Operating System Course Report - First Half of the Semester

B class

October 7, 2024

Contents

1 Introduction **3**

2 Course Overview **3**
2.1 Objectives 3
2.2 Course Structure 3

3 Topics Covered **4**
3.1 Basic Concepts and Components of Computer Systems 4
3.2 System Performance and Metrics 4
3.3 System Architecture of Computer Systems 4
3.4 Process Description and Control 4
3.5 Scheduling Algorithms 5
3.6 Process Creation and Termination 5
3.7 Introduction to Threads 5
3.8 File Systems 5
3.9 Input and Output Management 6
3.10 Deadlock Introduction and Prevention 6
3.11 User Interface Management 6
3.11.1 Command-Line Interface (CLI) 7
3.12 Virtualization in Operating Systems 8

4 Assignments and Practical Work **8**
4.1 Assignment 1: Process Scheduling 8
4.1.1 Group 1 9
4.2 Assignment 2: Deadlock Handling 12
4.3 Assignment 3: Multithreading and Amdahl's Law 12
4.4 Assignment 4: Simple Command-Line Interface (CLI) for User Interface Management 13
4.5 Assignment 5: File System Access 13

5 Conclusion **13**

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

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

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

Figure 1: Ini adalah gambar contoh dari multithreading.

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

3.11.1 Command-Line Interface (CLI)

Meskipun GUI dominan dalam penggunaan umum, CLI tetap menjadi alat penting, terutama untuk *administrator* sistem, pengembang, dan pengguna tingkat lanjut. CLI memungkinkan interaksi dengan sistem operasi melalui perintah teks, menawarkan tingkat kontrol dan efisiensi yang sulit dicapai melalui GUI.

Komponen utama CLI dalam sistem operasi modern meliputi:

- Shell: *Interpreter* yang memproses perintah pengguna (contoh: Bash, Zsh, PowerShell).
- Terminal Emulator: Program yang menyediakan antarmuka untuk berinteraksi dengan shell.
- Utilitas Command-line: Kumpulan program yang dapat dijalankan dari CLI untuk berbagai tugas sistem.
- Scripting Capabilities: Kemampuan untuk mengotomatisasi tugas melalui skrip shell.

Keunggulan CLI meliputi:

- Efisiensi: Pengguna dapat melakukan tugas kompleks dengan cepat.
- Otomatisasi: Tugas berulang dapat dengan mudah diskrip dan diotomatisasi.
- Penggunaan Sumber Daya Rendah: CLI memerlukan sumber daya sistem yang jauh lebih sedikit dibandingkan GUI.
- Akses Jarak Jauh: CLI sangat cocok untuk mengelola sistem jarak jauh melalui koneksi jaringan dengan bandwidth rendah.
- Presisi: Memungkinkan kontrol yang sangat terperinci atas operasi sistem.

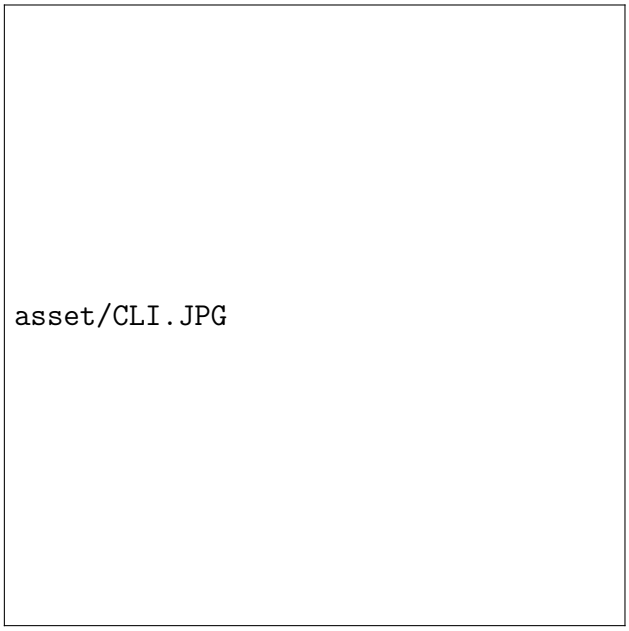

Figure 2: CLI IMAGE

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  # Turnaround time (
                                    completion - arrival)
        self.waiting_time = 0  # Waiting time (turnaround -
                                    burst)

    def calculate_times(self):
        self.turnaround_time = self.completion_time - self.
                                    arrival_time
        self.waiting_time = self.turnaround_time - self.
                                    burst_time

