4.4 Common Problems in Synchronization

1. **Deadlock**
   A deadlock is a situation where two or more tasks are blocked forever, each waiting for a resource that is held by another task in the cycle.

   Example Deadlock Scenario:
   Task A successfully locks Mutex 1. Task B successfully locks Mutex 2. Task A now tries to lock Mutex 2, but it must wait because Mutex 2 is held by Task B. Task B now tries to lock Mutex 1, but it must wait because Mutex 1 is held by Task A. Both tasks are now waiting for each other indefinitely, and neither can proceed.

   The Four Conditions for Deadlock **(Coffman's Conditions)**:
   1. **Mutual Exclusion**: At least one resource must be non-sharable (can only be used by one task at a time).
   2. **Hold and Wait**: A task holds at least one resource while waiting for another resource held by a different task.
   3. **No Preemption**: A resource cannot be forcibly taken from the task holding it; it can only be released voluntarily.
   4. **Circular Wait**: A circular chain of two or more tasks exists, where each task is waiting for a resource held by the next task in the chain.

2. **Priority Inversion**
   Priority inversion is a scenario where a high-priority task is forced to wait for a much lower-priority task. This happens when the low-priority task is holding a lock (e.g., a mutex) that the high-priority task needs. The problem worsens if a medium-priority task starts running, as it will preempt the low-priority task, preventing it from releasing the lock. As a result, the high-priority task never gets a chance to run.

3. **Starvation**
   Starvation is a condition where a task is perpetually denied access to the resources it needs to complete its execution. This often happens to low-priority tasks that are constantly being preempted by higher-priority tasks, preventing them from ever getting their slice of CPU time.

Revision #3
Created 2025-08-28 16:12:52 UTC by NS
Updated 2025-08-28 16:21:24 UTC by NS