OPERATING SYSTEM

Deadlock

Definition

A deadlock is a situation in an operating system where two or more processes are permanently blocked because each process is waiting for a resource held by another process.

As a result, no process can proceed, and the system comes to a halt for those processes.

Example of Deadlock

- Process **P1** holds Resource **R1** and waits for **R2**
- Process **P2** holds Resource **R2** and waits for **R1**

Neither process can continue → **Deadlock occurs**

P1 → R2 (held by P2)

P2 → R1 (held by P1)

Necessary Conditions for Deadlock

Deadlock occurs only if all four conditions hold simultaneously:

1. Mutual Exclusion

- At least one resource must be **non-shareable**.
- Only one process can use the resource at a time.

2. Hold and Wait

- A process is **holding at least one resource** while waiting for additional resources.

3. No Preemption

- Resources **cannot be forcibly taken** from a process.
- They must be released voluntarily.

4. Circular Wait

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

Methods for Handling Deadlock

1. Deadlock Prevention

Idea

Ensure that at least one deadlock condition never occurs.

Examples

- Eliminate hold and wait
- Allow resource preemption
- Impose resource ordering

2. Deadlock Avoidance

Idea

Avoid deadlock by careful resource allocation.

Example

- Banker's Algorithm
- Requires advance knowledge of resource needs

3. Deadlock Detection and Recovery

Idea

- Allow deadlock to occur
- **Detect** it and then **recover**

Recovery Methods

- Terminate processes
- Preempt resources

4. Ignore Deadlock (Ostrich Approach)

Idea

- Assume deadlock is rare
- Take no action

Used In

- Most general-purpose operating systems

Banker's Algorithm

Definition

Banker's Algorithm is a **deadlock avoidance algorithm** used by an operating system to decide whether a resource request can be safely granted without leading the system into a **deadlock state**.

It works by ensuring the system always remains in a **safe state**.

Why is it Called Banker's Algorithm?

It is named after the banking system, where:

- A bank never gives out all its money at once
- It ensures it can satisfy the maximum possible needs of all customers

Similarly, the OS checks whether it can satisfy the **maximum resource needs** of all processes.

Basic Concepts Used

For n processes and m resource types:

1. Available

- Number of available instances of each resource.

2. Max

- Maximum demand of each process.

3. Allocation

- Resources currently allocated to each process.

4. Need

Remaining resource requirement.

Need = Max - Allocation

Safe State and Safe Sequence

Safe State

A system is in a **safe state** if there exists at least one order (sequence) of process execution such that **all processes can complete** without deadlock.

Safe Sequence

An order of processes where each process can obtain its needed resources and finish.

Steps of Banker's Algorithm

1. Check if the requested resources are **less than or equal to Need**
2. Check if requested resources are **less than or equal to Available**
3. Temporarily allocate resources
4. Check if the system is still in a **safe state**
5. If safe → Grant request
   If unsafe → Deny request

Advantages

- Prevents deadlock
- Ensures system safety

Disadvantages

- Requires advance knowledge of maximum resource needs
- High overhead
- Not practical for large systems