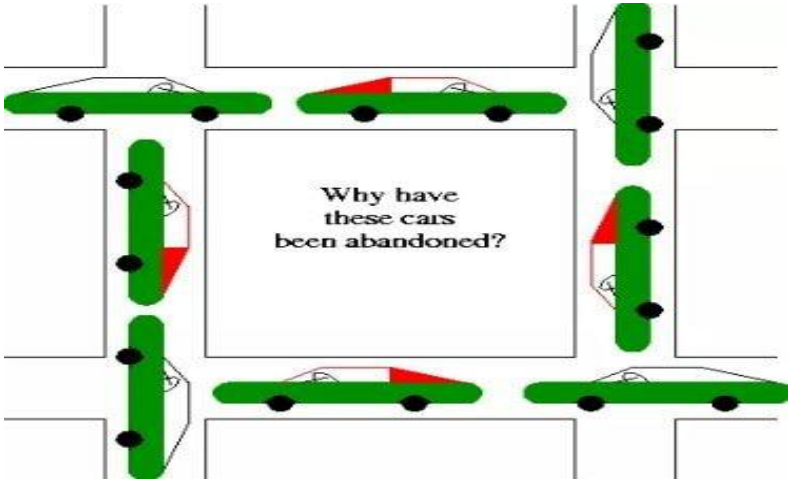
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER -5
DEADLOCK



Subject:- OS **Code:-3140702**	**Prepared by:** **Asst.Prof.Foram Patel** **(Computer Department,ACET)**	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

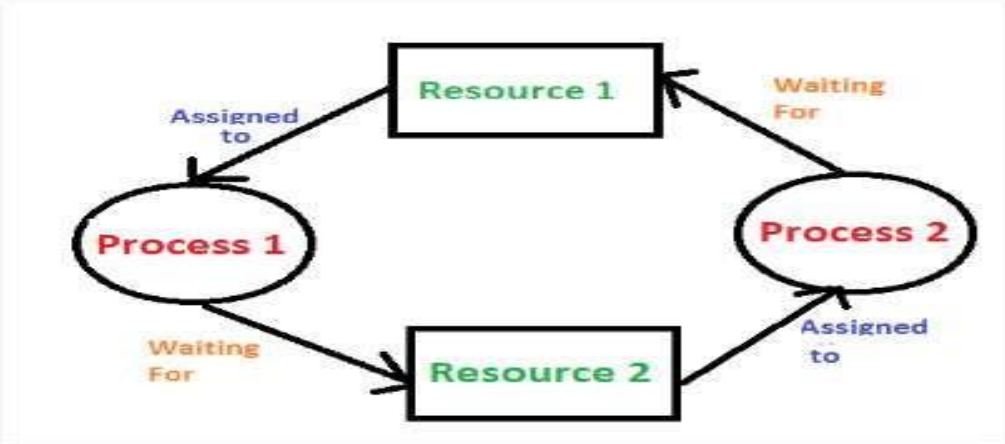
WHAT IS DEADLOCK?

- Deadlock can occur on sharable resources such as files , printers , database , memory, CPU etc.
- Deadlock is a situation where set of processes are blocked because one process is waiting for a resource which is held by other process.
- Process utilize the resource in following sequence.

1)Requests a resource

2)Use the resource

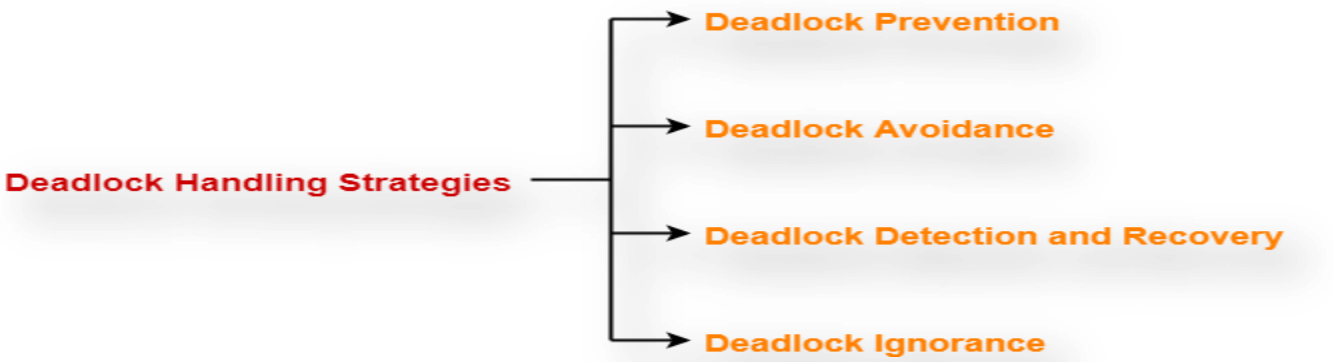
3) Releases the resource



TYPES OF RESOURCE

- **Reusable Resources:**
- It is used only by one process at a time.
- Process can release resource after use.
- Example: Processors , I/O device , Database, Primary and secondary Memory.
- **Consumable Resource:**
- Consumable resource is one that can be created and destroyed.
- There is no limit on the number of consumable resource.
- Example: Messages

