CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Bachelor of Engineering (Computer Science & Engineering)

Operating System (CST-328)

Subject Coordinator: Er. Puneet kaur(E6913)

Lecture 11
Deadlock Handling

Deadlock Avoidance

- In a deadlock state, the execution of multiple processes is blocked.
- Deadlock avoidance strategy involves maintaining a set of data.
- The data is used to make a decision whether to entertain any new request or not.
- If entertaining the new request causes the system to move in an unsafe state, then it is discarded.

Resource Allocation Graph (RAG)

- Resource Allocation Graph (RAG) is a graph that represents the state of a system pictorically.

Resource-Allocation Graph

- It gives complete information about the state of a system such as-

- How many processes exist in the system?
- How many instances of each resource type exist?
- How many instances of each resource type are allocated?
- How many instances of each resource type are still available?
- How many instances of each resource type are held by each process?
- How many instances of each resource type does each process need for execution?

Resource-Allocation Graph

In some cases deadlocks can be understood more clearly through the use of **Resource-Allocation Graphs**, having the following properties:

- A set of resource categories, { R1, R2, R3, . . ., RN }, which appear as square nodes on the graph. Dots inside the resource nodes indicate specific instances of the resource. (E.g. two dots might represent two laser printers.)

- A set of processes, { P1, P2, P3, . . ., PN }

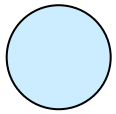
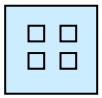
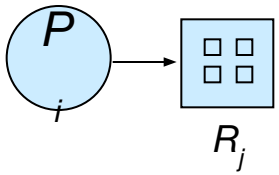
Resource-Allocation Graph

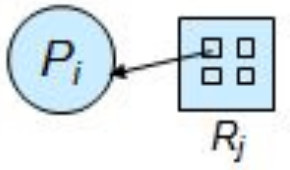
- **Request Edges -** A set of directed arcs from Pi to Rj, indicating that process Pi has requested Rj, and is currently waiting for that resource to become available.

- **Assignment Edges -** A set of directed arcs from Rj to Pi indicating that resource Rj has been allocated to process Pi, and that Pi is currently holding resource Rj.

- Note that a **request edge** can be converted into an **assignment edge** by reversing the direction of the arc when the request is granted. (However note also that request edges point to the category box, whereas assignment edges emanate from a particular instance dot within the box.)

Resource-Allocation Graph (Cont.)

