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 termination conditions Details how processes are created and terminated by the operating system, including:
  - Process spawning
  - Process termination conditions

3.6.1 Process Normal Termination

Proses Normal Termination adalah suatu kondisi di mana sebuah proses menyelesaikan fungsinya yang dimaksudkan tanpa mengalami kesalahan atau pengecualian. Ini adalah akhir eksekusi proses yang terkontrol dan disengaja. Dalam kode program, biasanya ada instruksi atau statement yang menunjukkan akhir dari proses, seperti return di fungsi main dalam bahasa C atau Java. Proses ini adalah bagian dari siklus hidup normal sebuah proses.

Skenario umum untuk terminasi normal:

* Penyelesaian proses: Proses telah menyelesaikan semua tugasnya dan mencapai titik akhir alami.
* Terminasi yang diinisiasi oleh pengguna: Pengguna telah secara eksplisit meminta proses untuk berhenti, biasanya melalui perintah atau antarmuka.

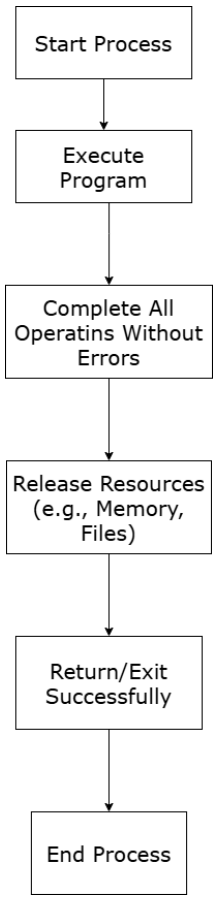

Figure 1: Normal Termination

* Pengembalian dari fungsi: Sebuah proses dapat terminasi ketika sebuah fungsi yang dijalankan mengembalikan nilai yang mengindikasikan penyelesaian.

Karakteristik terminasi normal:

* Proses terminasi dengan cara yang dapat diprediksi dan teratur.
* Tidak ada masalah atau pengecualian yang tidak terduga yang menyebabkan proses berakhir sebelum waktunya.
* Proses melepaskan semua sumber daya yang digunakannya (misalnya, memori, file, koneksi jaringan) sebelum terminasi.

Contoh terminasi normal:

* Sebuah proses browser web menyelesaikan pemuatan halaman web.
* Sebuah proses editor teks menyimpan dokumen dan kemudian keluar.
* Sebuah tugas latar belakang menyelesaikan pekerjaannya yang dijadwalkan dan terminasi.

– 3.6.2 Abnormal Termination

– Abnormal Termination adalah Proses berakhir tidak normal akibat adanya kesalahan yang tidak dapat ditangani dengan benar. Hal ini sering kali disebabkan oleh kondisi yang tidak diharapkan yang membuat proses tidak dapat melanjutkan eksekusi

Penyebab Terminasi Abnormal:

* Kesalahan Logika: Kesalahan dalam kode program, seperti pembagian dengan nol, akses array di luar batas, atau penggunaan pointer yang tidak valid.
* Kehabisan Sumber Daya: Proses kehabisan sumber daya yang dibutuhkan untuk beroperasi, seperti memori, file descriptor, atau waktu CPU.
* Sinyal: Proses menerima sinyal yang memaksanya untuk berhenti, seperti sinyal SIGKILL (terminate immediately) atau SIGSEGV (segmentation fault).

