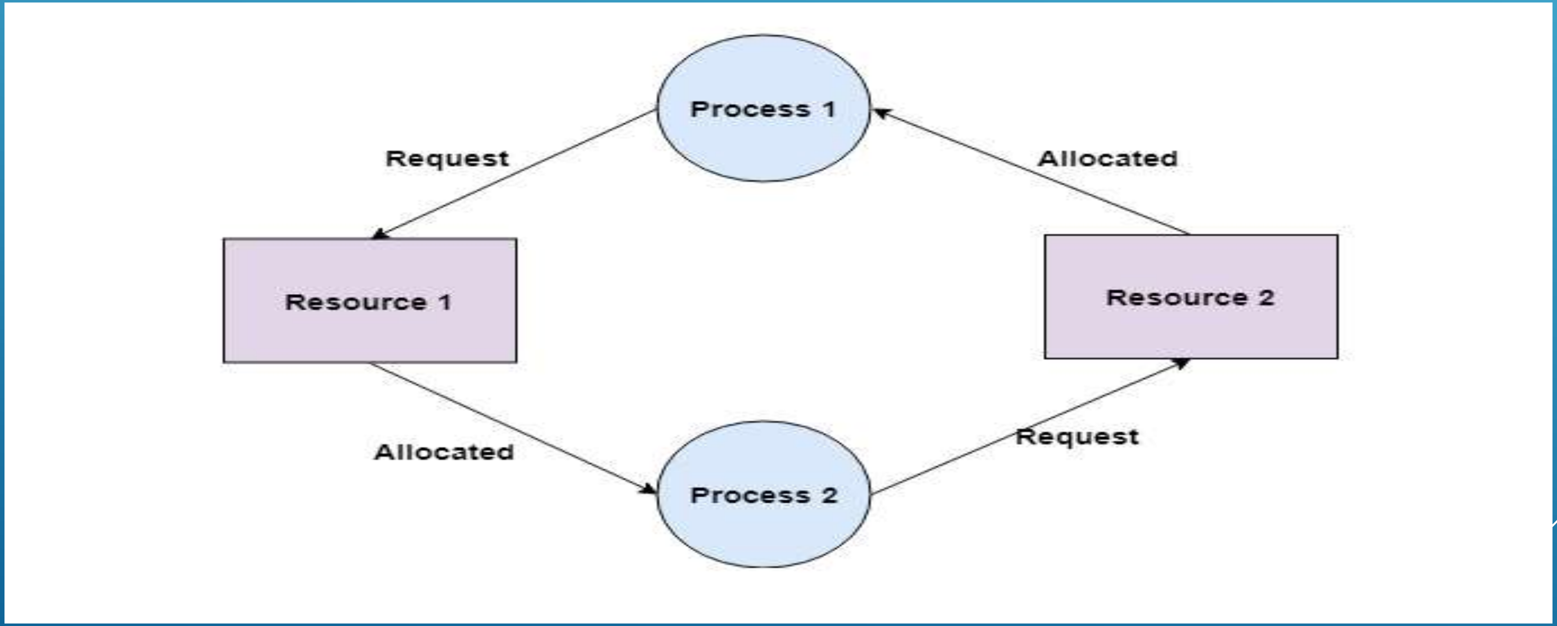
- **No Preemption:** A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



- **Hold and Wait:** A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



- **Circular Wait:** A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource 2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



