UNDERSTANDING DEADLOCK IN OPERATING SYSTEMS

**A C O M P R E H E N S I V E R E P O R T**

P r e p a r e d b y :

SHAIK SAMEER - R A 2 4 1 1 0 3 0 0 1 0 2 2 7

V.CHARAN RAJU - R A 2 4 1 1 0 3 0 0 1 0 2 3 2

G.PAVAN SARVAN - R A 2 4 1 1 0 3 0 0 1 0 2 5 1

D E PA R M E NT O F N E T W O R K I N G C O M M U N I C AT I O N S

INTRODUCTION:

Deadlock occurs in an operating system when a set of processes become permanently blocked, each waiting for resources held by others.

This halts system progress and can cause freezing or crashes.

Impact:

 Causes system paralysis or freezing  Reduces performance

 May lead to crashes or data loss

Necessary Conditions for Deadlock (Coffman Conditions)

1. Mutual Exclusion

 At least one resource is non-sharable.

 Example: A printer can’t be shared by two processes at once.

1. Hold and Wait

 A process holds one resource while waiting for another.

 Example: Process A holds a printer and waits for a scanner.

1. No Preemption

 Resources can’t be forcibly taken from a process.

 Example: A process must finish printing before releasing the printer.

1. Circular Wait

 A circular chain of processes exists, each waiting for a resource held by the next.

 Example: P1 → waits for R2, P2 → waits for R3, P3 → waits for R1.

# HANDLING METHODS:

Deadlock prevention ensures that at least one of the four necessary conditions can never hold.

Methods of Prevention:

 Eliminating Hold and Wait:

 Require processes to request all resources at once or release held ones before requesting new resources.

 Eliminating No Preemption:

 Allow the system to take resources away from a process if necessary.

 Eliminating Circular Wait:

a. Impose an order on resource types and force processes to request them in a specific sequence.

# DEADLOCK AVOIDANCE

Deadlock avoidance means making careful decisions about resource allocation to ensure the system always remains in a safe state.

The most common approach is the Banker’s Algorithm, proposed by Dijkstra.

It requires prior knowledge of the maximum number of resources each process might request.

The system then checks whether granting a resource keeps it in a safe state before approving the request

.

Advantages:

 Provides better resource utilization  Ensures system safety

Disadvantages:

 Requires advance information about process needs

 Adds extra computational overhead

DEADLOCK WESITE LINK : https://docs.google.com/document/d/1eMM--HghtfHIgkm4DxawlzNMRkmE4d8yP4bOkobc7Ns/edit?usp=drivesdk

# Real-World Examples

 Database Systems: Transactions waiting for each other’s locked records.

 Multithreaded Programs: Threads each holding a lock and waiting for another thread’s lock.

 Distributed Systems: Processes on different machines waiting for remote resources.

 Device Management: Two programs each need a printer and scanner but acquire them in opposite order.

# CONCLUSION

 Deadlock is a core issue in concurrent and parallel systems that can completely stop progress.

 Understanding the four necessary conditions is essential for identifying and preventing deadlocks.

 Prevention, avoidance, and detection with recovery are the main handling techniques.

 In real practice, many operating systems rely on developers to design deadlock-free code, a strategy often called the “Ignorance Approach.”