Introduction of Deadlock in Operating System

A process in operating systems uses different resources and uses resources in following way.

1) Requests a resource

2) Use the resource

2) Releases the resource

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Consider an example when two trains are coming toward each other on same track and there is only one track, none of the trains can move once they are in front of each other. Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.
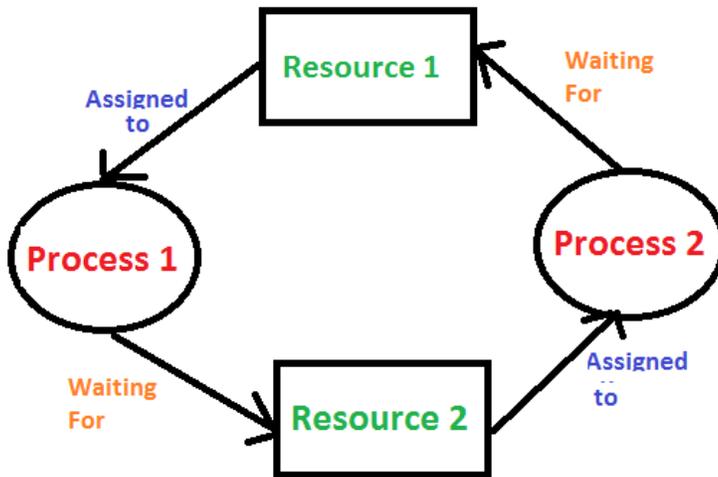
Deadlock can arise if following four conditions hold simultaneously (Necessary Conditions)

Mutual Exclusion: One or more than one resource are non-sharable (Only one process can use at a time)

Hold and Wait: A process is holding at least one resource and waiting for resources.

No Preemption: A resource cannot be taken from a process unless the process releases the resource.

Circular Wait: A set of processes are waiting for each other in circular form.

Methods for handling deadlock

There are three ways to handle deadlock

1) Deadlock prevention or avoidance: The idea is to not let the system into deadlock state.

One can zoom into each category individually, Prevention is done by negating one of above mentioned necessary conditions for deadlock.

Avoidance is kind of futuristic in nature. By using strategy of "Avoidance", we have to make an assumption. We need to ensure that all information about resources which process WILL need are known to us prior to execution of the process. We use Banker's algorithm (Which is in-turn a gift from Dijkstra) in order to avoid deadlock.

2) Deadlock detection and recovery: Let deadlock occur, then do preemption to handle it once occurred.

3) Ignore the problem all together: If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

Deadlock Prevention And Avoidance

Deadlock Characteristics

As discussed in the previous post, deadlock has following characteristics.

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

Deadlock Prevention

We can prevent Deadlock by eliminating any of the above four conditions.

Eliminate Mutual Exclusion

It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

Eliminate Hold and wait

1. Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. for example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.

2. The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

HOLD AND WAIT
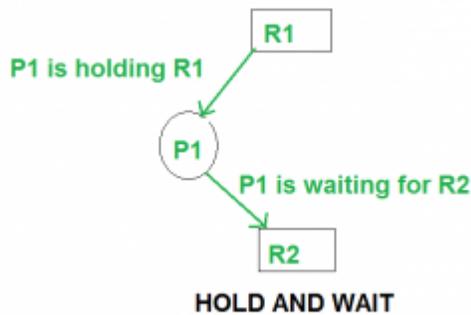
Eliminate No Preemption

Preempt resources from the process when resources required by other high priority processes.

Eliminate Circular Wait

Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing. order of numbering.

For Example, if P1 process is allocated R5 resources, now next time if P1 ask for R4, R3 lesser than R5 such request will not be granted, only request for resources more than R5 will be granted.

Deadlock Avoidance

Deadlock avoidance can be done with Banker's Algorithm.

Banker's Algorithm

Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

Inputs to Banker's Algorithm:

1. Max need of resources by each process.
2. Currently allocated resources by each process.
3. Max free available resources in the system.

The request will only be granted under the below condition:

1. If the request made by the process is less than equal to max need to that process.
2. If the request made by the process is less than equal to the freely available resource in the system.

Example:

```
Total resources in system:
A B C D
6 5 7 6


Available system resources are:
A B C D
3 1 1 2


Processes (currently allocated resources):
    A B C D
P1  1 2 2 1
P2  1 0 3 3
P3  1 2 1 0


Processes (maximum resources):
    A B C D
P1  3 3 2 2
P2  1 2 3 4
P3  1 3 5 0


Need = maximum resources - currently allocated resources.
Processes (need resources):
    A B C D
P1  2 1 0 1
P2  0 2 0 1
P3  0 1 4 0
```

Deadlock Detection And Recovery

In the previous post, we have discussed Deadlock Prevention and Avoidance. In this post, Deadlock Detection and Recovery technique to handle deadlock is discussed.

Deadlock Detection

1. If resources have single instance:
   In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.
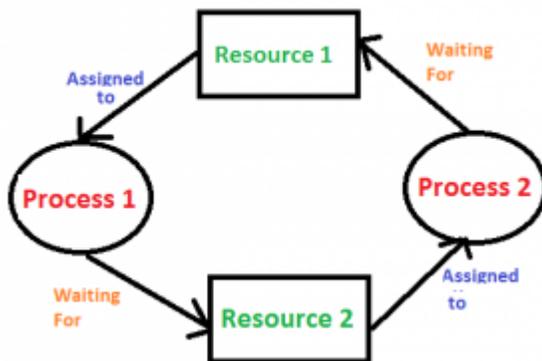
   In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.

2. If there are multiple instances of resources:
   Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

Deadlock Recovery

A traditional operating system such as Windows doesn't deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use Deadlock recovery.

Recovery method

1. **Killing the process:** killing all the process involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till system recover from deadlock.
2. **Resource Preemption:** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.