- Process 
- Resource Type with 4 instances 
- P_i requests instance of R_j 

- P_i is holding an instance of R_j 

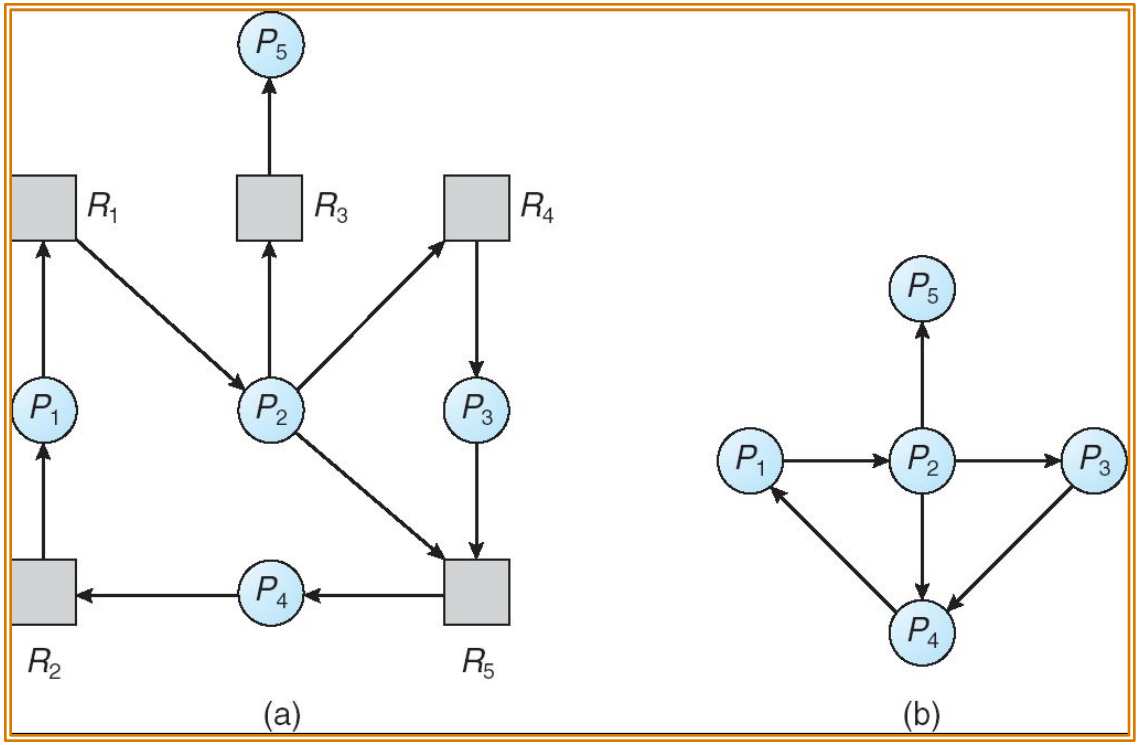
Basic Facts

- If graph contains no cycles ⇒ no deadlock
- If graph contains a cycle ⇒
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

Single Instance of Each Resource Type

- Maintain *wait-for* graph
  - Nodes are processes.
  - $P_i \rightarrow P_j$ if P_i is waiting for P_j.
- Periodically invoke an algorithm that searches for a cycle in the graph.
- An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph.

Resource-Allocation Graph and Wait-for Graph



Resource-Allocation Graph Corresponding wait-for graph

Deadlock Detection

- Using Resource Allocation Graph, it can be easily detected whether system is in a Deadlock state or not. The rules are-

Rule-01:

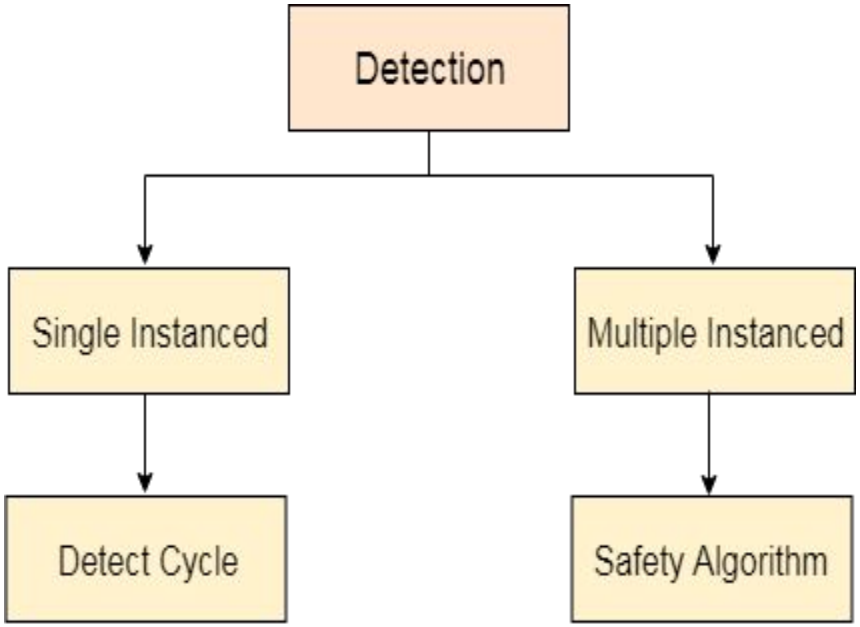
- In a Resource Allocation Graph where all the resources are single instance,
  - If a cycle is being formed, then system is in a deadlock state.
  - If no cycle is being formed, then system is not in a deadlock state.

Rule-02:

- In a Resource Allocation Graph where all the resources are **NOT** single instance,
- If a cycle is being formed, then system may be in a deadlock state.
- **Banker's Algorithm** is applied to confirm whether system is in a deadlock state or not.
- If no cycle is being formed, then system is not in a deadlock state.
- Presence of a cycle is a necessary but not a sufficient condition for the occurrence of deadlock.

Deadlock Detection and Recovery

- This strategy involves waiting until a deadlock occurs.
- After deadlock occurs, the system state is recovered.
- The main challenge with this approach is detecting the deadlock.

