Operating System Course Report - First Half of the Semester

B class

October 1, 2024

Contents

1 Introduction

This report summarizes the topics covered during the first half of the Operating System course. It includes theoretical concepts, practical implementations, and assignments. The course focuses on the fundamentals of operating systems, including system architecture, process management, CPU scheduling, and deadlock handling.

2 Course Overview

2.1 Objectives

The main objectives of this course are:

- To understand the basic components and architecture of a computer system.
- To learn process management, scheduling, and inter-process communication.
- To explore file systems, input/output management, and virtualization.
- To study the prevention and handling of deadlocks in operating systems.

2.2 Course Structure

The course is divided into two halves. This report focuses on the first half, which covers:

- Basic Concepts and Components of Computer Systems
- System Performance and Metrics
- System Architecture of Computer Systems
- Process Description and Control
- Scheduling Algorithms
- Process Creation and Termination

- Introduction to Threads
- File Systems
- Input and Output Management
- Deadlock Introduction and Prevention
- User Interface Management
- Virtualization in Operating Systems

3 Topics Covered

3.1 Basic Concepts and Components of Computer Systems

This section explains the fundamental components that make up a computer system, including the CPU, memory, storage, and input/output devices.

3.2 System Performance and Metrics

This section introduces various system performance metrics used to measure the efficiency of a computer system, including throughput, response time, and utilization.

3.3 System Architecture of Computer Systems

Describes the architecture of modern computer systems, focusing on the interaction between hardware and the operating system.

3.4 Process Description and Control

Processes are a central concept in operating systems. This section covers:

- Process states and state transitions
- Process control block (PCB)
- Context switching

3.5 Scheduling Algorithms

This section covers:

- First-Come, First-Served (FCFS)
- Shortest Job Next (SJN)
- Round Robin (RR)

It explains how these algorithms are used to allocate CPU time to processes.

3.6 Process Creation and Termination

Details how processes are created and terminated by the operating system, including:

- Process spawning
- Process termination conditions

3.7 Introduction to Threads

This section introduces the concept of threads and their relation to processes, covering:

- Single-threaded vs. multi-threaded processes
- Benefits of multithreading

3.8 File Systems

3.8.1 Access Methods

Metode akses file mengacu pada cara file dibaca, ditulis, atau dimodifikasi dalam sistem file. Ada beberapa metode akses yang digunakan dalam sistem file untuk memungkinkan aplikasi atau pengguna mengambil dan memanipulasi data dalam file. Berikut adalah penjelasan rinci mengenai metode akses file yang umum digunakan:

1. Akses Berurutan *(Sequential Access)*

   Sequential Access adalah metode akses file di mana data diakses secara berurutan, dari awal hingga akhir. Ini mirip dengan cara kita membaca sebuah buku, satu halaman per satu secara berurutan.

   Deskripsi:

   - Dalam metode akses ini, file diakses dalam urutan linier tertentu, dari awal hingga akhir. Data dibaca atau ditulis secara berurutan, mirip dengan cara kerja kaset magnetik, di mana untuk membaca bagian tertentu dari data, seluruh file harus dilalui dari awal hingga posisi data tersebut ditemukan.
   - Ini adalah metode akses yang paling sederhana dan sering digunakan untuk aplikasi yang memerlukan pemrosesan data dalam urutan tetap.

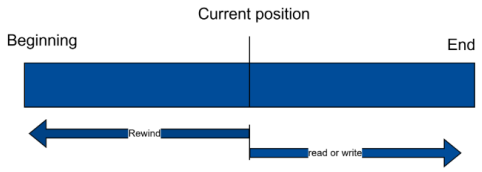

Figure 1: Metode Akses Berurutan

Deskripsi gambar:

- *Current Position*: Menunjukkan posisi saat ini dalam file di mana data akan dibaca atau ditulis. Posisi ini akan bergerak maju setiap kali data dibaca atau ditulis.
- *Rewind*: Jika ingin kembali ke bagian sebelumnya dari file, perlu dilakukan *"rewind"* atau mundur dari posisi saat ini untuk kembali ke posisi sebelumnya (atau bahkan ke awal file).
- *Read and write*: File dibaca atau ditulis secara linier dari posisi saat ini menuju ke akhir file. Data diproses satu per satu dalam urutan dari kiri ke kanan

Kasus penggunaan:

- Arsip
- Media file
- Sistem *backup*: Penyimpanan data yang diakses atau dipulihkan secara berurutan.

Kelebihan:

- Sederhana: Metode ini mudah diimplementasikan dan ideal untuk penggunaan di mana urutan data penting.
- Kinerja: Dapat memberikan kinerja yang baik untuk aplikasi yang membaca atau menulis seluruh file dari awal hingga akhir.

Kekurangan:

- Tidak Fleksibel: Untuk mengakses data di tengah file, seluruh bagian sebelumnya harus dilewati, membuat akses menjadi lambat untuk file besar.
- Tidak efisien: Untuk aplikasi yang memerlukan pengambilan data secara acak, metode ini tidak efisien.

2. Akses Langsung atau Acak *(Direct or Random Access)*

   Direct access adalah metode akses file di mana kita dapat langsung mengakses data pada posisi tertentu tanpa harus membaca seluruh file dari awal. Ini memungkinkan akses yang lebih cepat dan efisien.

   Deskripsi:

   - Dalam metode akses langsung, setiap bagian file dapat diakses secara langsung tanpa perlu membaca data secara berurutan. Sistem dapat melompat ke posisi mana pun dalam file, memungkinkan pengguna untuk membaca atau menulis data di bagian tertentu tanpa harus memproses seluruh file.
   - Akses langsung dimungkinkan karena perangkat penyimpanan modern seperti hard disk drive (HDD) dan solid-state drive (SSD) mendukung lompatan ke lokasi byte atau blok tertentu.