# First Come First Served (FCFS)
def fcfs_scheduling(processes):
    current_time = 0
    for process in sorted(processes, key=lambda p: p.
                                    arrival_time):
        if current_time < process.arrival_time:
            current_time = process.arrival_time
        process.completion_time = current_time + process.
                                    burst_time
        process.calculate_times()
        current_time = process.completion_time

# Shortest Job Next (SJN)
def sjn_scheduling(processes):
    current_time = 0
    remaining_processes = sorted(processes, key=lambda p: (p.
                                    arrival_time, p.burst_time
                                    ))

    while remaining_processes:
        # Mengambil proses dengan burst time terpendek
        available_processes = [p for p in remaining_processes
                                     if p.arrival_time <=
                                    current_time]
        if available_processes:
```

```
            next_process = min(available_processes, key=
                                        lambda p: p.
                                        burst_time)
        else:
            # jika tidak ada proses yang tiba, majukan ke
                                        arrival berikutnya
            next_process = remaining_processes[0]
            current_time = next_process.arrival_time

        current_time += next_process.burst_time
        next_process.completion_time = current_time
        next_process.calculate_times()
        remaining_processes.remove(next_process)

# Round Robin (RR)
def rr_scheduling(processes, quantum):
    current_time = 0
    queue = sorted(processes, key=lambda p: p.arrival_time)
    time_queue = []

    while queue or time_queue:
        # tambahkan proses yang telah sampai di antrian
        while queue and queue[0].arrival_time <= current_time
                                            :
            time_queue.append(queue.pop(0))

        if time_queue:
            process = time_queue.pop(0)

            if process.burst_time > quantum:
                process.burst_time -= quantum
                current_time += quantum
                # mengembalikan proses ke antrian jika belum
                                            selesai
                time_queue.extend([p for p in queue if p.
                                            arrival_time <
                                            = current_time
                                            ])
                time_queue.append(process)
            else:
                current_time += process.burst_time
                process.completion_time = current_time
                process.calculate_times()
        else:
```

```
            current_time = queue[0].arrival_time  # Maju ke
                                                    arrival time
                                                    selanjutnya jika
                                                    antrian kosong

# Fungsi untuk mengatur ulang completion, turnaround, dan
                              waiting times untuk proses
def reset_processes(processes):
    for process in processes:
        process.completion_time = 0
        process.turnaround_time = 0
        process.waiting_time = 0

# Contoh penggunaan:
processes = [
    Process(1, 0, 5),
    Process(2, 1, 3),
    Process(3, 2, 8),
    Process(4, 3, 6)
]

# FCFS
fcfs_scheduling(processes)
print("\n--- FCFS Scheduling ---")
print("PID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting")
for process in processes:
    print(f"{process.pid}\t{process.arrival_time}\t{process.
                                    burst_time}\t"
          f"{process.completion_time}\t{process.
                                        turnaround_time}\t\t
                                        {process.
                                        waiting_time}")

# Mengatur ulang proses untuk algoritma selanjutnya
reset_processes(processes)

# SJN
sjn_scheduling(processes)
print("\n--- SJN Scheduling ---")
print("PID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting")
for process in processes:
    print(f"{process.pid}\t{process.arrival_time}\t{process.
                                    burst_time}\t"
          f"{process.completion_time}\t{process.
                                        turnaround_time}\t\t
```

```
                                                {process.
                                                waiting_time}")

# Mengatur ulang proses untuk algoritma selanjutnya
reset_processes(processes)

# Round Robin Scheduling dengan Quantum = 2
quantum = 2
rr_scheduling(processes, quantum)
print("\n--- Round Robin Scheduling (Quantum = 2) ---")
print("PID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting")
for process in processes:
    print(f"{process.pid}\t{process.arrival_time}\t{process.
                                        burst_time}\t"
          f"{process.completion_time}\t{process.
                                                turnaround_time}\t\t
                                                {process.
                                                waiting_time}")
```

Algorithm	PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
FCFS	1	0	5	5	5	0
	2	1	3	8	7	4
	3	2	8	16	14	6
	4	3	6	22	19	13
SJN	1	0	5	9	9	4
	2	1	3	4	3	0
	3	2	8	23	21	13
	4	3	6	15	12	6
RR	1	0	5	9	9	4
	2	1	3	6	5	2
	3	2	8	22	20	12
	4	3	6	18	15	9

Table 1: Perbandingan FCFS, SJN, dan RR

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.