**Deadlock Avoidance is a process used by the Operating System to avoid Deadlock.** Let's first understand what is Deadlock in Operating System. Deadlock is a situation that occurs in Operating System when any Process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software.

**But how can Operating System avoid Deadlock?**

Operating System avoids Deadlock by knowing the maximum resources requirements of the processes initially, and also Operating System knows the free resources available at that time. Operating System tries to allocate the resources according to the process requirements and checks if the allocation can lead to a safe state or an unsafe state. If the resource allocation leads to an unsafe state, then Operating System does not proceed further with the allocation Sequence.

How does this happen? How does the Operating System allocate resources to the processes? We will try to understand the process of allocation of resources with an intuitive example further in this article.

**How does Deadlock avoidance work?**

Let's understand the working of Deadlock Avoidance with the help of an intuitive example.

| **Process** | **Maximum Required** | **current Available** | **Need** |
| --- | --- | --- | --- |
| P1 | 9 | 5 | 4 |
| P2 | 5 | 2 | 3 |
| P3 | 3 | 1 | 2 |

Let's consider three processes P1, P2, P3. Some more information on which the processes tells the Operating System are :

* P1 process needs a maximum of 9 resources (Resources can be any software or hardware Resource like tape drive or printer etc..) to complete its execution. P1 is currently allocated with 5 Resources and needs 4 more to complete its execution.
* P2 process needs a maximum of 5 resources and is currently allocated with 2 resources. So it needs 3 more resources to complete its execution.
* P3 process needs a maximum of 3 resources and is currently allocated with 1 resource. So it needs 2 more resources to complete its execution.
* Operating System knows that only 2 resources out of the total available resources are currently free.

**But only 2 resources are free now**. Can P1, P2, and P3 satisfy their requirements? Let's try to find out.

As only 2 resources are free for now, then only P3 can satisfy its need for 2 resources. If P3 takes 2 resources and completes its execution, then P3 can release its 3 (1+2) resources. Now the three free resources which P3 released can satisfy the need of P2. Now, P2 after taking the three free resources, can complete its execution and then release 5 (2+3) resources. Now five resources are free. P1 can now take 4 out of the 5 free resources and complete its execution. So, with 2 free resources available initially, all the processes were able to complete their execution leading to Safe State. The order of execution of the processes was <P3, P2, P1>.

What if initially there was only 1 free resource available? None of the processes would be able to complete its execution. Thus leading to an unsafe state.

We use two words, safe and unsafe states. What are those states? Let's understand these concepts.

**Safe State and Unsafe State**

**Safe State** - In the above example, we saw that Operating System was able to satisfy the need of all three processes, P1, P2, and P3, with their resource requirements. So all the processes were able to complete their execution in a certain order like P3->P2->P1.

So, **If Operating System is able to allocate or satisfy the maximum resource requirements of all the processes in any order then the system is said to be in Safe State.**

**So safe state does not lead to Deadlock**.

**Unsafe State** - If Operating System is not able to prevent Processes from requesting resources which can also lead to Deadlock, then the System is said to be in an Unsafe State.

**Unsafe State does not necessarialy cause deadlock it may or maynot causes deadlock**.

**Deadlock Avoidance Example**

Let's take an example that has multiple resources requirement for every Process. Let there be three Processes P1, P2, P3, and 4 resources R1, R2, R3, R4. The maximum resources requirements of the Processes are shown in the below table.

| **Process** | **R1** | **R2** | **R3** | **R4** |
| --- | --- | --- | --- | --- |
| P1 | 3 | 2 | 3 | 2 |
| P2 | 2 | 3 | 1 | 4 |
| P3 | 3 | 1 | 5 | 0 |

A number of currently allocated resources to the processes are:

| **Process** | **R1** | **R2** | **R3** | **R4** |
| --- | --- | --- | --- | --- |
| P1 | 1 | 2 | 3 | 1 |
| P2 | 2 | 1 | 0 | 2 |
| P3 | 2 | 0 | 1 | 0 |

The total number of resources in the System are :

| **R1** | **R2** | **R3** | **R4** |
| --- | --- | --- | --- |
| 7 | 4 | 5 | 4 |

We can find out the no of available resources for each of P1, P2, P3, P4 by subtracting the currently allocated resources from total resources.

Available Resources are :

| **R1** | **R2** | **R3** | **R4** |
| --- | --- | --- | --- |
| 2 | 1 | 1 | 1 |

Now, The need for the resources for the processes can be calculated by :

Need = Maximum Resources Requirement - Currently Allocated Resources.

