**DEADLOCK RECOVERY**

When a detection algorithm determines that a deadlock exists, several alternatives are available. There are two options for breaking a deadlock. One is simply to abort one or more processes to break the circular wait. The other is to preempt some resources from one or more of the deadlocked processes.

**1)Process Termination:**

Process Termination is one method to recover from deadlock. We can use two options for Process Termination and these are:.  
i. Abort all deadlocked processes.

ii. Abort one by one process until the deadlock cycle is eliminated.

**i. Abort all deadlocked processes:**

This method clearly will break the deadlock cycle, but it is great expensive method. It means that kill all processes in the deadlocked state and all processes will execute again from starting. The disadvantage of this method is The deadlocked processes may have computed for a long time, and the results of these partial computations must be discarded and probably will have to be recomputed later.

**ii. Abort one by one process until the deadlock cycle is eliminated:**

Abort one deadlocked process at a time, until the [deadlock](https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/)cycle is eliminated from the system. Due to this method, there may be considerable overhead, because, after aborting each process, we have to run a[deadlock detection algorithm](https://www.geeksforgeeks.org/deadlock-detection-algorithm-in-operating-system/) to check whether any processes are still deadlocked.

If the partial termination method is used, then we must determine which deadlocked process should be terminated. Abort those processes whose termination will incur minimum costs.

Many factors may affect which process is chosen including:

1. What the priority of the process is

2. How long the process has computed and how much longer the process

will compute before completing its designated task

3. How many and what types of resources the process has used

4. How many more resources the process needs in order to complete

5. How many processes will need to be terminated.

6. Whether the process is interactive or batch.

**2) Resource Preemption:**

To eliminate deadlocks using resource preemption, we successively preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.

If preemption is required to deal with deadlocks, then three issues need to be addressed:

**i) Selecting a victim**:

we must determine which resource(s) is/are to be pre-empted from which process. As in process termination, we must determine the order of preemption to minimize cost. Cost factors may include such parameters as the number of resources a deadlocked process is holding and the amount of time the process has thus far consumed.

**ii)Rollback:**

If we preempt a resource from a process, what should be done with that process? Clearly, it cannot continue with its normal execution; it is missing some needed resource. We must roll back the process to some safe state and restart it from that state. This method requires the system to keep more information about the state of all the running processes. It is difficult to determine what a safe state is, the simplest solution is a total rollback: abort the process and then restart it.

**iii)Starvation:**

In a system, it may happen that the resources to preempt from same process is always picked as a victim. As a result, that process will never complete its designated task. This situation is called [**Starvation**](https://www.geeksforgeeks.org/difference-between-deadlock-and-starvation-in-os/)and must be avoided. The most common solution is to include the number of rollbacks in the cost factor. That means a process must be picked as a victim only a finite number of times.