OS CLASS ASSIGNMENT

REVISION ON DEADLOCKS

NAME:KOLA LOKESH

REG.NO:19BCS056

**1.**

There are 4 conditions for the deadlock to occur:

1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait
5. **Mutual Exclusion:**

When two people meet in the landings, they can’t just walk through because there is space only for one person. This condition to allow only one person to use the step between them is the first condition necessary for the occurrence of the deadlock.

1. **Hold and Wait:**

When the 2 people refuses to retreat and hold their grounds, it is called holding. This is the next necessary condition for the the deadlock.

1. **No Preemption:**

For resolving the deadlock one can simply cancel one of the processes for other to continue. But Operating System doesn’t do so. It allocates the resources to the processors for as much time needed until the task is completed. Hence, there is no temporary reallocation of the resources. It is third condition for deadlock.

1. **Circular Wait:**

When the two people refuses to retreat and wait for each other to retreat, so that they can complete their task, it is called circular wait. It is the last condition for the deadlock to occur.

Deadlock prevention algorithms ensure that at least one of these necessary conditions does not hold true. However most prevention algorithms have poor resource utilization, and hence result in reduced throughputs

2.

The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.It also contains the information about all the instances of all the resources whether they are available or being used by the processes.In Resource allocation graph, the process is represented by a Circle while the Resource is represented by a rectangle.

The RAG 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.

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.

If a process is using a resource, an arrow is drawn from the resource node to the process node. If a process is requesting a resource, an arrow is drawn from the process node to the resource node.

If there is a cycle in the Resource Allocation Graph and each resource in the cycle provides only one instance, then the processes will be in deadlock. For example, if process P1 holds resource R1, process P2 holds resource R2 and process P1 is waiting for R2 and process P2 is waiting for R1, then process P1 and process P2 will be in deadlock.

Here’s another example, that shows Processes P1 and P2 acquiring resources R1 and R2 while process P3 is waiting to acquire both resources. In this example, there is no deadlock because there is no circular dependency.So, cycle in single-instance resource type is the sufficient condition for deadlock.

3.

**Recovery from deadlock:** process termination

* Abort all dead Cocked process
* Abort one process at a time until the deadlock cycle is eliminated
* The order to chose while aborting is
* priority of process
* How long process has computed , how much longer to complete.
* Resources the process has used
* Resources needed to complete
* How many processes will need to be terminated.
* Is process interactive or batch

**Resource preemption:**

* selecting a victim
* Roll back: return to state state and restart from there
* Stanation : Same process may always be picked as victim, number of roll back in lost factor

4.

**(a).**

The values of Need for processes P1 through P5 respectively are (0, 0, 0, 0), (0, 7, 5, 0), (1,0, 0, 2), (0, 0, 2, 0), and (0, 6, 4, 2).

**(b).**

The sequence is P1, P3, P4, P5, P2. So it is in safe state.

**(c).**

The request from P1(0,4,2,0) available (1,5,2,0) but P1(0,4,2,0) is not less than need. Then it raised error condition, since process have been executed its max claim.

5.

**Deadlock Avoidance:**  
  
• The goal for deadlock avoidance is to the system must not enter an unsafe state.  
  
• Deadlock avoidance is often impossible to implement.  
  
• The system requires additional information regarding the overall potential use of each resource for each process.  
  
• In order for the system to be able to figure out whether the next state will be safe or unsafe, it must know in advance at any time the number and type of all resources in existence, available, and requested.  
  
• Deadlock avoidance techniques include Banker’s algorithm, Wait/Die, Wound/Wait etc.  
  
• Resource allocation strategy for deadlock avoidance selects midway between that of detection and prevention.  
  
• Needs to be manipulated until at least one safe path is found.  
  
• There is no preemption.

**Deadlock Prevention:**

The goal is to ensure that at least one of the necessary conditions for deadlock can never hold.  
  
• Deadlock prevention is often impossible to implement.  
  
• The system doesnot require additional information regarding the overall potential use of each resource for each process.  
  
• In order for the system to prevent the deadlock condition it does not need to know all the details of all resources in existence, available and requested.  
  
• Deadlock prevention techniques include non-blocking synchronization algorithms, serializing tokens, Dijkstra’s algorithm etc.  
  
• Resource allocation strategy for deadlock prevention is conservative, it under commits the resources.  
  
• All resources are requested at once.  
  
• In some cases preempts more than often necessary.