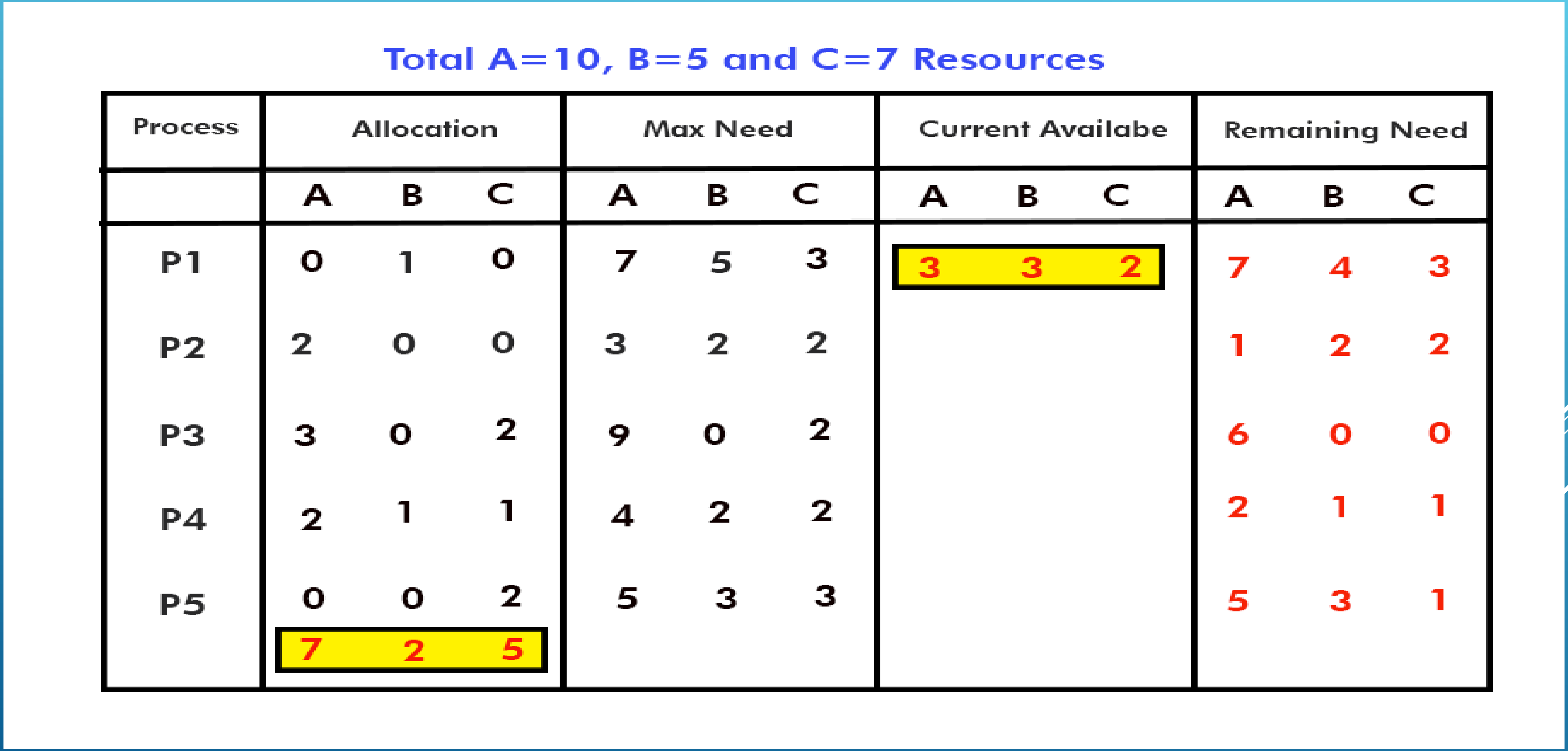
METHODS TO HANDLE DEADLOCK

- Deadlock Ignorance (OSTRICH ALGORITHM)

The Ostrich algorithm means that the deadlock is simply ignored and it is assumed that it will never occur. This is done because in some systems the cost of handling the deadlock is much higher than simply ignoring it as it occurs very rarely. So, it is simply assumed that the deadlock will never occur and the system is rebooted if it occurs by any chance.

- Deadlock Prevention - Deadlock can be prevented by eliminating any of the four necessary conditions, which are mutual exclusion, hold and wait, no preemption, and circular wait. Mutual exclusion, hold and wait and no preemption cannot be violated practically. Circular wait can be feasibly eliminated by assigning a priority to each resource.

➢ Deadlock Avoidance (BANKER's ALGORITHM)



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

Process	Allocation			Max Need			Current Availabe			Remaining Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2				1	2	2
P3	3	0	2	9	0	2				6	0	0
P4	2	1	1	4	2	2				2	1	1
P5	0	0	2	5	3	3				5	3	1
	7	2	5									

- Deadlock Detection and Recovery

Deadlock can be detected by the resource scheduler as it keeps track of all the resources that are allocated to different processes. After a deadlock is detected, it can be handed using the given methods –

- All the processes that are involved in the deadlock are terminated. This approach is not that useful as all the progress made by the processes is destroyed.
- Resources can be preempted from some processes and given to others until the deadlock situation is resolved.