Deadlocks: Characterization

Deadlocks occur in concurrent systems when two or more processes or threads are indefinitely blocked, each waiting for the other to release a resource, which they themselves hold. The four necessary conditions for deadlock are:

Mutual Exclusion: Resources cannot be shared; only one process can use a resource at a time.

Hold and Wait: Processes hold resources while waiting for others. A process can hold allocated resources while requesting additional ones.

No Preemption: Resources cannot be forcibly taken from a process; they must be released voluntarily.

Circular Wait: There exists a circular chain of two or more processes, each waiting for a resource held by the next one.

Deadlocks: Prevention

Deadlock prevention aims to eliminate one of the necessary conditions for deadlock. Techniques include:

Mutual Exclusion Elimination: Allow sharing of resources that can be safely shared.

Hold and Wait Elimination: Require processes to request all necessary resources at once, or release held resources and restart if all cannot be obtained.

No Preemption: Introduce preemption, where lower-priority processes can have their resources forcefully removed if necessary.

Circular Wait Elimination: Impose a total ordering on resource types and require processes to request resources in increasing order.

Deadlocks: Avoidance

Deadlock avoidance involves dynamically ensuring that the system remains in a safe state by carefully scheduling resource allocations. Techniques include:

Resource Allocation Graph: Use a graph-based algorithm to dynamically check for the absence of cycles in the resource allocation graph.

Banker's Algorithm: Analyze the resource needs of each process and only grant requests if doing so will not lead to an unsafe state.

Deadlocks: Detection

Deadlock detection involves periodically checking the system's state to determine whether a deadlock has occurred. Techniques include:

Resource Allocation Graph Algorithm: Use graph algorithms to detect cycles in the resource allocation graph, indicating the presence of deadlock.

Deadlock Detection and Recovery: Employ algorithms that periodically inspect the system to detect deadlocks and then take corrective action, such as aborting processes or rolling back transactions.

Deadlocks: Recovery

Deadlock recovery involves resolving deadlocks once they have been detected. Techniques include:

Process Termination: Abort one or more processes involved in the deadlock to break the circular wait.

Resource Preemption: Forcefully remove resources from one or more processes to break the deadlock.

Rollback: Roll back the progress of one or more processes to a safe state before the deadlock occurs.

Restart: Restart the entire system or affected processes from a known safe state.

Each of these techniques has its advantages and disadvantages, and the choice depends on factors such as system requirements, performance considerations, and the severity of deadlock occurrences.