Operating System Course Report - First Half of the Semester

B class

October 8, 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

3.3.1 Sistem Operasi

Definisi Sistem Operasi:

Sistem operasi merupakan sebuah penghubung antara pengguna komputer dengan perangkat keras komputer. Pengertian sistem operasi secara umum ialah pengelola seluruh sumber daya yang terdapat pada sistem komputer dan menyediakan sekumpulan layanan *(sistem calls)* kepada pemakai sehingga mempermudah dan membuat penggunaan lebih nyaman. Sistem operasi atau dalam bahasa Inggris disebut dengan *operating system* (OS) adalah perangkat lunak sistem yang bertugas melakukan kontrol dan manajemen

terhadap perangkat keras, serta operasi-operasi dasar sistem, termasuk menjalankan aplikasi seperti program-program pengolah kata, desain animasi, pengolah kata, *database* dan *browser web*.

Secara umum, sistem operasi adalah *software* lapisan pertama yang diletakkan pada memori komputer pada saat komputer dinyalakan, sedangkan *software* lainnya dijalankan setelah sistem operasi bekerja. Sistem operasi akan melakukan layanan inti umum untuk semua *software* aplikasi itu. Layanan inti umum tersebut seperti akses ke *disk*, manajemen memori, *skeduling task*, dan antarmuka *user* sehingga masing-masing *software* tidak perlu lagi melakukan tugas-tugas inti umum tersebut, karena dapat dilayani dan dilakukan oleh sistem operasi.

Sebuah Komputer yang *modern* pada saat ini terdiri dari satu atau lebih prosesor, *harddisk*, memori, *printer*, *keyboard*, *monitor*, kartu jaringan, dan beberapa *input/ouput* yang ada di komputer tersebut. Semua itu menjadi sebuah kesatuan dalam suatu sistem yang lengkap.

Dalam banyak kasus, sistem operasi menyediakan suatu pustaka dari fungsi-fungsi standar di mana aplikasi lain dapat memanggil fungsi-fungsi itu sehingga dalam setiap pembuatan program baru tidak perlu membuat fungsi-fungsi tersebut dari awal.

Secara umum, Sistem operasi terdiri dari beberapa bagian:

- Mekanisme *boot*, yaitu meletakkan kernel ke dalam memori.
- Kernel, yaitu inti dari sebuah sistem operasi.
- *Command interpreter* atau *shell*, yang bertugas membaca *input* dari pengguna.
- Pustaka-pustaka, yaitu yang menyediakan kumpulan fungsi dasar dan standar yang dapat dipanggil oleh aplikasi lain.

Tujuan adanya sistem operasi:

Sistem operasi merupakan program yang mengontrol eksekusi program- program aplikasi dan berfungsi sebagai *interface* antara *user* dengan *hardware*. Sistem operasi memiliki tujuan :

- Sistem operasi membuat komputer menjadi lebih mudah dan nyaman untuk digunakan.
- Sistem operasi memungkinkan sumber daya sistem komputer untuk digunakan dengan cara yang efisien.
- Sistem operasi harus disusun sedemikian rupa sehingga memungkinkan pengembangan yang efektif, pengujian dan penerapan fungsi-fungsi sistem yang baru tanpa mengganggu layanan yang telah ada.

Dalam *software* sistem operasi, terdapat istilah seperti *platform*, serta CLI *(Command Line Interfaces)* dan GUI *(Graphical User Interfaces)*. Platform yaitu *software* sistem operasi yang digunakan pada sebuah komputer. CLI dan GUI mempunyai perbedaan. CLI adalah sistem operasi yang menggunakan perintah-perintah yang ditulis dalam baris teks. GUI adalah sistem operasi *macintosh* yang hanya bisa dihasilkan oleh perusahaan komputer, *Apple*.

