Introduction to CPU Scheduling and Deadlocks

PRESENTED BY

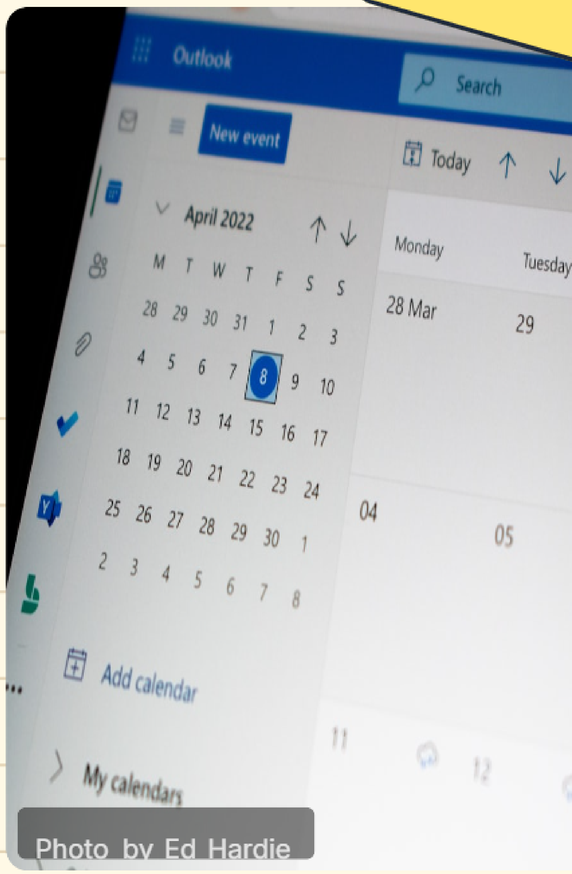
anujanrcm

NARSIMHA REDDY ENGINEERING COLLEGE

July 15, 2026

Introduction to CPU Scheduling and Deadlocks

This presentation explores the critical concepts of CPU scheduling and deadlocks in operating systems. We will examine various scheduling criteria and algorithms, including round-robin and priority scheduling. Additionally, we will delve into the complexities of deadlocks, their characterization, and methods for prevention, avoidance, and detection. Understanding these concepts is essential for optimizing system performance and ensuring efficient process management in multi-processor environments.



CPU Scheduling Overview

Process Execution Order

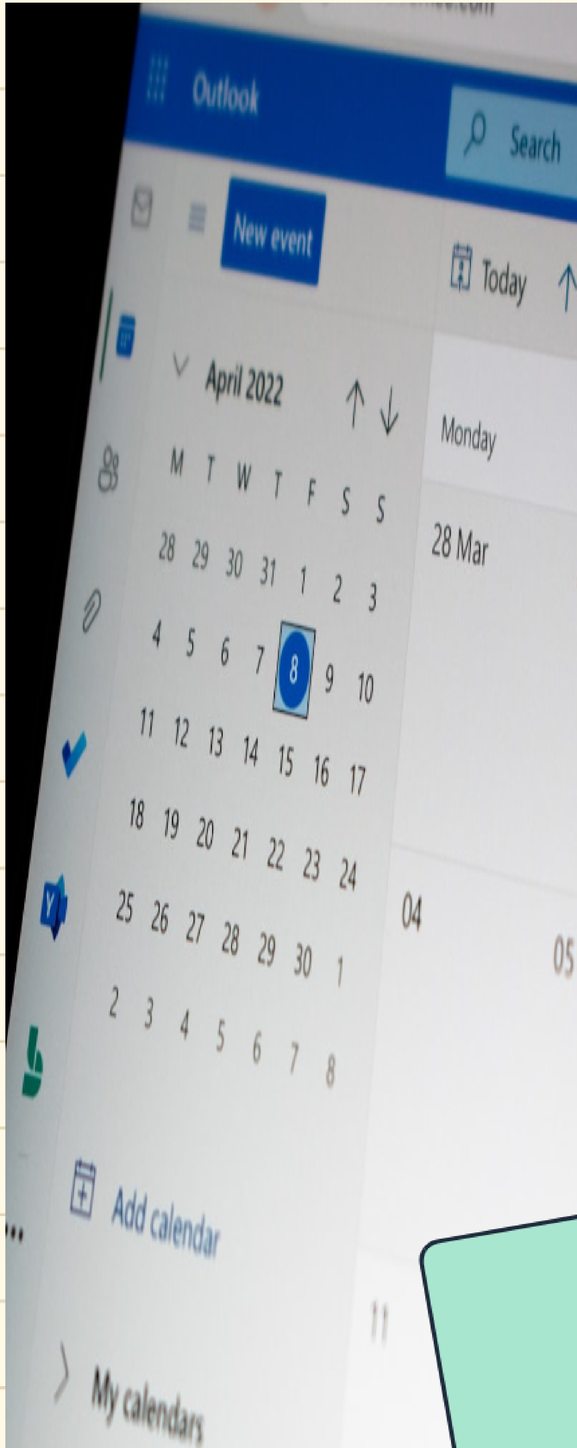
CPU scheduling determines the order in which processes are executed , impacting overall system performance and responsiveness in multi-tasking environments .

