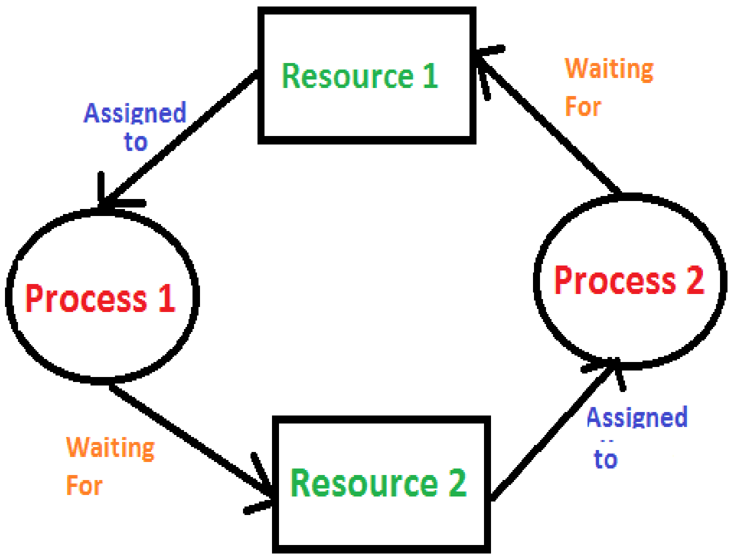
| | |
|---|---|
| **Name** | **Iftekhar Khan** |
| **ID** | **14693** |
| **Assignment** | **Operating System** |
| **Program** | **BS SE** |

1. Explain the necessary conditions that may lead to a deadlock situation. 2. What are the various methods for handling deadlocks?

Answer:

***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.

Question2: Is it possible to have a deadlock involving only one single process? Explain your answer

Answer:

A deadlock situation can only arise if the following four conditions hold simultaneously in a system:

- Mutual Exclusion
- Hold and Wait
- No Preemption
- Circular-wait

It is impossible to have circular-wait when there is only one single-threaded process. There is no second process to form a circle with the first one. One process cannot hold a resource, yet be waiting for another resource that it is holding.

So it is not possible to have a deadlock involving only one process

Question3:  Consider a system consisting of 4 resources of the same type that are shared by 3 processes, each of which needs at most 2 resources. Show that the system is deadlock free.

Answer:

Suppose the system is deadlocked. This implies that each process is holding one resource and is waiting for one more. Since there are three processes and four resources, on process must be able to obtain two resources. This process requires no more resources and therefore it will return its resources when done.

Question 4:

1. What is a resource allocation graph? How do you obtain a wait-for graph from it? Explain their uses.

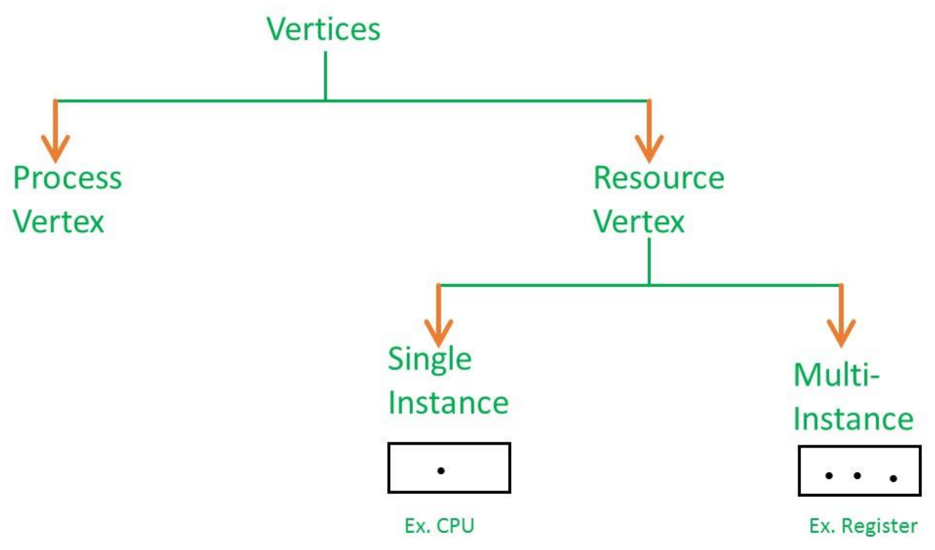
Answer:

## Resource Allocation Graph (RAG) in Operating System

As Banker's algorithm using some kind of table like allocation, request, available all that thing to understand what is the state of the system. Similarly, if you want to understand the state of the system instead of using those table, actually tables are very easy to represent and understand it, but then still you could even represent the same information in the graph. That graph is called **Resource Allocation Graph (RAG)**.

So, resource allocation graph is explained to us what is the state of the system in terms of **processes and resources**. Like 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.



Question 5:

2. Can a system detect that some of its processes are starving? If you answer “yes,” explain how it can. If you answer “no,” explain how the system can deal with the starvation problem.

   Answer:

Detection of starvation requires future knowledge since no amount of record-keeping statistics on processes can determine if it is making 'progress' or not. However, starvation can be prevented by 'aging' a process. This means maintaining a rollback count for each process, and including this as part of the cost factor in the selection process for a victim for preemption/rollback.