PRACTICAL NO: - 11

OBJECTIVE :- simulate Banker algorithm for deadlock Prevention.

- **Introduction to Deadlocks**

In computer science, a deadlock refers to a situation in which two or more processes are unable to proceed because each is waiting for the other to release a resource. Deadlocks are a critical issue in multitasking operating systems, as they can lead to system-wide halts, impacting performance and user experience.

- **Understanding Resource Allocation**

To understand the Banker's algorithm, it's essential to grasp the concept of resource allocation. In an operating system, processes require various resources to execute tasks, such as CPU time, memory, and I/O devices. These resources are often limited and must be managed efficiently to prevent deadlocks.

- **The Need for Deadlock Avoidance**

One approach to dealing with deadlocks is deadlock avoidance, which aims to prevent deadlocks from occurring in the first place. The Banker's algorithm is a prime example of a deadlock avoidance technique used in operating systems.

- **Principles of the Banker's Algorithm**

The Banker's algorithm operates on the principle of resource allocation with safety. It considers the current allocation of resources, the maximum resources each process may need to complete, and the available resources in the system.

- **Data Structures Used**

The Banker's algorithm employs several data structures to manage resource allocation:

- Available: An array representing the available instances of each resource type.
- Max: A matrix indicating the maximum demand of each process for each resource type.
- Allocation: A matrix showing the resources currently allocated to each process.
- Need: A matrix indicating the remaining resources needed by each process to complete its tasks.
- **Initialization**

Before executing the Banker's algorithm, the system initializes the data structures mentioned above. It determines the available resources and the maximum resource requirements of each process. Then, it calculates the remaining resource needs based on the allocated resources.

- **Safety Algorithm**

The core of the Banker's algorithm is the safety algorithm, which determines whether a system is in a safe state before allocating additional resources. The safety algorithm simulates resource allocation to ensure that no deadlock will occur.

- **Resource Request Handling**

When a process requests additional resources, the Banker's algorithm checks whether granting the request would lead to a safe state. It verifies that the requested resources are available and that allocating them won't exceed the maximum resource requirements of any process.

- **Resource Release**

Once a process completes its tasks, it releases the allocated resources. The Banker's algorithm updates the available resources and reallocates them to other processes if needed.

- **Deadlock Detection**

While the Banker's algorithm focuses on deadlock avoidance, some systems may also incorporate deadlock detection mechanisms. These mechanisms periodically check for deadlock conditions and take corrective actions if necessary.

Example Scenario

Let's illustrate the Banker's algorithm with a simple scenario involving three processes (P1, P2, P3) and three resource types (R1, R2, R3).

1. Initialization:

- Available resources: (10, 5, 7)
- Max resource needs:
  - P1: (7, 5, 3)
  - P2: (3, 2, 2)
  - P3: (9, 0, 2)
- Allocated resources:
  - P1: (0, 1, 0)
  - P2: (2, 0, 0)
  - P3: (3, 0, 2)
- Need matrix:
  - P1: (7, 4, 3)
  - P2: (1, 2, 2)
  - P3: (6, 0, 0)

2. Safety Algorithm:

- Initial state: Work = Available = (10, 5, 7), Finish = (false, false, false)
- Iteration 1: P2 can proceed (Work >= Need[P2]), so allocate its resources and mark it as finished.
- Iteration 2: P3 can proceed, allocate its resources, and mark it as finished.
- Iteration 3: P1 can proceed, allocate its resources, and mark it as finished.
- All processes finished, system in a safe state.

3. Resource Request Handling:

- Suppose P1 requests (1, 0, 2) additional resources.
- Check if Need[P1] >= Request and Available >= Request. If true, simulate allocation and run safety algorithm.
- If the system remains in a safe state, grant the request; otherwise, deny it.

4. Resource Release:

- When a process completes, release its allocated resources and update the available resources accordingly.
- **Conclusion**

The Banker's algorithm is a crucial deadlock avoidance technique used in operating systems to manage resource allocation efficiently. By carefully analyzing resource requests and system states, it ensures that processes can execute without encountering deadlocks. Understanding and implementing the Banker's algorithm is essential for developing robust and reliable operating systems.