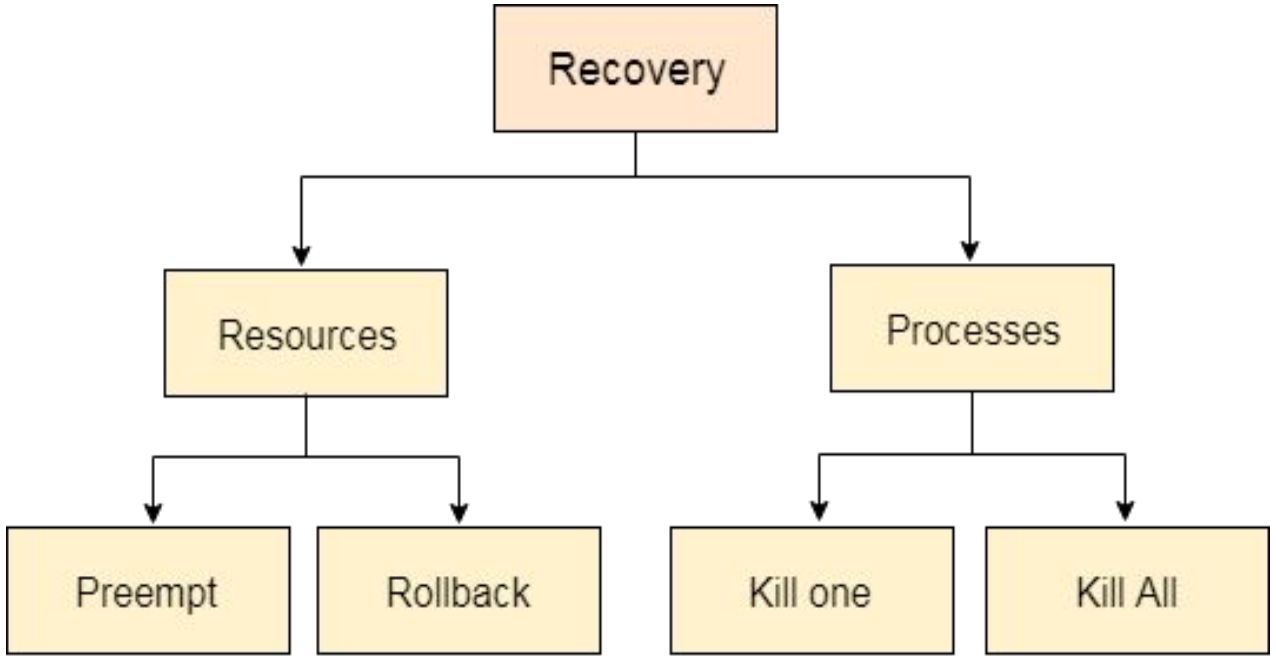


Deadlock Detection and Recovery

- The OS can detect the deadlocks with the help of Resource allocation graph.
- In single instanced resource types, if a cycle is being formed in the system then there will definitely be a deadlock.
- On the other hand, in multiple instanced resource type graph, detecting a cycle is not just enough.
- We have to apply the safety algorithm on the system by converting the resource allocation graph into the allocation matrix and request matrix.

Deadlock Recovery

In order to recover the system from deadlocks, either OS considers resources or processes.



Deadlock Recovery

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.

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.

Deadlock Ignorance

- This strategy involves ignoring the concept of deadlock and assuming as if it does not exist.
- This strategy helps to avoid the extra overhead of handling deadlock.
- Windows and Linux use this strategy and it is the most widely used method.
- It is also called as **Ostrich approach**.

Conclusion

This lecture give you a deep insight to Deadlock avoidance-safe state, resource allocation graph , Deadlock detection, Recovery from deadlock etc.

References

https://www.includehelp.com/c-programming-questions/

https://www.studytonight.com/operating-system/

https://computing.llnl.gov/tutorials/

https://www.tutorialspoint.com/operating_system/index.htm#:~:text=An%20operating%20system%20(OS)%20is,software%20in%20a%20computer%20system.

https://www.javatpoint.com/os-tutorial

https://www.guru99.com/operating-system-tutorial.html

https://www.geeksforgeeks.org/operating-systems/