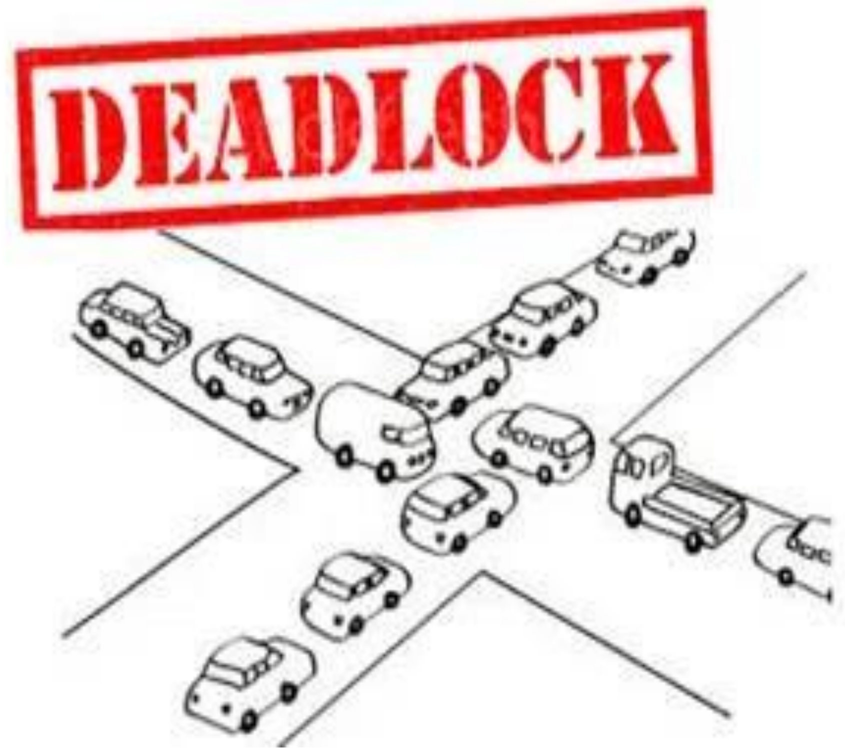
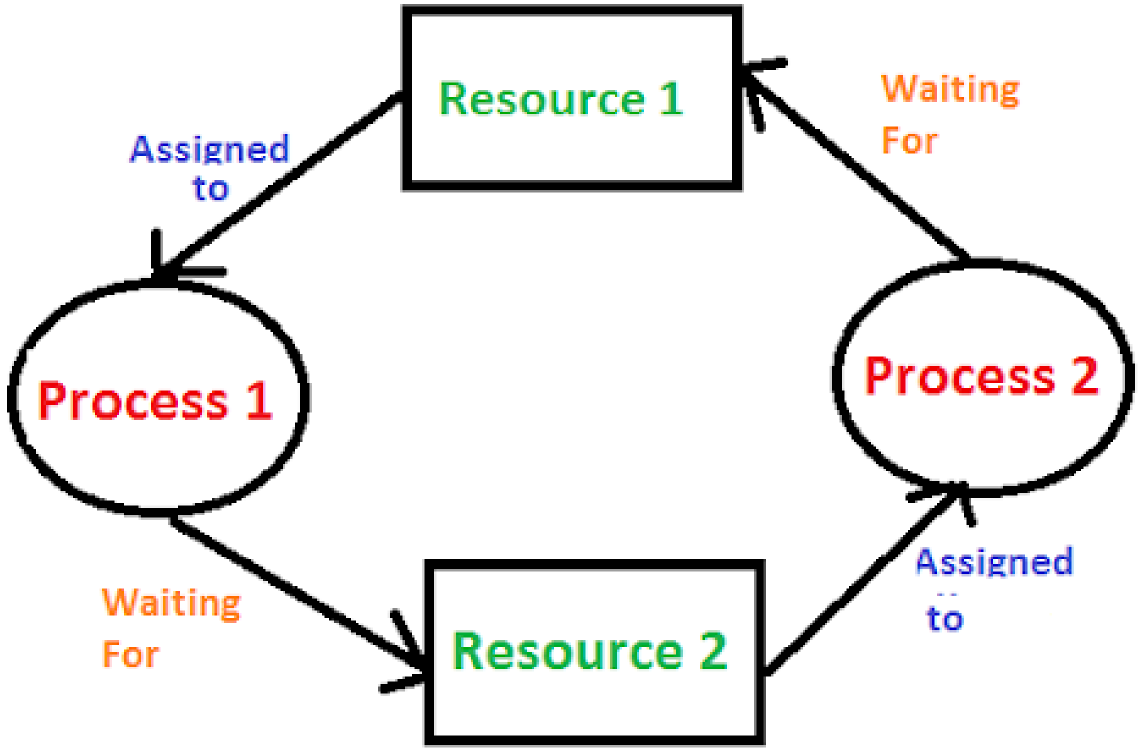
Deadlock in DBMS

Deadlock

- A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks.
- but none of the task is willing to give up the resources that other task needs.
- In this situation no task ever gets finished and is in waiting state forever



