Subject Name: System Design trouble shooting and Operating System

Semester/Year—2nd SEM

Name of the Teacher:-Beauty Sarkar

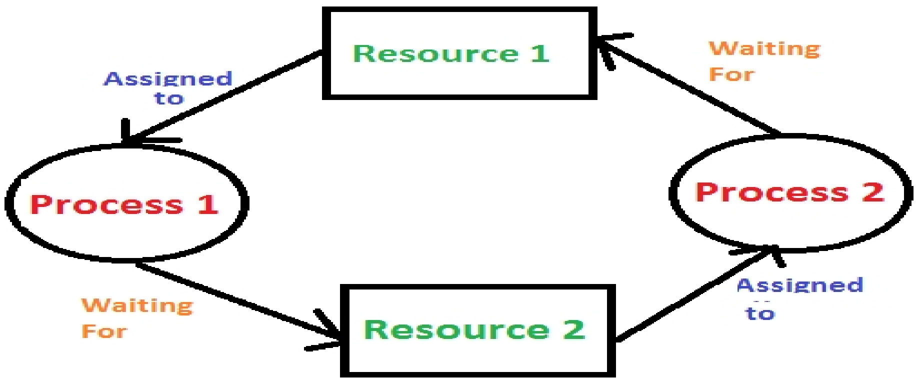
Topic: - DEADLOCK

❖ Introduction to Deadlock:

A process in operating systems uses different resources and uses resources in following way.
1) Requests a resource 2) Use the resource 3) Releases the resource

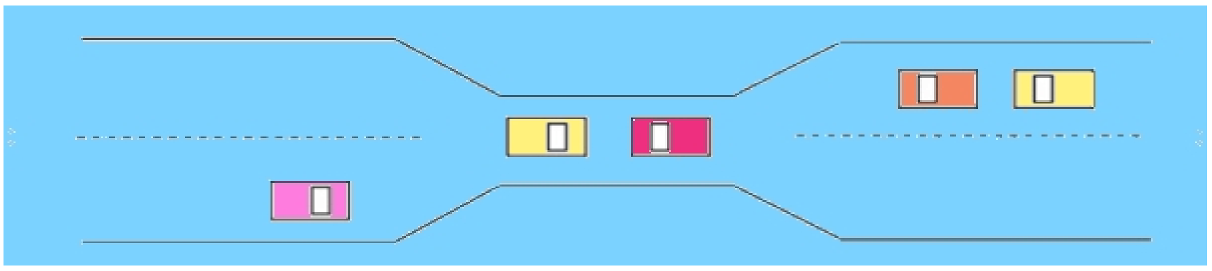
In an operating system, a deadlock is the occurs when a process enters into a waiting state because a resource request is being made by the other waiting process, which in turn become a waiting status for the other resource. When the process is unable to change its state indefinitely because the resources requested by it are being used by another waiting process, then the system in a deadlock state.

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.



❖ Another Example of Deadlock:-

- A real-world example would be traffic, which is going only in one direction.
- Here, a bridge is considered a resource.
- So, when Deadlock happens, it can be easily resolved if one car backs up (Preempt resources and rollback).
- Several cars may have to be backed up if a deadlock situation occurs.
- So starvation is possible.



[**Note: Starvation** is a situation where all the low priority processes got blocked, and the high priority processes proceed. In any system, requests for high/low priority resources keep on happening dynamically. Thereby, some policy is require to decide who gets support when.

Using some algorithms, some processes may not get the desired serviced even though they are not deadlocked. Starvation occurs when some threads make shared resources unavailable for a long period of time.]

❖ 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

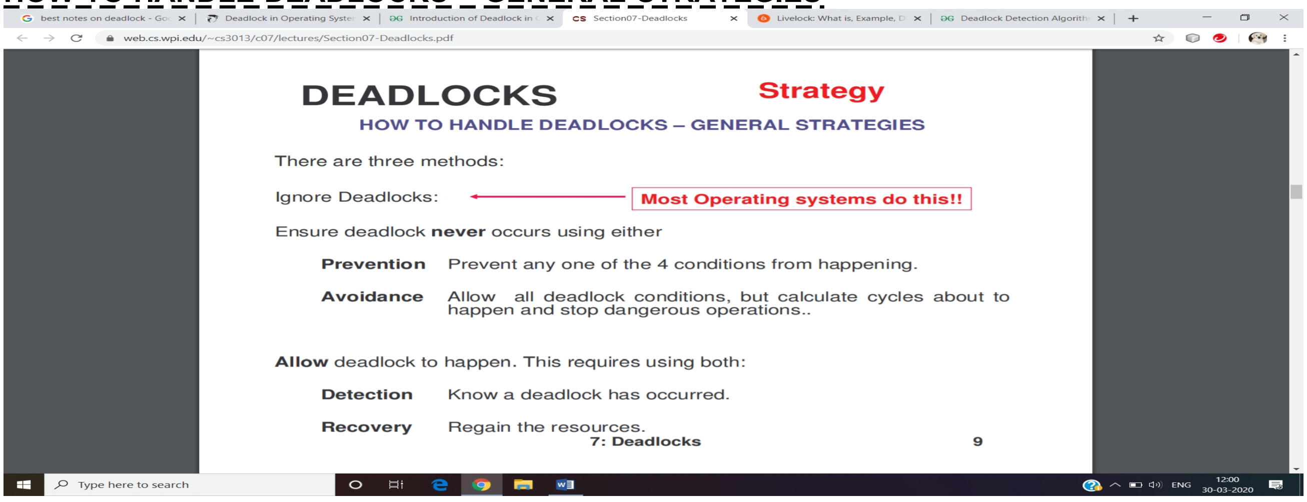
There are three ways to handle deadlock:

