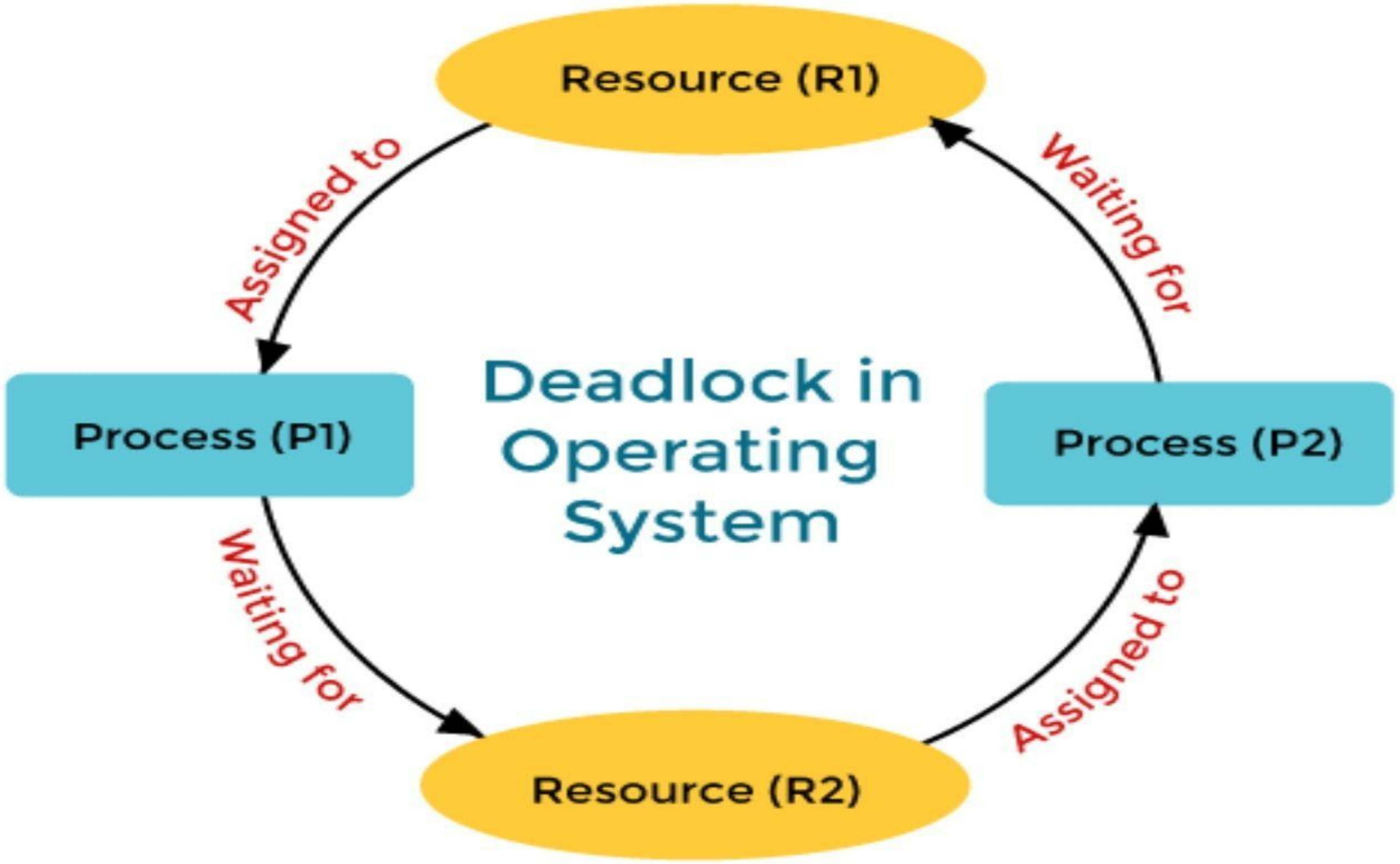
Deadlock

Definition
Characteristics
Conditions
Prevention
Avoidance
Detection & Recovery

DeadLock

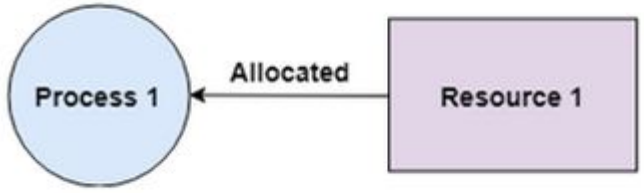

Resource (R1)
Assigned to
Waiting for
Process (P1)
Deadlock in Operating System
Process (P2)
Waiting for
Assigned to
Resource (R2)