Example of deadlock:



- Process P1 holds resource R1 and waiting for resource R2
- Process P2 holds resource R2 and waiting for resource R1
- Deadlock

Conditions of Deadlock:

- Coffman stated four conditions for a deadlock occurrence. A deadlock may occur if all the following conditions holds true.
- **Mutual exclusion**
- **Hold and wait**
- **No preemption**
- **Circular wait**

- **Mutual exclusion condition**: There must be at least one resource that cannot be used by more than one process at a time.
- **Hold and wait condition**: A process that is holding a resource can request for additional resources that are being held by other processes in the system.
- **No preemption condition**: A resource cannot be forcibly taken from a process. Only the process can release a resource that is being held by it.
- **Circular wait condition**: A condition where one process is waiting for a resource that is being held by second process and second process is waiting for third process ….so on and the last process is waiting for the first process. Thus making a circular chain of waiting.

Deadlock Handling

1. Deadlock Ignorance-(Ostrich algorithm)

- Pretend that there's no problem
- Deadlock Ignorance is the most widely used approach among all the mechanism.
- In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock.
- This approach is best suitable for a single end user system where User uses the system only for browsing and all other normal stuff.

2. Deadlock prevention

- Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously. If it is possible to violate one of the four conditions at any time, then the deadlock can never occur in the system.

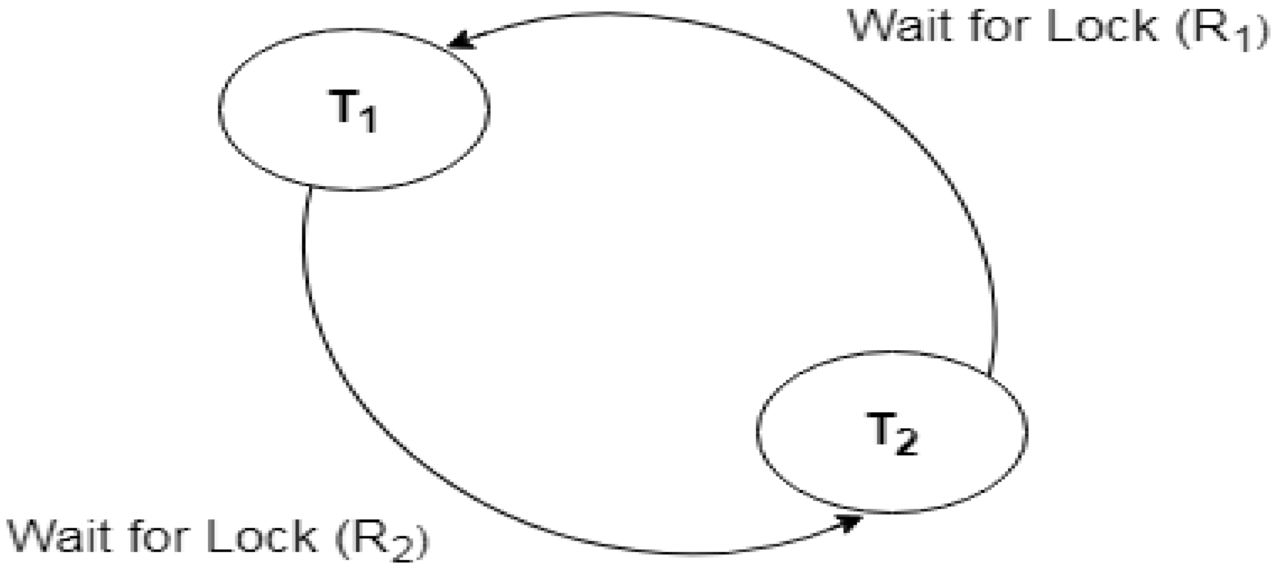
- **Removing mutual exclusion**: All resources must be sharable that means at a time more than one processes can get a hold of the resources. That approach is practically impossible.
- **Removing hold and wait condition**: This can be removed if the process acquires all the resources that are needed before starting out. Another way to remove this to enforce a rule of requesting resource when there are none in held by the process.
- **Preemption of resources**: Preemption of resources from a process can result in rollback and thus this needs to be avoided in order to maintain the consistency and stability of the system.
- **Avoid circular wait condition**: This can be avoided if the resources are maintained in a hierarchy and process can hold the resources in increasing order of precedence. This avoid circular wait. Another way of doing this to force one resource per process rule – A process can request for a resource once it releases the resource currently being held by it. This avoids the circular wait.

3.Deadlock Detection

- In a database, when a transaction waits indefinitely to obtain a lock, then the DBMS should detect whether the transaction is involved in a deadlock or not.
- The lock manager maintains a Wait for the graph to detect the deadlock cycle in the database.

Wait for Graph

- This is the suitable method for deadlock detection. In this method, a graph is created based on the transaction and their lock. If the created graph has a cycle or closed loop, then there is a deadlock.


Wait for Lock (R_1)
T_1
T_2
Wait for Lock (R_2)