Deadlock and starvation

Deadlock

Definition:

Deadlock is a condition in which a set of processes are blocked, each waiting for a resource held by another process in the set. This creates a cycle of dependencies that prevent any of the processes from continuing.

Conditions for Deadlock (Deadlock Coffins):

1. **Mutual Exclusion**: At least one resource must be held in a non-shareable mode. If a resource is allocated to a process, no other process can use it until it is released.

2. **Hold and Wait**: Processes holding resources can request additional resources without releasing the resources they currently hold.

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

4. **Circular Wait**: A set of processes exists such that each process is waiting for a resource held by the next process in the set, forming a circular chain.

Example:

Consider two processes, P1 and P2, and two resources, R1 and R2:

- P1 holds R1 and requests R2.

- P2 holds R2 and requests R1.

Both processes end up waiting indefinitely for each other's resources, resulting in a deadlock.

Prevention Techniques:

1. **Eliminate Mutual Exclusion: Make resources sharable, if possible.**

2. **Eliminate Hold and Wait: Require processes to request all resources at once or release all resources before requesting new ones.**

3. **Eliminate No Preemption: Allow resources to be preempted from processes if necessary.**

4. **Eliminate Circular Wait: Impose a total ordering of resources and require processes to request resources in a specific order.**

****Detection and Recovery:****

- **Detection: Use resource allocation graphs or other algorithms to detect deadlock.**

- **Recovery: Terminate processes or preempt resources to break the deadlock cycle.**

Starvation

****Definition:****

Starvation occurs when a process is perpetually denied necessary resources because other processes are continually given priority. This results in the affected process never getting a chance to proceed.

****Causes:****

1. **Priority Scheduling: High-priority processes might always be given resources before low-priority processes, causing the latter to starve.**

2. **Resource Allocation Policies: Certain policies might favor some processes over others, leading to long-term deprivation of resources for some processes.**

****Example:****

In a system using priority scheduling:

- Process P1 has high priority and constantly preempts Process P2.

- Process P2 may never get CPU time if P1 is always arriving or requiring resources, leading to starvation of P2.

****Prevention Techniques:****

1. **Aging: Gradually increase the priority of processes that have been waiting for a long time to ensure they eventually get resources.**

2. **Fair Scheduling: Use algorithms like round-robin or fair-share scheduling that ensure all processes get a chance to access resources.**

****Detection and Recovery:****

- **Detection: Monitor processes to detect those that have been waiting excessively.**

- **Recovery: Adjust priorities or scheduling policies to ensure all processes are eventually serviced.**