DeadLock Conditions

- Mutual Exclusion
- Hold and Wait
- Circular Wait
- No Preemption

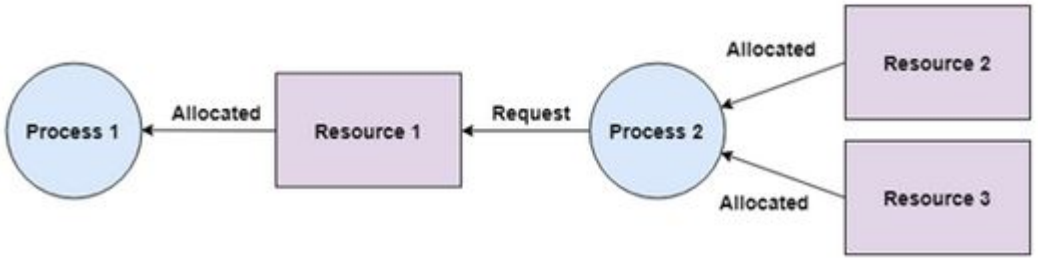
Mutual Exclusion

- Mutual Exclusion means at least maintain one resource as non-sharable state that is only one process can be use resource at a time. Other processes should wait until the allocated process to complete its work.



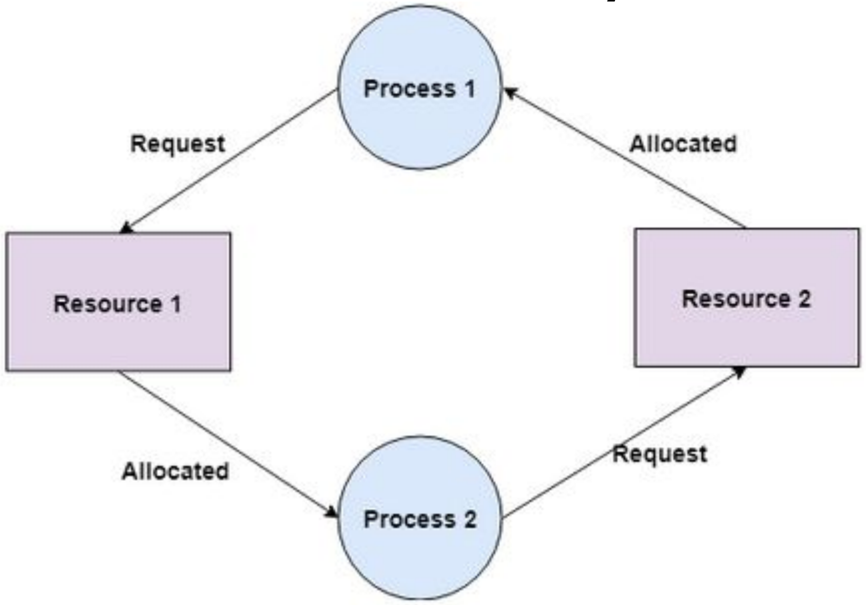
Hold and Wait

- Hold and Wait in Deadlock condition is preventing the process to hold one or more resources while other processes are waiting for the resources.



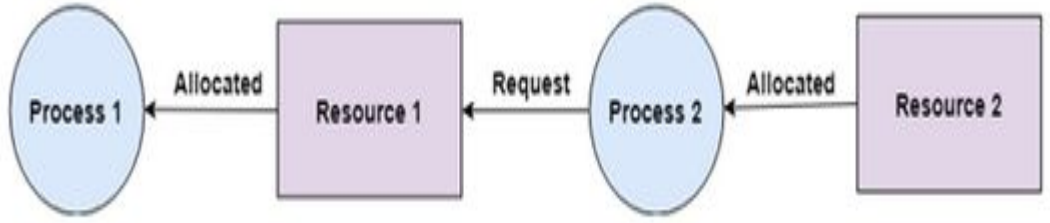
Circular Wait

- Circular Wait is the situation where one process is waiting for a Resource , which is hold by another process while that process is waiting for another process to release a Resource which is needed by it to complete its work.



No Preemption

- Once a Resource is allocated to a Process on Request , the process release the Resource only when it completed its work otherwise there is no way to take it from the process.



Dead lock prevention

- Deadlock prevention is a technique used in computer science to avoid situations where multiple processes or threads are blocked and unable to proceed because they are waiting for each other to release resources that they need to complete their tasks.
- Deadlock prevention techniques may include resource allocation graph, preemptive scheduling, detection and recovery, or a combination of these methods.