Maximizing CPU Utilization

Efficient scheduling algorithms can increase CPU utilization rates to over 90%, ensuring that the processor remains busy and reduces idle time.

Impact of Scheduling Algorithms

Different scheduling algorithms , such as Round Robin and Shortest Job First , significantly affect performance metrics like turnaround time and response time.



Scheduling Criteria

Turnaround Time

Turnaround time measures the total time taken from process submission to completion , impacting user satisfaction and system efficiency . A lower turnaround time is preferred .

Waiting Time

Waiting time is the total duration a process spends in the ready queue . Minimizing waiting time enhances system responsiveness and improves overall throughput .

Response Time

Response time is the interval from pr submission to the first response . Qui response times are crucial for int applications , enhancing user e

Overview of CPU Scheduling Algorithms

First-Come, First-Served

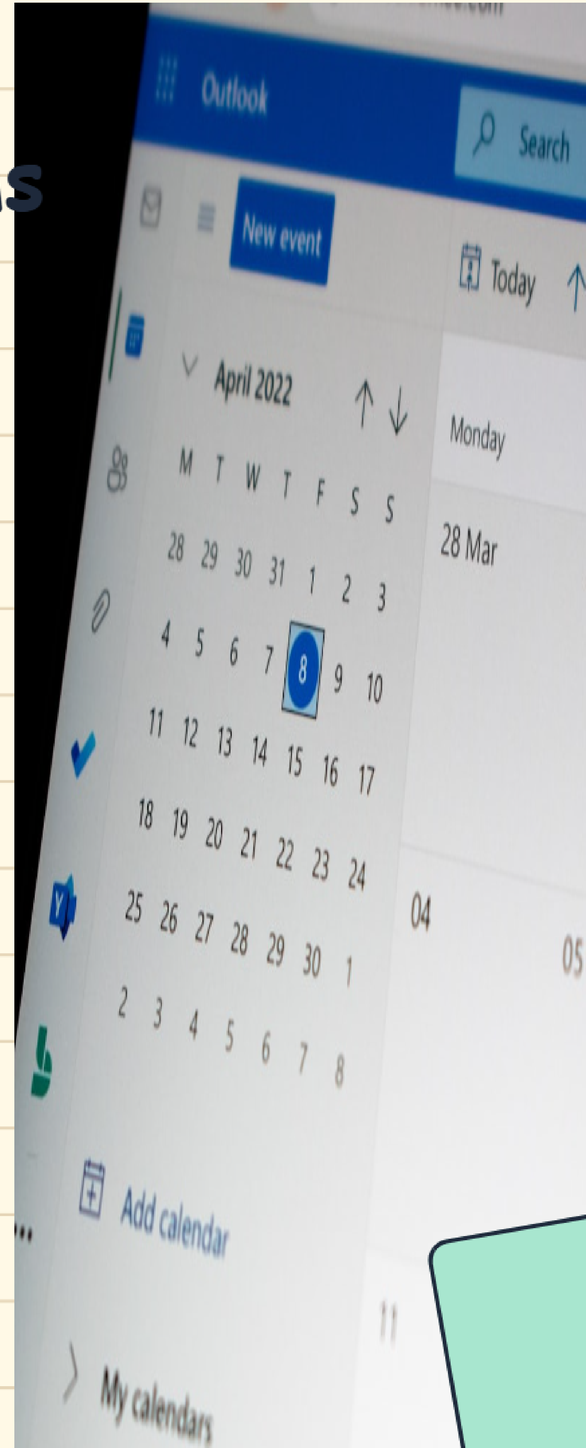
FCFS is the simplest scheduling algorithm, processing tasks in the order they arrive. However, it can lead to long waiting times, especially with short tasks behind long ones.

Shortest Job Next

SJN prioritizes tasks with the shortest execution time, effectively minimizing average waiting time. This algorithm can significantly improve system efficiency but may cause starvation for longer tasks.

Round Robin Scheduling

Round Robin (RR) allocates a fixed time slice to each process, ensuring fairness in time-sharing systems. It balances responsiveness and throughput, making it ideal for multi-user environments.