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. When the time interval between the occurrences of deadlocks are seems to be large in addition to the increment in data loss incurred each time is tolerable then in such case Ignoring deadlock is used. This is the approach that both Windows and UNIX take.

❖ HOW TO HANDLE DEADLOCKS – GENERAL STRATEGIES


best notes on deadlock - Go
Deadlock in Operating Syster
Introduction of Deadlock in
Section07-Deadlocks
Livelock: What is, Example, D
Deadlock Detection Algorith
web.cs.wpi.edu/~cs3013/c07/lectures/Section07-Deadlocks.pdf
Type here to search
ENG
12:00
30-03-2020

DEADLOCKS **Strategy**

HOW TO HANDLE DEADLOCKS – GENERAL STRATEGIES

There are three methods:

Ignore Deadlocks: ← **Most Operating systems do this!!**

Ensure deadlock **never** occurs using either

Prevention Prevent any one of the 4 conditions from happening.

Avoidance Allow all deadlock conditions, but calculate cycles about to happen and stop dangerous operations..

Allow deadlock to happen. This requires using both:

Detection Know a deadlock has occurred.

Recovery Regain the resources.

7: Deadlocks **9**

❖ Resource Allocation Graph (RAG)

Resource allocation graph is explained to us what is the state of the system in terms of processes and resources. i.e how many resources are available, how many are allocated and what is the request of each process. Everything can be represented in terms of the diagram. One of the advantages of having a diagram is, sometimes it is possible to see a deadlock directly by using RAG, but then you might not be able to know that by looking at the table. But the tables are better if the system contains lots of process and resource and Graph is better if the system contains less number of process and resource.

We know that any graph contains vertices and edges. So RAG also contains vertices and edges. In RAG vertices are two type –

1. Process vertex – Every process will be represented as a process vertex. Generally, the process will be represented with a circle.

2. Resource vertex – Every resource will be represented as a resource vertex. It is also two type –

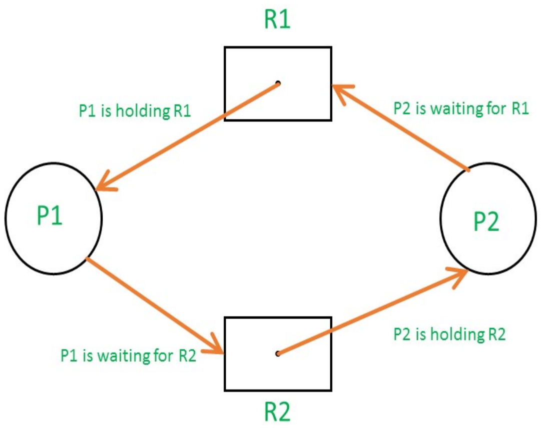
Single instance type resource – It represents as a box, inside the box, there will be one dot. So the number of dots indicate how many instances are present of each resource type.

Multi-resource instance type resource – It also represents as a box, inside the box, there will be many dots present.

Now coming to the edges of RAG. There are two types of edges in RAG –

1. Assign Edge – If you already assign a resource to a process then it is called Assign edge.

2. Request Edge – It means in future the process might want some resource to complete the execution that is called request edge.



SINGLE INSTANCE RESOURCE TYPE WITH DEADLOCK

A visual (mathematical) way to determine if a deadlock has, or may occur.

G = (V, E) The graph contains nodes and edges.

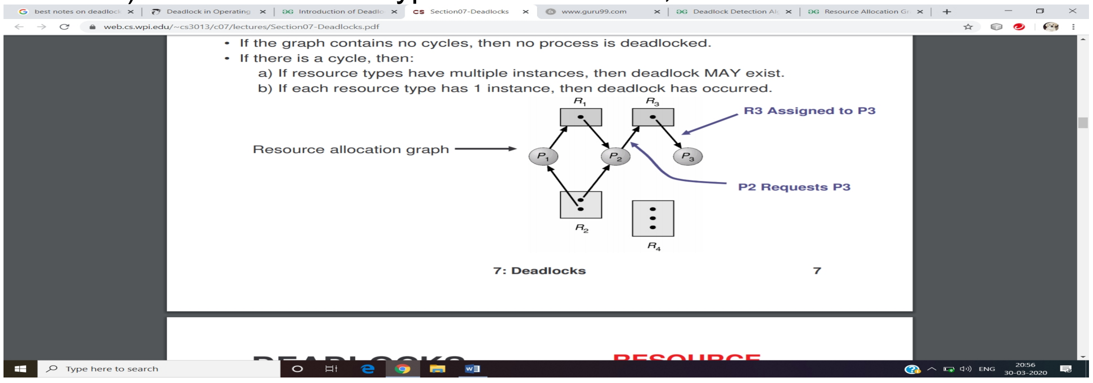
V Nodes consist of processes = { P1, P2, P3, ...} and resource types { R1, R2, ...}

