International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 5, Sep-Oct 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Review on Deadlock Handling

Shruti Kedare*, Shruti Shengule**, Ruchi Kamble***, Sonali Ingole****
*(Computer Engineering, P.E.S College of Polytechnic, and Chh. Sambhaji Nagar
Email: shrutikedare882@gmail.com)
**(Computer Engineering, P.E.S College of Polytechnic, and Chh. Sambhaji Nagar
Email: shrutishengule@gmail.com)
***(Computer Engineering, P.E.S College of Polytechnic, and Chh. Sambhaji Nagar
Email: ruchi.kamble2007@gmail.com)
****(Computer Engineering, P.E.S College of Polytechnic, and Chh. Sambhaji Nagar
Email: sonaliingole2007@gmail.com)

----------**********----------

Abstract:

In operating system one of the most important task is sharing and allocating resources in the process which are new and running. In multiprocessing environment, resource allocation is can become a point of conflict and this conflict is called deadlock. Deadlock detection is a very important, in this paper overlooked problem that can significantly overall deadlock handling performance affect. In operating system Deadlock is consider as critical problem where each process is waiting for resources which is held by other process, because of these multiple processes are stuck forever. Because of deadlock, performance of system is reducing and if this problem is not handled it can cause to complete system freeze. This paper shows deadlock concept, the condition in which deadlock occurs, and different types of approach like, prevention, avoidance, detection and recovery. The purpose is to explain how operating system manage deadlock and what future methods can improve system reliability

Keywords **— Operating System; Deadlock, Resource Allocation, Multiprocessing Environment, System Freeze.**

----------**********----------

I. INTRODUCTION

Operating system is a program that acts as an interface between user and the computer and manage all computer resources. In today's computing environment, modern operating system allows multiprocessing in which multiple processes are run at the same time, and these processes require shared resources such as files, memory, printers, CPU for completing their task. Since, system have limited shared resources so, allocating and sharing resources to the task to be completed according to their requirement or need in proper manner is very necessary. When more than one processes are unable to proceed their task because one process require resource held by another process and is not ready to release resource it means that deadlock is occurred.

Deadlock is matters in operating system and if we ignore that instead of handling it may slow down the system or crash which is harmful for our system. It is lead to many problems like one car cannot move forward, need to wait for reach destination, if try to move forward an accident may happen because it causes underutilization of CPU.

II. DEADLOCK CONDITIONS

Deadlock does not occur randomly in the system for deadlock occur there are 4 conditions identified by coffman in 1971 and these conditions must present simultaneously.

- **There are four conditions of deadlock:**

1. Mutual Exclusion.
2. Hold and Wait.

3. No Preemption.
4. Circular Wait.

1. Mutual Exclusion

In this condition at least one resource is must hold in non-sharable mode by the process hence only one process can hold this resource for its execution if another process have wish to use same resource and makes request for this resource then it is impossible to share the resource because this resource is in the non- sharable mode. **Example:** two friends (process) want to attend party only one of them (process) has party dress(resource) so she cannot lend to another (process).

2. Hold and Wait

This condition state that, at least one resource is hold by a process and simultaneously waits for one more additional resource that is already held by another process this condition completes its execution but also create problem **Example:** There are two process p1 and p2, p1 holds resource r1 and request to resource r2 while resource r2 is hold by p2 Waiting for resource r1 which is hold by p1. Combination of these process and resources leads to chances of occur deadlock and begin to circular waiting chain.

3. No Preemption

This condition state that once the resource is allocated to the one process other process cannot forcibly taken away that resource until execution is complete of current process even if that process high priority. So Other process have to wait until execution of current process completes because once resource the resource allocated to the process, process keeps the resource until it releases resource either by terminating or by switching to waiting state So, this condition also contributes chances of deadlock. In short resource can only be released by choice of process.

4. Circular Wait

In this condition each process is waiting for resource held by next process it makes circular chain, where chain of two or more process there is existence of circular chain thus each of these processes cannot complete its execution. **Example:** Process p1 is waiting for the resource hold by the process p2 and process p2 is also waiting for the resource held by process p3.