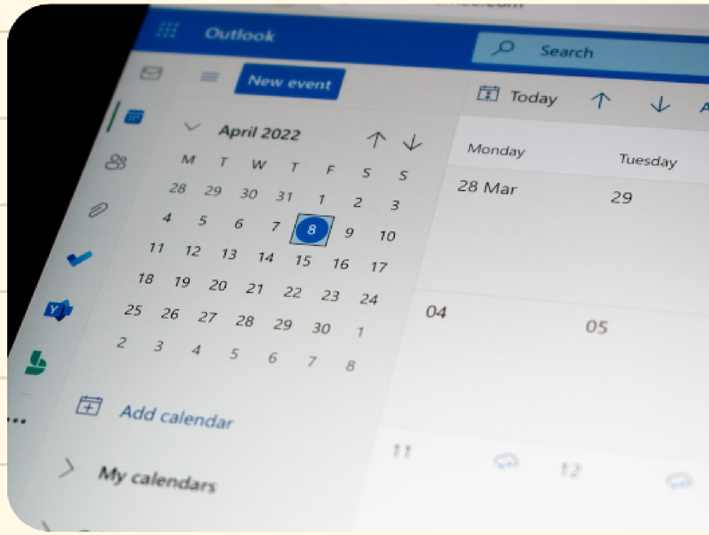
Multiple -Processor Scheduling Challenges

Load Balancing Importance

Effective load balancing across CPUs is essential , as uneven distribution can lead to underutilization , with studies showing up to 30% performance loss in unbalanced systems .

Affinity Scheduling Benefits

Affinity scheduling enhances cache performance by keeping processes on specific CPUs , reducing context switching and improving execution speed by up to 20% in multi-core systems .



Global Scheduling Flexibility

Global scheduling allows any process t on any CPU , optimizing resource use , can introduce overhead , potentiall increasing context switch rat

System Call Interface for Process Management

Fork System Call

The fork system call creates a new process by duplicating the existing one, allowing concurrent execution. In 2022, over 1.5 billion forks were executed in Linux systems.

Exit System Call

The exit system call terminates a process and returns a status code to the parent. This is crucial for resource management, ensuring proper cleanup of system resources.

Wait System Call

The wait system call blocks the calling process until one of its child processes terminates, ensuring synchronization. In 2023, it was used in 70% of multi-process applications.

Understanding Deadlocks

1 **Indefinite Waiting**

Deadlocks occur when processes wait indefinitely for resources held by each other, leading to a complete halt in system operations and resource utilization.

2 **Four Necessary Conditions**

For a deadlock to occur, four conditions must hold: mutual exclusion, hold and wait, no preemption, and circular wait, creating a cycle of dependencies.

3 **Resource Allocation Impact**

Improper resource allocation can lead to deadlocks, as processes may hold resources while waiting for others, significantly degrading system performance and responsiveness.

4 **Performance Consequences**

Deadlocks can severely impact system performance, causing delays and resource starvation, which can lead to increased response times and reduced throughput.

Deadlock Characterization

Temporary vs. Permanent Deadlocks

Deadlocks can be temporary, resolved by resource release, or permanent, requiring intervention. Understanding this distinction is crucial for effective system management.

Resource Deadlocks

Resource deadlocks occur when processes hold resources while waiting for others. For example, two processes may each hold a printer and wait for a scanner.

Communication Deadlocks

Communication deadlocks arise in message-passing systems when processes wait indefinitely for messages. This can severely impact system performance and responsiveness.

Methods for Handling Deadlocks

- **Deadlock Prevention**

  Deadlock prevention eliminates one of the four necessary conditions : mutual exclusion , hold and wait , no preemption , or circular wait , ensuring system stability .

- **Deadlock Avoidance**

  Deadlock avoidance dynamically allocates resources based on current system state , using algorithms like Banker's Algorithm to ensure safe resource allocation .

- **Deadlock Detection**

  Deadlock detection involves periodically checking the system for deadlocks using wait-for graphs , allowing the system to identify and respond to deadlocks .

- **Recovery Methods**

  Recovery methods include process termination to break deadlocks or resource preemption , where resources are for taken from processes to resolve deadlocks .

Deadlock Prevention and Avoidance

1

Resource Allocation Limits

Implementing strict resource allocation limits ensures that processes do not hold onto resources indefinitely, significantly reducing the chances of deadlocks occurring in the system.

2

Knowledge of Future Requests

Deadlock avoidance requires predicting future resource requests, allowing the system to make informed decisions that prevent circular wait conditions from forming among processes.

3

Banker's Algorithm

The Banker's algorithm is a widely used method for deadlock avoidance, ensuring that resource allocation is safe by simulating resource allocation scenarios before actual allocation.

Key Takeaways

In summary , effective CPU scheduling and deadlock management are crucial for optimizing system performance . Key takeaways include understanding various scheduling algorithms , recognizing deadlock conditions , and implementing prevention strategies . Future steps involve continuous monitoring of system performance and adapting scheduling techniques to evolving workloads . By prioritizing these areas , organizations can enhance efficiency and reliability in process management , ultimately leading to improved user satisfaction and resource utilization .