   Deskripsi gambar:

   - *Reset*

Sequential accses	implementation of direct accses
reset	cp = 0;
read next	read cp; cp = cp+ 1;
write next	write cp; cp = cp + 1;

Figure 2: Metode Akses Langsung

  - *Sequential Access*: Reset berarti mengembalikan posisi file ke awal, agar dapat diakses dari posisi pertama.
  - *Direct Accses*: Pada akses langsung, ini diimplementasikan dengan mengatur variabel cp (current position) menjadi 0 (), yang berarti mengarahkan penunjuk file ke posisi pertama

- *Read Next*:
  - *Sequential Access*: Membaca file berikutnya berarti membaca data secara berurutan, satu per satu.
  - *Direct Accses*: Dalam akses langsung, membaca data dilakukan dari posisi yang ditunjukkan oleh cp (). Setelah membaca data, posisi file kemudian ditingkatkan dengan 1 (), sehingga penunjuk siap membaca data berikutnya
- *Write Next*: File dibaca atau ditulis secara linier dari posisi saat ini menuju ke akhir file. Data diproses satu per satu dalam urutan dari kiri ke kanan
  - *Sequential Access* : Menulis data berikutnya dilakukan secara berurutan, setelah data yang sudah ada.
  - *Direct Accses*: Dalam akses langsung, menulis dilakukan pada posisi cp saat ini (). Setelah menulis data, posisi penunjuk juga ditingkatkan dengan 1 (), sehingga siap untuk menulis data selanjutnya.

Kasus penggunaan:

- Basis data (Database): Akses langsung memungkinkan pengambilan catatan individu dalam database tanpa perlu membaca seluruh file.
- File indeks: Dalam file indeks, posisi tertentu di dalam file dapat diakses dengan cepat berdasarkan nilai kunci.
- File konfigurasi: File yang memerlukan modifikasi pada bagian tertentu tanpa mengubah keseluruhan isi file.

Kelebihan:

- Efisiensi: Metode ini sangat efisien untuk aplikasi yang memerlukan akses cepat ke bagian tertentu dari file tanpa harus membaca keseluruhan file.
- Fleksibilitas: Data dapat dimodifikasi, ditambah, atau dihapus pada bagian mana pun dalam file

Kekurangan:

- Lebih rumit: Implementasi akses langsung membutuhkan manajemen lebih kompleks untuk melacak lokasi fisik data pada media penyimpanan.
- Overhead: Mengelola dan memelihara indeks atau metadata untuk mendukung akses langsung bisa menambah overhead.

3. Akses Terindex *(Indexes Access)*

   Deskripsi:

   - Akses terindeks menggunakan indeks yang mengaitkan nilai kunci dengan lokasi blok data di dalam file. Indeks berfungsi seperti daftar isi yang memungkinkan sistem untuk menemukan lokasi data tanpa harus menelusuri seluruh file.
   - Setiap entri dalam indeks berisi kunci pencarian dan pointer ke lokasi data yang sesuai di file.

   Kasus penggunaan:

   - Sistem basis data: Pengindeksan sering digunakan dalam database untuk mempercepat pencarian data berdasarkan kunci unik (misalnya, nomor ID atau nama).

- File besar: Dalam file yang sangat besar atau file yang berisi jutaan catatan, indeks memungkinkan pencarian data yang cepat.

Kelebihan:

- Pencarian cepat: Akses ke data sangat cepat, bahkan dalam file yang sangat besar, karena pencarian langsung dilakukan melalui indeks tanpa membaca file secara keseluruhan.
- Skalabilitas: Dapat diimplementasikan dalam sistem besar dengan banyak data.

Kekurangan:

- Overhead penyimpanan: Indeks memerlukan ruang tambahan untuk menyimpan informasi indeks.
- Overhead pemeliharaan: Indeks harus diperbarui setiap kali ada perubahan pada file (misalnya, penambahan atau penghapusan data), yang dapat menyebabkan peningkatan beban kerja.

3.9 Deadlock Introduction and Prevention

Explores the concept of deadlocks and methods for preventing them:

- Deadlock conditions
- Deadlock prevention techniques

3.10 User Interface Management

This section discusses the role of the operating system in managing the user interface. Topics covered include:

- Graphical User Interface (GUI)
- Command-Line Interface (CLI)
- Interaction between the user and the operating system

3.11 Virtualization in Operating Systems

Virtualization allows multiple operating systems to run concurrently on a single physical machine. This section explores:

- Concept of virtualization
- Hypervisors and their types
- Benefits of virtualization in modern computing

4 Assignments and Practical Work

4.1 Assignment 1: Process Scheduling

Students were tasked with implementing various process scheduling algorithms (e.g., FCFS, SJN, and RR) and comparing their performance under different conditions.

4.2 Assignment 2: Deadlock Handling

In this assignment, students were asked to simulate different deadlock scenarios and explore various prevention methods.

4.3 Assignment 3: Multithreading and Amdahl's Law

This assignment involved designing a multithreading scenario to solve a computationally intensive problem. Students then applied **Amdahl's Law** to calculate the theoretical speedup of the program as the number of threads increased.

4.4 Assignment 4: Simple Command-Line Interface (CLI) for User Interface Management

Students were tasked with creating a simple **CLI** for user interface management. The CLI should support basic commands such as file manipulation (creating, listing, and deleting files), process management, and system status reporting.

4.5 Assignment 5: File System Access

In this assignment, students implemented file system access routines, including:

- File creation and deletion
- Reading from and writing to files
- Navigating directories and managing file permissions

5 Conclusion

The first half of the course introduced core operating system concepts, including process management, scheduling, multithreading, and file system access. These topics provided a foundation for more advanced topics to be covered in the second half of the course.