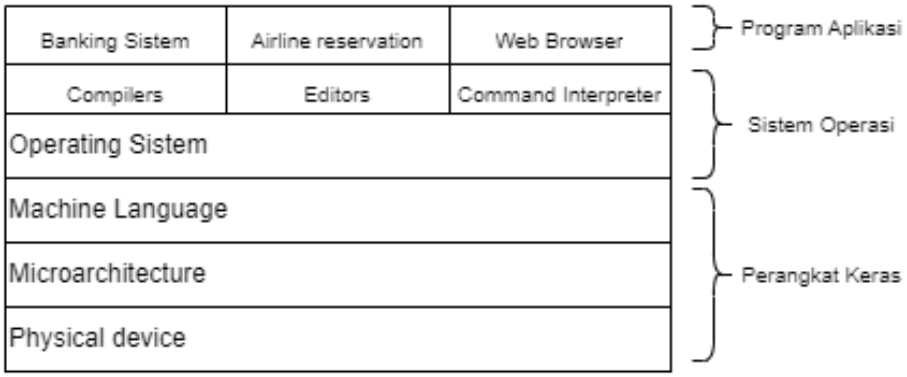

Figure 1: Sebuah komputer yang terdiri dari perangkat keras, sistem operasi, dan program aplikasi

Referensi :

Haryanto,E.V.(2012).*Sistem Operasi Konsep dan Teori*.CV Andi Offset

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 2, inilah cara menambahkan gambar dengan keterangan.

Figure 2: 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

4.1.2 Group 2

4.1.3 Group 3

Implementasikan algoritma *First-Come*, *First-Served* (FCFS), *Shortest Job Next* (SJN), dan *Round Robin* (RR) untuk proses *scheduling*. Bandingkan waktu eksekusi rata-rata *(average waiting time)* dari ketiga algoritma tersebut dengan jumlah proses yang berbeda.

```
    # FCFS Scheduling
def fcfs(processes):
    n = len(processes)
    wait_time = [0] * n
    for i in range(1, n):
        wait_time[i] = processes[i-1][1] + wait_time[i-1]
    avg_wait_time = sum(wait_time) / n
    return avg_wait_time

# SJN Scheduling
def sjn(processes):
    processes.sort(key=lambda x: x[1])  # Sort by burst time
    return fcfs(processes)

# RR Scheduling
def round_robin(processes, quantum):
    wait_time = [0] * len(processes)
    rem_burst_time = [p[1] for p in processes]
    t = 0
    while any(rem_burst_time):
        for i in range(len(processes)):
            if rem_burst_time[i] > 0:
```

```
                if rem_burst_time[i] > quantum:
                    t += quantum
                    rem_burst_time[i] -= quantum
                else:
                    t += rem_burst_time[i]
                    wait_time[i] = t - processes[i][1]
                    rem_burst_time[i] = 0
    avg_wait_time = sum(wait_time) / len(processes)
    return avg_wait_time

# Example usage
processes = [(1, 5), (2, 9), (3, 6)]  # (process_id,
                                      burst_time)
print("FCFS:", fcfs(processes))
print("SJN:", sjn(processes))
print("Round Robin:", round_robin(processes, 4))
```

4.2 Assignment 2: Deadlock Handling

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

4.2.1 Group 1

4.2.2 Group 2

4.2.3 Group 3

Simulasikan kondisi *deadlock* pada sistem dengan 3 proses dan 2 *resource*. Jelaskan dan implementasikan metode pencegahan *deadlock* dengan metode *"Bankers Algorithm"*

```
    # Banker's Algorithm
def is_safe(processes, available, max_demand, allocation):
    n = len(processes)
    m = len(available)
    need = [[max_demand[i][j] - allocation[i][j] for j in
                                    range(m)] for i in range(n
                                    )]
    finish = [False] * n
    safe_seq = []
    work = available[:]
```

```
    while len(safe_seq) < n:
        allocated = False
        for i in range(n):
            if not finish[i] and all(need[i][j] <= work[j]
                                     for j in range(m))
                                     :
                for j in range(m):
                    work[j] += allocation[i][j]
                safe_seq.append(i)
                finish[i] = True
                allocated = True
        if not allocated:
            return False, []

    return True, safe_seq

# Example Usage
processes = [0, 1, 2]
available = [3, 3]  # Available resources
max_demand = [[7, 5], [3, 2], [9, 0]]  # Maximum demand of
                                       each process
allocation = [[0, 1], [2, 0], [3, 3]]  # Allocated resources

safe, seq = is_safe(processes, available, max_demand,
                                       allocation)
if safe:
    print("Safe sequence exists:", seq)
else:
    print("System is in deadlock.")
```

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.