When each of above conditions are true then deadlock occurs

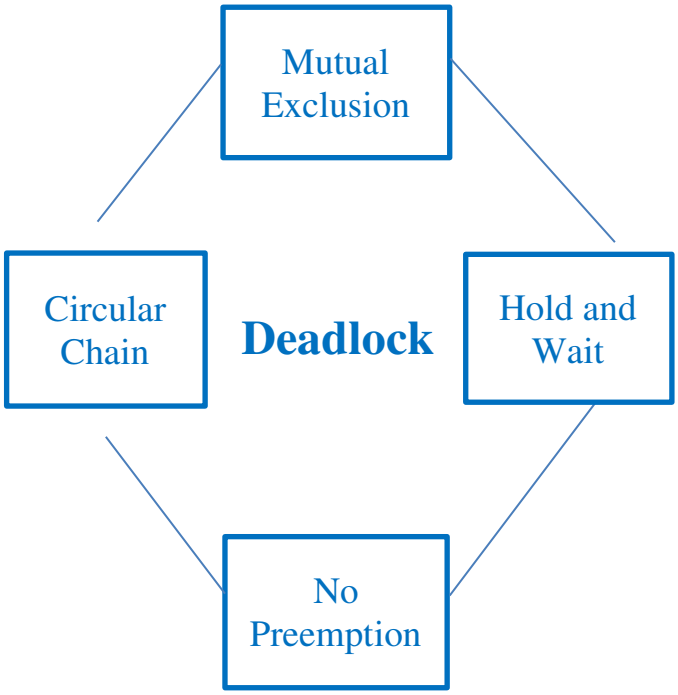

Fig: Conditions of Deadlock.

III. DEADLOCK PREVENTION

To protect the system from deadlock in operating system the method is used called as deadlock prevention it is useful to ensure that from four condition of deadlock at least one condition never occurs if prevent at least one condition, the system has no chance to deadlock will occur. This idea protects the system from deadlock before Deadlock occurs instead of after deadlock happen.

- ***There are four methods of deadlock prevention:***

1. Eliminate Mutual Exclusion.
2. Eliminate Hold and Wait.
3. Eliminate No Preemption.
4. Eliminate Circular Chain.

1. Eliminate Mutual Exclusion

Since in mutual exclusion condition, resource is hold by the process in non-shareable mode now we Try to share resource in shareable mode. **Example:**

Multiple processes can share resources such as read only file and code segment simultaneously. But there is no guarantee of all-time resources like printer or I/O device are in shareable mode. So, Eliminate mutual exclusion not always possible.

2. Eliminate Hold and Wait

In this method, process prevent by a process cannot hold resource while waiting for another resource.

- ***This method has two sub methods:***

A. All-or-Nothing Allocation.
B. Release Before Request.

A. All-or-Nothing Allocation: Before starting execution of process, it must request all the resource at once it requires.

B. Release Before Request: A process before making new request for resource. It is must to release current all resources. But, this method of prevention also leads to drawback of low resource utilization and possible starvation (indefinitely waiting for resource).

3. Eliminate No Preemption

No Preemption means once the resource is allocated to a process it cannot forcibly remove from that process till this process release that resource. To eliminate this condition, the operating system says use preemption means resource allocated to the process can forcibly remove during execution of process for current executing process when another process has high priority or other scenario of execution. If a process holds one resource and also waits for another resource held by another process, can preempt means forcibly take away and gives to current process for its execution so no preemption is eliminated. **Example:** There are two processes P1 and P2 and two resources R1 and R2.
When P1 holds R1 and P2 holds R2, P1 has need of R2 then P1 forcibly take away R2 from P2 by stopping execution of P2. After complete execution of P1 then it gives R2 to P2 for remaining execution

4. Eliminate Circular Wait

In a circular wait deadlock, each process waiting for a resource which is hold by another process in a circular chain. **Example:**
p1 -- wait for resources R2 (that are hold by p2)
p2 -- wait for resources R3(that are hold by p3)
p3--wait for resources R1(that are hold by p1).
That cause a deadlock loop where no one came proceed. Let's we see how to prevent deadlock in eliminating circular wait. To release of circular wait we follow a strict order on resources allocation.
Rule: All resources must be requested in a predefined fixed order. That's means every resource has a unique number and processes can only request in forward order of that numbering.

IV. DEADLOCK AVOIDANCE

Deadlock prevention is a active approach method implemented in operating system to make sure that system prevent the unsafe condition that might create a deadlock. In contrast to deadlock prevention, which not allow the managing resources to avoid the risk of deadlock. Deadlock avoidance enables better process execution by deciding based on current resource use and expected future needs. In this method, the system wants more knowledge of each process peak resources are required. Before giving a resource request, the system check whether giving the request, will maintain the system in a safe place. A save condition is one in which all process can fulfill their completion without causing a deadlock. If the process remains save, resources is provided, otherwise, the process must wait. A more used algorithm for deadlock avoidance is the Bankers Algorithm, suggested by Dijkstra. This test the allocation of resources for every potential request and make sure that system stay safe before giving the request. By using deadlock prevention method, operating system get harmony between resources utilization and system safety, stopping deadlocks while sustaining effective task performance.

