Report - Operating Systems

Constant Nortey Jr.

December 4, 2024

1 Introduction

Deadlock occurs in a system when processes are unable to proceed because they are each waiting for resources that other processes hold. For a deadlock to arise, the following four conditions must simultaneously be true:

2 Conditions Necessary for Deadlock

2.1 Mutual Exclusion

With this condition, at least one resource must be held in a non-shareable mode. In other words, only one process at a time can use the resource. At least one resource must be held in a non-shareable mode. In other words, only one process at a time can use the resource. Resources such as printers, files, or database records are often mutually exclusive. For example, a printer cannot simultaneously print two different jobs. Risk: Mutual exclusion is essential for many resources, but it inherently limits simultaneous access, making deadlocks possible if other conditions are met.

2.2 Hold and Wait

A process holding one or more resources is waiting to acquire additional resources that are currently being held by other processes. Imagine a process holding a file lock and waiting for access to a printer, while another process holds the printer and waits for the file lock. Risk: This condition creates a dependency chain among processes, leading to potential deadlock when resources are scarce.

2.3 No Preemption

Resources cannot be forcibly taken from a process holding them. A process must release its held resources voluntarily. If a process does not release its held resources until it completes its task, no other process can use those resources in the meantime. Risk: Without preemption, a process that encounters an error or becomes non-responsive could block other processes indefinitely, increasing the likelihood of deadlock.

2.4 Circular Wait

A set of processes exists such that each process is waiting for a resource held by the next process in the chain, forming a circular dependency. For instance, process P1 holds resource R1 and waits for R2, while process P2 holds R2 and waits for R3, and so on, with the last process waiting for R1. Risk: Circular wait is the final condition that locks the system in a deadlock. It creates a closed loop where no process can make progress.

2.4.1 Understanding the Risks

- Deadlock conditions often arise in systems with shared resources and multiple processes.
- These conditions can severely disrupt process execution by freezing progress and wasting system resources.

- In real-world systems, operating systems and resource management strategies aim to prevent or mitigate these conditions through techniques like resource allocation graphs, timeouts, and deadlock avoidance algorithms (e.g., Banker's algorithm).

Understanding these conditions is essential for designing robust systems that handle resource contention efficiently.