- Resource allocation graph is a technique used for preventing deadlocks in computer systems. In this technique, resources are represented as nodes in a directed graph, and the edges represent the requests for resources. There are two types of edges in the graph - request edges and assignment edges. A request edge points from a process to a resource that the process needs, while an assignment edge points from a resource to a process that has been allocated that resource.

- Preemptive scheduling is a technique used for preventing deadlocks in computer systems. In this technique, the system is designed to preempt a process that is holding a resource for too long. When a process is preempted, the system forces the process to release the resource, which can then be allocated to another process that needs it.

- Detection and recovery is a technique used for preventing deadlocks in computer systems. In this technique, the system periodically checks for the presence of deadlocks and takes appropriate steps to recover from them if they occur.

Resource allocation graph:Adv & DisAdv

- **Advantages**
- Simple and easy to understand
- Provides a visual representation of the system's resource usage
- Can be used to detect potential deadlocks
- **Disadvantages**
- May not be suitable for complex systems with a large number of resources
- Requires constant monitoring and analysis to prevent deadlocks
- May introduce overhead and impact system performance

Preemptive scheduling: adv & disadv

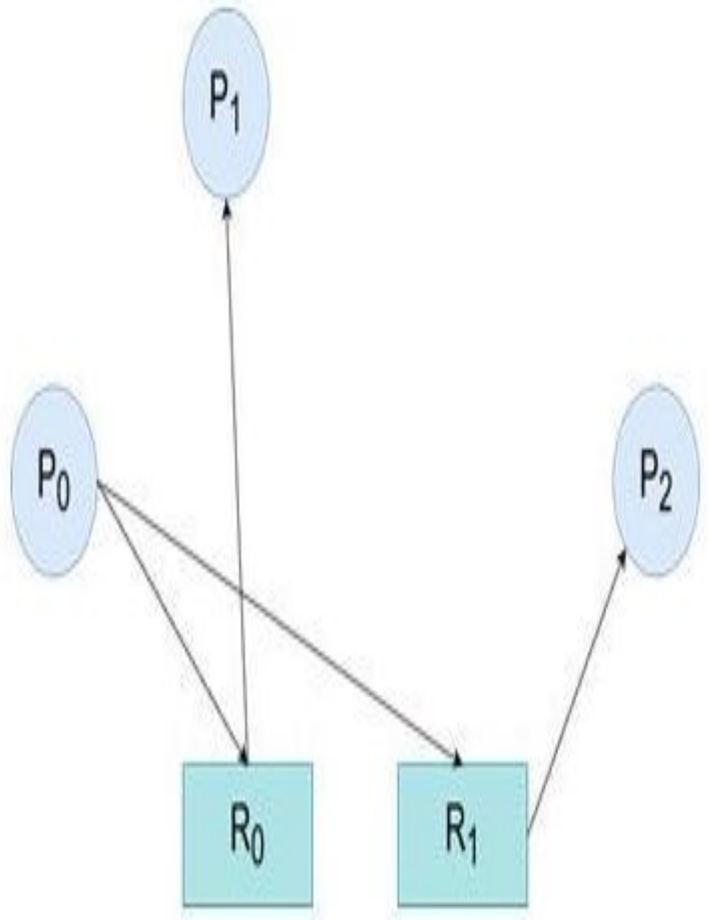
- **Advantages**
- Can be effective in preventing deadlocks
- Allows the system to preempt processes that are holding resources for too long
- Can be used in real-time systems
- **Disadvantages**
- May impact system performance if too many processes are preempted
- May not be suitable for all systems
- **Appropriate use case** − Preemptive scheduling is suitable for real-time systems where timely execution is critical, and the system can handle the overhead introduced by preemptive scheduling.