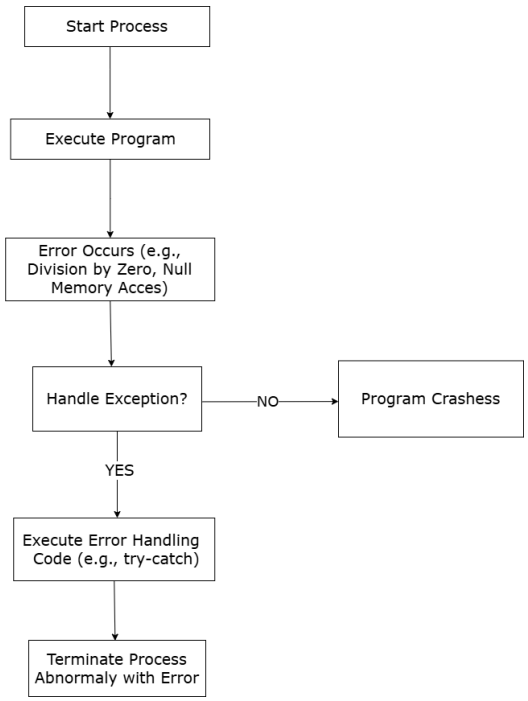

Figure 2: Abnormal Termination

* Kesalahan Perangkat Keras: Kerusakan perangkat keras dapat menyebabkan proses berhenti secara tiba-tiba.
* Deadlock: Dua atau lebih proses saling menunggu satu sama lain, sehingga tidak ada yang dapat melanjutkan eksekusi.
* Interupsi: Terjadinya interupsi yang tidak dapat ditangani oleh sistem operasi.

Kontras dengan terminasi abnormal:

* Terminasi abnormal terjadi ketika sebuah proses berakhir secara tidak terduga karena kesalahan, pengecualian, atau keadaan tak terduga lainnya. Hal ini dapat menyebabkan perilaku yang tidak terkendali, kebocoran sumber daya, atau ketidakstabilan sistem.

Dampak Terminasi Abnormal :

* Kehilangan Data: Data yang sedang diproses oleh proses yang terminasi secara abnormal mungkin hilang atau rusak.
* Ketidakstabilan Sistem: Terminasi abnormal dapat menyebabkan sistem menjadi tidak stabil atau bahkan crash.
* Kerusakan Data: Data yang disimpan di disk atau memori dapat rusak jika proses tidak sempat menyimpan data dengan benar sebelum terminasi.
* Gangguan Proses Lain: Terminasi abnormal suatu proses dapat mempengaruhi proses lain yang berinteraksi dengannya.

Mekanisme Penanganan Terminasi Abnormal:

Sistem Operasi: Sistem operasi akan berusaha untuk mendeteksi dan menangani terminasi abnormal. Ini melibatkan mekanisme seperti:

* Penanganan Pengecualian: Sistem operasi akan menangkap pengecualian yang terjadi selama eksekusi proses dan mengambil tindakan yang sesuai, seperti menghentikan proses atau menampilkan pesan kesalahan.
* Pengawasan Memori: Sistem operasi akan memantau penggunaan memori oleh proses untuk mencegah terjadinya akses memori yang tidak sah.
* Pengawasan Waktu: Sistem operasi akan membatasi waktu eksekusi suatu proses untuk mencegah proses berjalan terlalu lama dan menghabiskan sumber daya sistem.

* Debugging: Pengembang dapat menggunakan debugger untuk melacak kesalahan yang menyebabkan terminasi abnormal dan memperbaiki kode program.

Contoh Terminasi Abnormal

* Program crash: Sebuah program tiba-tiba berhenti bekerja dan menampilkan pesan kesalahan.
* Blue Screen of Death (BSOD): Sistem operasi Windows mengalami kegagalan fatal dan menampilkan layar biru.
* Kernel panic: Sistem operasi Linux mengalami kegagalan yang sangat serius dan tidak dapat melanjutkan operasi.

Pencegahan Terminasi Abnormal:

* Pengujian yang Memadai: Melakukan pengujian yang menyeluruh pada perangkat lunak untuk menemukan dan memperbaiki bug sebelum di-deploy.
* Penanganan Pengecualian: Menulis kode yang dapat menangani berbagai jenis pengecualian yang mungkin terjadi.
* Manajemen Memori yang Baik: Memastikan alokasi dan dealokasi memori dilakukan dengan benar untuk mencegah kebocoran memori.
* Validasi Input: Memeriksa semua input pengguna untuk memastikan bahwa input tersebut valid dan tidak akan menyebabkan kesalahan.

  Wahab, A., Mubarok, R. (2019). Pengenalan Sistem Operasi (Edisi Revisi). Penerbit Andi. Hidayat, M., Santoso, A. (2018). "Analisis Penanganan Error dan Exception Handling Pada Bahasa Pemrograman Java." Jurnal Ilmu Komputer dan Teknologi Informasi, 5(1), 55-62.

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

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.

4.2 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.