E Edges are (Pi, Rj) or (Ri, Pj)

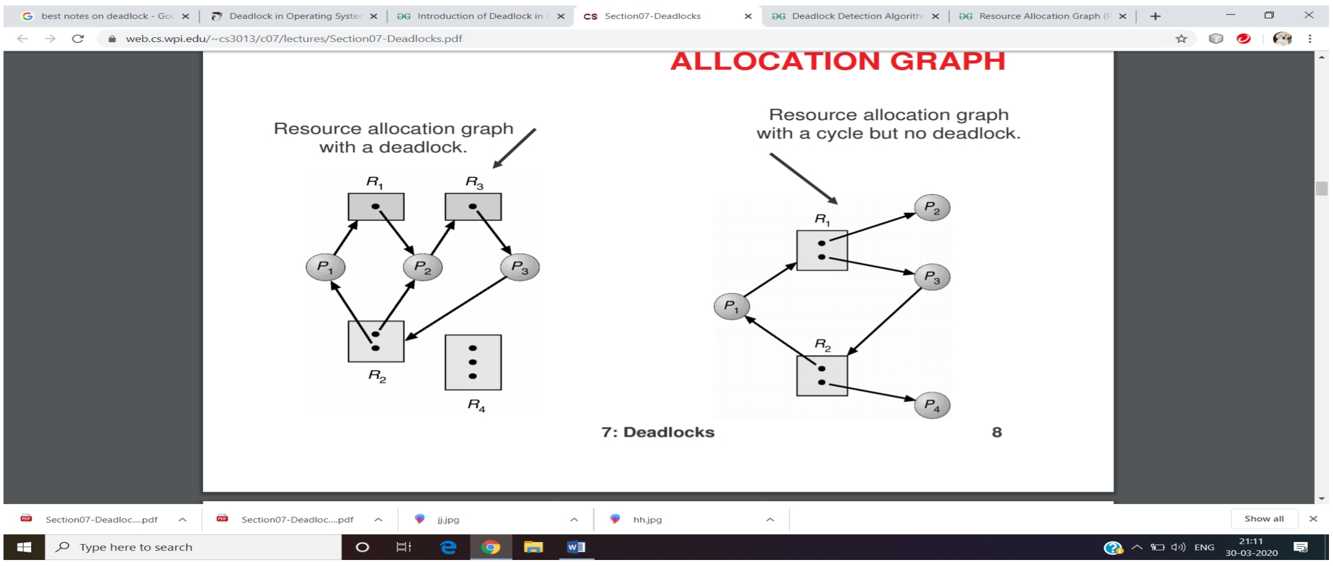
An arrow from the **process** to **resource** indicates the process is **requesting** the resource. An arrow from **resource** to **process** shows an instance of the resource has been **allocated** to the process.

Process is a circle, resource type is square; dots represent number of instances of resource in type. Request points to square, assignment comes from dot.

- If the graph contains no cycles, then no process is deadlocked.
- If there is a cycle, then:
  a) If resource types have multiple instances, then deadlock MAY exist.
  b) If each resource type has 1 instance, then deadlock has occurred.



- If the graph contains no cycles, then no process is deadlocked.
- If there is a cycle, then:
  a) If resource types have multiple instances, then deadlock MAY exist.
  b) If each resource type has 1 instance, then deadlock has occurred.


web.cs.wpi.edu/~cs3013/c07/lectures/Section07-Deadlocks.pdf
ALLOCATION GRAPH
Resource allocation graph with a deadlock.
Resource allocation graph with a cycle but no deadlock.
R1
R2
R3
R4
P1
P2
P3
P4
7: Deadlocks
8

DEADLOCK PREVENTION :

Mutual exclusion:
a) Automatically holds for printers and other non-sharable.
b) Shared entities (read only files) don't need mutual exclusion (and aren’t susceptible to deadlock.) c) Prevention not possible, since some devices are intrinsically non-sharable.
Hold and wait:
a) Collect all resources before execution.
b) A particular resource can only be requested when no others are being held. A sequence of resources is always collected beginning with the same one.
c) Utilization is low, starvation possible.
No preemption:
a) Release any resource already being held if the process can't get an additional resource.
b) Allow preemption - if a needed resource is held by another process, which is also waiting on some resource, steal it. Otherwise wait.
Circular wait:
a) Number resources and only request in ascending order.
b) EACH of these prevention techniques may cause a decrease in utilization and/or resources. For this reason, prevention isn't necessarily the best technique.
c) Prevention is generally the easiest to implement.