The banker's Algorithm is a deadlock avoidance algorithm which is developed by Edsger Dijkastra.it is used for suppose a system is in a safe state or not by simulating the allocation of resources to processes. It situation that the system never enters in

an unsafe state that could lead to a deadlock. Banker's algorithm works like banker giving loans: Let's see the banker (operating system) will only grant a loan (resource) if it knows it can still satisfy all other stakeholders (processes) in the future.
Bankers denies request temporarily when if giving the resource makes it impossible to fulfill other's maximum needs.

Important terms
1) Available
2) Max
3) Allocation
4) Need

Algorithm steps :
1.Safety algorithm.
2.Resources Request Algorithm.
3.pretend to allocates resources temporarily.
4)Run the safety algorithm

V. DEADLOCK RECOVERY

Deadlock Recovery is nothing but the fixing the situation means when the deadlock occurs in the system.

There is some way to do this:

1. Priority method.
2. Checkpoint and Rollback Method.
3. Progress Termination.

1. Priority method:

One simple way is to give importance to some processes first. The system can give another process to release a resource so that the waiting process can run its work, if there is any process in stuck. In some cases, a problem happens again when the same process that release and resource give it back later and run as nothing happened at all.

2. Checkpoint and Rollback Method:

In this method, there is a system that takes snapshots checkpoint time to time of its state. If a deadlock occurs, the system returns previous snapshots and restart from there before the deadlock. It helps to make the locked resource free.
At the same time, some progress create after the last checkpoint will gone. It will again have to wait, if there is a low priority process ask for any resource.

3. Progress Termination:

Another way to stop recovery from deadlock is involving one or more processes. It is very effective method. The system needs to decide that which process to stop because it depends on that what loss will be happen if it is stopped. Usually, there is no rules fixed this but something the process that can be reopened effortlessly and don't have any important work is selected for termination.

VI. DEADLOCK SIDE-EFFECTS

Undesirable system condition in a same group are also starvation and live lock. Both are considered as derivatives of deadlock. Restricted the process, it has the common attributes involved from making progress. When some scheduling algorithm determines the acquisition and utilization of resources, starvation happens but there are some processes that even though are not deadlock and never acquired required resources so for that reason making no progress. Live lock or busy-waiting describe in a state that where a process is 'spinning' forever waiting for a condition that not satisfied, again blocking progress. Thus, the efforts to avoid deadlock and starvation condition, the equity of nation threatened. Impartiality indicates the process that something has to executed and complete the task. To accomplish this, the existence of arbitrary planning scheduling ensure that all processes executed within an acceptable time frame. The solution to justice related problems is addressing normal with the process prioritization handled by the operating system with a specific algorithm.

CONCLUSION

Deadlock is a serious issue in operating systems where processes compete for limited resources. It can halt system operations if not properly managed. This paper explained the conditions for deadlock and outlined key strategies for handling it—prevention, avoidance, detection, and recovery. Effective deadlock management ensures better resource utilization, system performance, and overall reliability.

ACKNOWLEDGE
I really thankful and grateful to my Prof. Nisha Pawar mam for continuous support and guidance and I also thankful to my institute P.E.S of polytechnic for providing resources and support in completing this work.

REFERENCE
[1] Coffman E. G., Elphick M. J., Shoshani A. 1971. System Deadlocks. ACM Computing Surveys. 3(2): 67–78.
https://dl.acm.org/doi/10.1145/356586.356588
[2] Dijkstra E. W. 1965. Cooperating Sequential Processes. Technical Report EWD-123, Technological University Eindhoven.
https://www.cs.utexas.edu/~EWD/transcriptions/EWD01xx/EWD123-2.html
[3] Silberschatz A., Galvin P. B., Gagne G. 2018. Operating System Concepts (10th ed.). Wiley.
https://www.wiley.com/enus/Operating%2BSystem%2BConcepts%2C%2B10th%2BEdition-p-9781119320913

[4] Dimitoglou G. 1998. Deadlocks and Methods for their Detection, Prevention and Recovery in Modern Operating Systems. ResearchGate. Available: https://www.researchgate.net/publication/220623929
[5] Mohanty M., Kumara P. 2013. Deadlock Prevention in Process Control Computer System. International Journal of Computer Applications. ICDCIT Special Issue.
https://ijcaonline.org/proceedings/icdcit/number1/10236-1003/