DEADLOCK STRATEGIES



- **Deadlock Prevention:** Aim is to condition a system to remove any possibility of deadlock occurring.
- **Deadlock Avoidance:** Avoided by identifying safe state and unsafe state
- **Deadlock Detection:** The process of determining that a deadlock exists and identifying the process and resource involved in the deadlock.
- **Deadlock Recovery:** Used to resolve the deadlock from a system.

DEADLOCK PREVENTION

- Aim is to condition a system to remove any possibility of deadlock occurring.
- Following four conditions :

  1)Mutual Exclusion

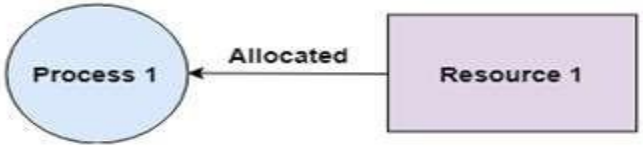
  2)Hold and Wait

  3)No Pre-emption

  4)Circular Wait

DEADLOCK PREVENTION

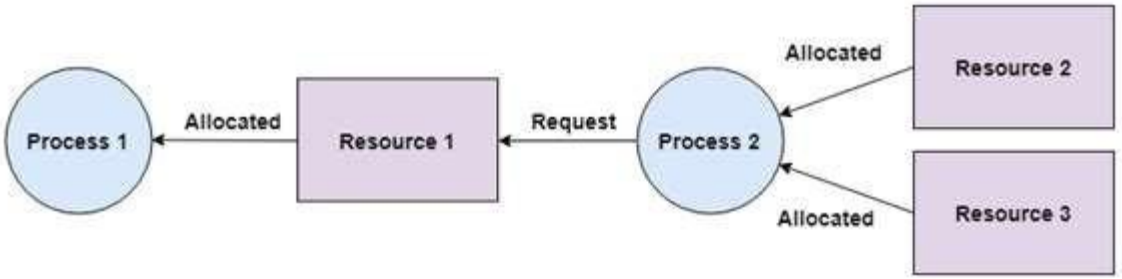
- **Mutual Exclusion:**
- A resource may be used by only one process at a time.
- If another process request that resources the requesting process will delayed until the resource has been released .



•In the diagram, there is a single instance of Resource 1 and it is held by Process 1 only.

DEADLOCK PREVENTION

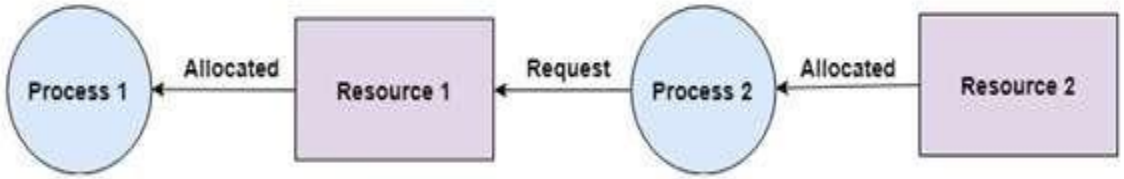
- **Hold and Wait:** A process that is holding atleast one resource & is waiting to get additional resource which are currently held by some other process.



•In the diagram given above, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.

DEADLOCK PREVENTION

- **No-pre emption:** Once a process has obtained a resource , the system can not remove it from the process control until the process has finished using the resource.

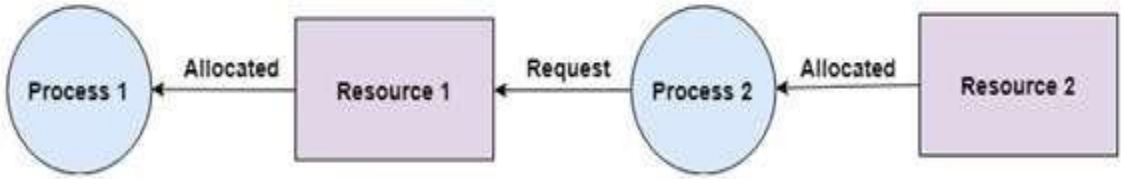


•In the diagram, Process 2 cannot pre-empt Resource 1 from Process.

•It will only be released when Process 1 release it voluntarily after its execution is complete.