Detection and recovery: Adv & Disadv

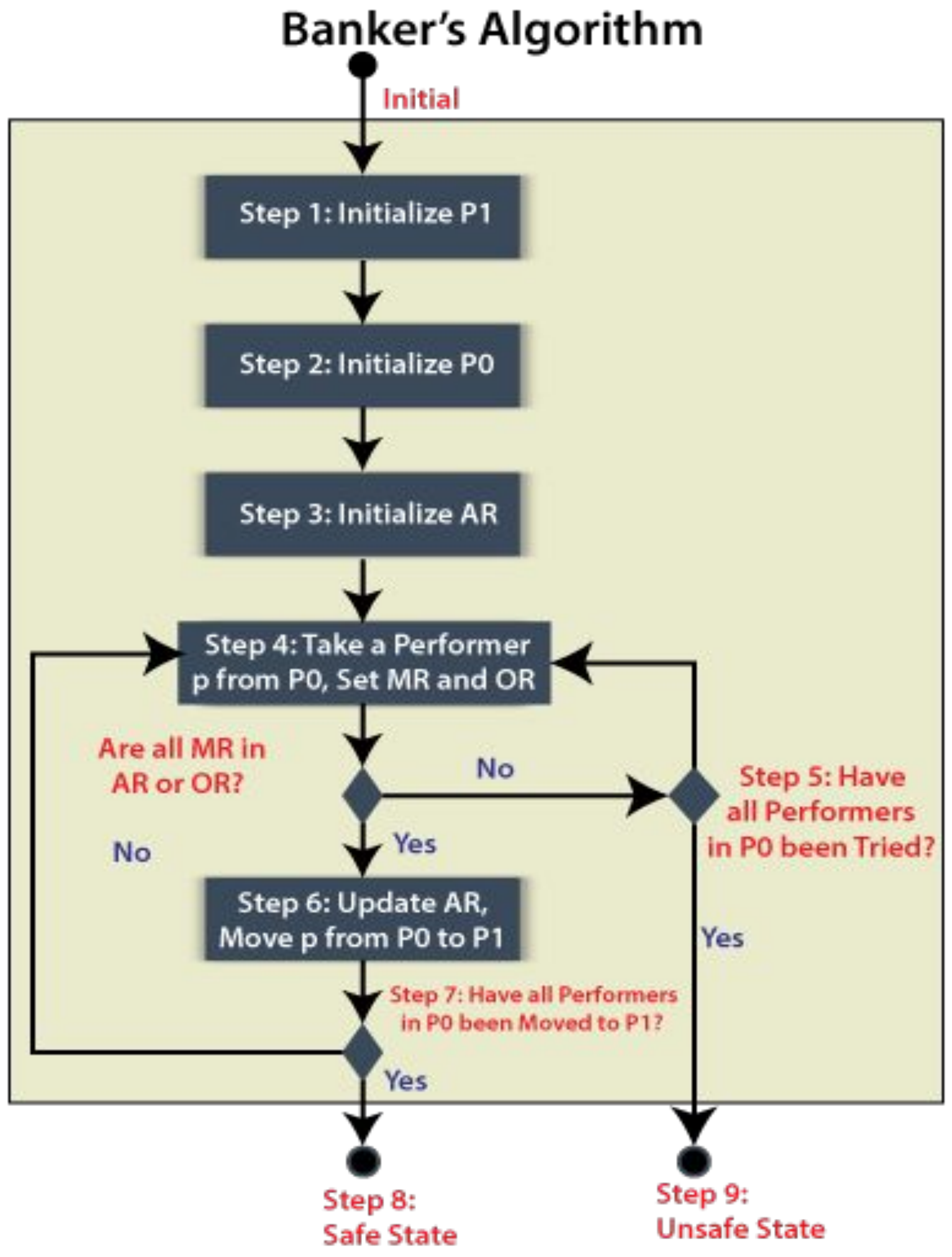
- **Advantages**
- Can detect and recover from deadlocks
- Allows the system to recover from deadlocks without disrupting system operations
- Can be used in a wide range of systems
- **Disadvantages**
- May introduce overhead and impact system performance
- May not be suitable for systems with strict performance requirements
- **Appropriate use case** – Detection and recovery is suitable for systems with a large number of resources and processes and can be used in a wide range of systems.

Dead lock avoidance

- When resource categories have only single instances of their resources, Resource- Allocation Graph Algorithm is used. In this algorithm, a cycle is a necessary and sufficient condition for deadlock.
- When resource categories have multiple instances of their resources, Banker’s Algorithm is used. In this algorithm, a cycle is a necessary but not a sufficient condition for deadlock.



Resource-Allocation Graph


Banker's Algorithm
Initial
Step 1: Initialize P1
Step 2: Initialize P0
Step 3: Initialize AR
Step 4: Take a Performer p from P0, Set MR and OR
Are all MR in AR or OR?
No
Step 5: Have all Performers in P0 been Tried?
Yes
No
Step 6: Update AR, Move p from P0 to P1
Yes
Step 7: Have all Performers in P0 been Moved to P1?
Yes
Step 8: Safe State
Step 9: Unsafe State

Dead lock detection and recovery