Operating System Course Report - First Half of the Semester

A class

October 7, 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 Proses Describes and control

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

3.4.1 Process Creation

Process creation adalah tahap di mana sistem operasi menciptakan proses baru:

1. **Inisialisasi PCB:** OS membuat Process Control Block (PCB) yang berisi informasi dasar proses, seperti Process ID (PID), status awal, dan program counter.

2. **Masuk ke Ready Queue:** Proses ditempatkan dalam Ready Queue, menunggu giliran untuk dijadwalkan dan dieksekusi oleh CPU.

3. **Alokasi sumber daya:** OS mengalokasikan sumber daya yang diperlukan oleh proses, seperti memori untuk kode dan data, serta akses ke perangkat I/O.

3.4.2 Process Scheduling

Process Scheduling adalah mekanisme yang berjalan secara terus-menerus untuk menentukan proses mana yang akan dijalankan pada CPU. Penjadwalan terjadi ketika sistem memutuskan proses yang siap untuk dieksekusi atau melakukan context switching antara beberapa proses. Tanpa penjadwalan yang baik, sistem tidak bisa mengalokasikan waktu CPU dengan efisien untuk berbagai proses yang ada.

3.4.3 Process Termination

Process Termination hanya terjadi ketika proses tersebut sudah tidak diperlukan lagi, entah karena telah selesai dieksekusi, terjadi kesalahan, atau karena dihentikan oleh pengguna atau OS. Proses yang dihentikan akan dibuang dari sistem (PCB dihapus, memori dibebaskan), dan tidak lagi dijadwalkan untuk berjalan.

3.4.4 Context Switching

Context switching adalah mekanisme yang memungkinkan sistem operasi untuk berpindah dari satu proses ke proses lain. Setiap kali terjadi perpindahan, OS perlu menyimpan state dari proses yang sedang berjalan (misalnya nilai register, program counter, stack pointer, dll.) dan memuat state dari

proses yang akan dijalankan. Proses penyimpanan dan pemuatan ini memerlukan waktu dan sumber daya, sehingga context switching dianggap sebagai overhead dalam sistem multitasking. Dalam sistem preemptive multitasking, di mana proses-proses dihentikan secara paksa oleh OS untuk memberikan giliran pada proses lain, context switching menjadi sangat penting untuk menjaga ilusi bahwa semua proses berjalan secara bersamaan. Proses ini terjadi ketika CPU berpindah dari satu proses ke proses lain, atau dari proses ke kernel mode, misalnya saat menangani interrupts.

3.5 Process Scheduling

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

Seperti yang terlihat pada Gambar 1, inilah cara menambahkan gambar dengan keterangan.

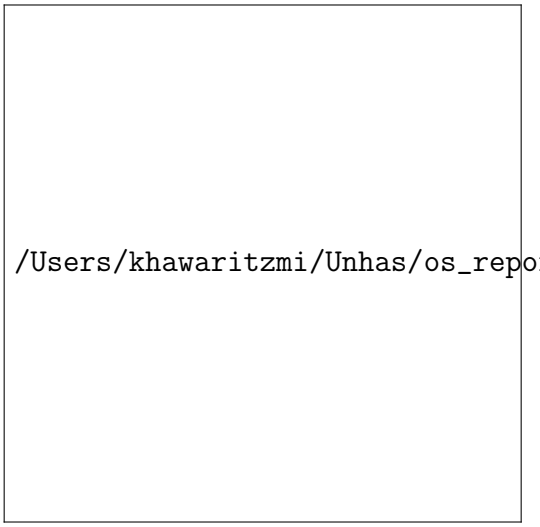

Figure 1: Ini adalah gambar contoh dari multithreading.

3.8 File Systems

File systems provide a way for the operating system to store, retrieve, and manage data. This section explains:

- File system structure
- File access methods
- Directory management

3.9 Input and Output Management

Input and output management is key for handling the interaction between the system and external devices. This section includes:

- Device drivers
- I/O scheduling

3.10 Deadlock Introduction and Prevention

Explores the concept of deadlocks and methods for preventing them:

- Deadlock conditions
- Deadlock prevention techniques

3.11 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

3.12 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.

4.1.1 Group 1

```
    class Process:
    def __init__(self, pid, arrival_time, burst_time):
        self.pid = pid
        self.arrival_time = arrival_time
```

```
        self.burst_time = burst_time
        self.completion_time = 0
        self.turnaround_time = 0
        self.waiting_time = 0
```

Header 1	Header 2	Header 3
Row 1, Column 1	Row 1, Column 2	Row 1, Column 3
Row 2, Column 1	Row 2, Column 2	Row 2, Column 3

Table 1: Your table caption

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**: File creation adalah proses menciptakan berkas baru dalam sistem berkas. Ini dilakukan dengan mendefinisikan

nama berkas dan format yang diinginkan, lalu menyimpan data ke dalam berkas tersebut. Contohnya, saat kita membuat berkas teks, kita dapat menyimpan berbagai informasi atau data di dalamnya.