The need for the Resources is shown below:

| **Process** | **R1** | **R2** | **R3** | **R4** |
| --- | --- | --- | --- | --- |
| P1 | 2 | 1 | 0 | 1 |
| P2 | 0 | 2 | 1 | 2 |
| P3 | 1 | 1 | 4 | 0 |

The available free resources are <2,1,1,1> of resources of R1, R2, R3, and R4 respectively, which can be used to satisfy only the requirements of process P1 only initially as process P2 requires 2 R2 resources which are not available. The same is the case with Process P3, which requires 4 R3 resources which is not available initially.

The Steps for resources allotment is explained below:

1. Firstly, Process P1 will take the available resources and satisfy its resource need, complete its execution and then release all its allocated resources. Process P1 is initially allocated <1,2,3,1> resources of R1, R2, R3, and R4 respectively. Process P1 needs <2,1,0,1> resources of R1, R2, R3 and R4 respectively to complete its execution. So, process P1 takes the available free resources <2,1,1,1> resources of R1, R2, R3, R4 respectively and can complete its execution and then release its current allocated resources and also the free resources it used to complete its execution. Thus P1 releases <1+2,2+1,3+1,1+1> = <3,3,4,2> resources of R1, R2, R3, and R4 respectively.
2. After step 1 now, available resources are now <3,3,4,2>, which can satisfy the need of Process P2 as well as process P3. After process P2 uses the available Resources and completes its execution, the available resources are now <5,4,4,4>.
3. Now, the available resources are <5,4,4,4>, and the only Process left for execution is Process P3, which requires <1,1,4,0> resources each of R1, R2, R3, and R4. So it can easily use the available resources and complete its execution. After P3 is executed, the resources available are <7,4,5,4>, which is equal to the maximum resources or total resources available in the System.

So, **the process execution sequence in the above example was <P1, P2, P3>**. But it could also have been <P1, P3, P2> if process P3 would have been executed before process P2, which was possible as there were sufficient resources available to satisfy the need of both Process P2 and P3 after step 1 above.

**Deadlock Avoidance Solution**

Deadlock Avoidance can be solved by two different algorithms:

* **Resource allocation Graph**
* **Banker's Algorithm**

We will discuss both algorithms in detail in their separate article.

**Resource Allocation Graph**

Resource Allocation Graph (RAG) is used to represent the state of the System in the form of a Graph. The Graph contains all processes and resources which are allocated to them and also the requesting resources of every Process. Sometimes if the number of processes is less, We can easily identify a deadlock in the System just by observing the Graph, which can not be done easily by using tables that we use in Banker's algorithm.

**Banker's Algorithm**

Banker's algorithm does the same as we explained the Deadlock avoidance with the help of an example. The algorithm predetermines whether the System will be in a safe state or not by simulating the allocation of the resources to the processes according to the maximum available resources. **It makes an "s-state" check before actually allocating the resources to the Processes**.

**When there are more number of Processes and many Resources, then Banker's Algorithm is useful.**

**Conclusion**

* Deadlock Avoidance is used by Operating System to avoid Deadlock by using resource allocation Graph or by Banker's algorithm.
* Deadlock Avoidance work by letting the Operating System know the Process requirements of resources to complete their execution, and accordingly operating system checks if the requirements can be satisfied or not.
* If all the resources required of the Process are satisfied with the available resources, then the System is said to be in a Safe state.
* If all the resources requirements of the Process are not satisfied with the available resources in any possible way, then the System is said to be in an unsafe state.
* **Safe state does not cause Deadlock**.
* **Unsafe state may or may not cause Deadlock**.
* Resource Allocation Graph is used to represent all the Processes and resource states in the form of a graph.
* Banker's Algorithm checks for the s-state(safe state or unsafe state) before actually allocating the resources to the Processes.

**DEADlock Recovery**

For Process:-

**Kill a process**

* + Killing a process can solve our problem but the bigger concern is to decide which process to kill. Generally, Operating system kills a process which has done least amount of work until now.
* **Kill all process**
  + This is not a suggestible approach but can be implemented if the problem becomes very serious. Killing all process will lead to inefficiency in the system because all the processes will execute again from starting.

**For Resource**

**Preempt the resource**

* We can snatch one of the resources from the owner of the resource (process) and give it to the other process with the expectation that it will complete the execution and will release this resource sooner. Well, choosing a resource which will be snatched is going to be a bit difficult.

**Rollback to a safe state**

* System passes through various states to get into the deadlock state. The operating system can rollback the system to the previous safe state. For this purpose, OS needs to implement check pointing at every state.
* The moment, we get into deadlock, we will rollback all the allocations to get into the previous safe state.