DEADLOCK PREVENTION

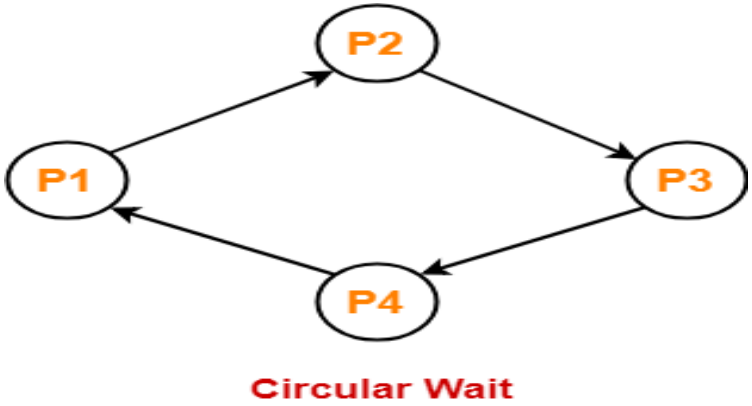
- **No-pre emption:** Once a process has obtained a resource , the system can not remove it from the process control until the process has finished using the resource.



•In the diagram, Process 2 cannot pre-empt Resource 1 from Process.

•It will only be released when Process 1 release it voluntarily after its execution is complete.

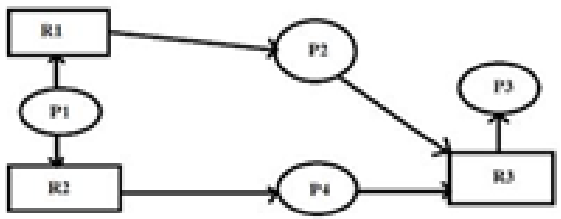
DEADLOCK PREVENTION



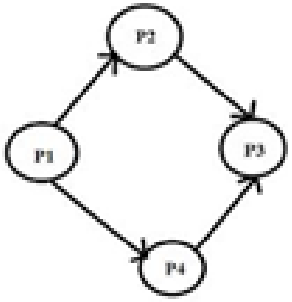
- **Circular wait:** A circular chain of hold and wait condition exists in the system.
- There exists a set {P0, P1, …, Pn} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and Pn is waiting for a resource that is held by P0.

DEADLOCK DETECTION

- The process of determining that a deadlock exists and identifying the process and resource involved in the deadlock.
- There is a deadlock in a system if and only if there is a loop in the wait for graph of that system.
- Wait for graph of a system is always smaller than the resource allocation graph.



Resource allocation graph



Wait for graph

DEADLOCK RECOVERY

- **Process Termination:**
- Deadlock is removed by aborting a process. All deadlocked processes are aborted.
- Selection of process for aborting is difficult. Circular wait is eliminated by aborting one by one process.
- **Resource pre-emption:**
- Resource temporarily take away from its current process and allocate it to another process.
- **For selection:**
- Priority of the process. Higher priority process are not selected.
- Process which is close to completion are not selected.

DEADLOCK RECOVERY

- **Recovery through Rollback :**
- When process in a system terminates, the system perform a rollback by undoing every operation.
- Use checkpoint.
- Check pointing a process means that its state is written to a file so that it can be restarted later.
- **Starvation :**
- Process waits for an event that might never occur in the system.

DEADLOCK AVOIDANCE

- The system must be able to decide whether granting a resource is safe or not.
- **Banker's algorithm:**
- Banker's algorithm is the deadlock avoidance algorithm
- Algorithm is check to see if granting the request to an unsafe state, If it does, the request is denied . If granting the request to a safe state, it is carried out.
- A safe state is not a deadlock state. Deadlock state is an unsafe state.
- **INPUTS:**
- Max free available resources in the system.
- Currently allocated resources by each process.
- Max need of resources by each process.

DEADLOCK AVOIDANCE

- **Available**: Available resource is equivalent to the total number of resource minus the sum of the allocation to all processes in the system.
- **Max**: Maximum number of resource that process requires during its execution.
- **Allocation**: Allocation is a table in which row represents process and column represents resource.
- Alloc[i , j]= Number of unit of resource Rj held by process Pi.
- **Need**: Process's need is equal to its maximum need minus its current allocation.
- **Need [i,j] = Max[i,j] – Allocation [i,j]**

EXAMPLE OF BANKER'S ALGORITHMS

- 5 processes P0 through P4;
- 3 resource types:
- A (10 instances), B (5instances), and C (7 instances)

	Allocation	Max	Available
	A B C	A B C	A B C
P0	0 1 0	7 5 3	3 3 2
P1	2 0 0	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

EXAMPLE OF BANKER'S ALGORITHMS

- Need of matrix is = Max – Allocation

NEED			
	A	B	C
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

- The system is in a safe state since the sequence < P1, P3, P4, P2, P0>