Sebaliknya, deletion adalah proses menghilangkan berkas yang sudah ada dan tidak lagi diperlukan. Hal ini penting untuk mengelola ruang penyimpanan serta menjaga sistem tetap rapi. Saat menghapus berkas, kita perlu memastikan bahwa berkas yang akan dihapus benar-benar ada, agar tidak terjadi kesalahan.

- **Reading from and writing to files**: Reading berarti mengambil data yang tersimpan dalam berkas untuk digunakan dalam program atau ditampilkan kepada pengguna. Proses ini memungkinkan kita memanfaatkan informasi yang telah ada tanpa harus membuatnya ulang. Misalnya, kita bisa membaca daftar nama dari sebuah berkas dan menampilkannya di layar.

  Writing adalah proses menyimpan data ke dalam berkas. Ini dapat dilakukan dengan menambahkan informasi baru ke berkas yang sudah ada atau membuat berkas baru jika belum ada. Proses ini sangat berguna untuk menyimpan hasil kerja, catatan, atau informasi penting lainnya agar dapat diakses di lain waktu.

- **Navigating directories and managing file permissions**: Navigating directories adalah kemampuan untuk berpindah antara berbagai folder atau direktori dalam sistem berkas. Ini penting karena berkas dan direktori biasanya disusun dalam hirarki, sehingga kita perlu dapat menavigasi struktur tersebut untuk menemukan berkas yang dibutuhkan.

  Managing file permissions adalah proses mengatur siapa yang dapat mengakses dan memodifikasi berkas tertentu. Dalam sistem operasi, izin dapat ditetapkan untuk pengguna tertentu, kelompok, atau semua orang. Hal ini membantu menjaga keamanan data dan memastikan bahwa hanya orang yang berwenang yang dapat mengubah atau menghapus berkas. Misalnya, izin baca, tulis, dan eksekusi dapat diatur untuk setiap berkas dan direktori, memberikan kontrol penuh terhadap akses data.

- **Contoh soal dan pengimplementasian pada code python**: Buatlah program Python yang melakukan hal berikut: Membuat berkas

bernama data.txt, menulis tiga kalimat ke dalam berkas tersebut. membaca isi berkas dan menampilkannya di layar, menghapus berkas data.txt.

```
import os

# 1. Membuat berkas dan menulis ke dalamnya
def create_and_write_file(filename, content):
    try:
        with open(filename, 'w') as file:
            file.write(content)
        print(f"Berkas '{filename}' telah dibuat dan
                                        ditulis.")
    except Exception as e:
        print(f"Terjadi kesalahan saat membuat atau
                                        menulis berkas: {e
                                        }")

# 2. Membaca isi berkas
def read_file(filename):
    try:
        if os.path.exists(filename):
            with open(filename, 'r') as file:
                content = file.read()
                print(f"Isi berkas '{filename}':\n{
                                                content}")
        else:
            print(f"Berkas '{filename}' tidak ditemukan."
                                                )
    except Exception as e:
        print(f"Terjadi kesalahan saat membaca berkas: {e
                                        }")

# 3. Menghapus berkas
def delete_file(filename):
    try:
        if os.path.exists(filename):
            os.remove(filename)
            print(f"Berkas '{filename}' telah dihapus.")
        else:
            print(f"Berkas '{filename}' tidak ditemukan."
                                                )
    except Exception as e:
        print(f"Terjadi kesalahan saat menghapus berkas:
                                        {e}")

# Implementasi
```

```
filename = input("Masukkan nama berkas (misalnya 'data.
                                txt'): ")
content = input("Masukkan konten yang ingin ditulis ke
                                dalam berkas:\n")

create_and_write_file(filename, content)  # Membuat dan
                                menulis berkas
read_file(filename)                        # Membaca isi
                                berkas
delete_file(filename)                      # Menghapus
                                berkas
```

- Output: Saat menjalankan program, output yang dihasilkan akan seperti berikut:

```
Masukkan nama berkas (misalnya 'data.txt'): data.txt
Masukkan konten yang ingin ditulis ke dalam berkas:
Ini adalah kalimat pertama.
Ini adalah kalimat kedua.
Ini adalah kalimat ketiga.
Berkas 'data.txt' telah dibuat dan ditulis.
Isi berkas 'data.txt':
Ini adalah kalimat pertama.
Ini adalah kalimat kedua.
Ini adalah kalimat ketiga.
Berkas 'data.txt' telah dihapus.
```

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.