Assignment

Site:	Eduvos LMS	Printed by:	Davey Machaka
Course:	Operating Systems Assessments	Date:	Tuesday, 29 July 2025, 9:31 PM
Book:	Assignment		

Table of contents

Assignment

Faculty:	Information Technology
Module Code:	ITOPA3-33
Module Name:	Operating Systems
Content Writer:	Herbert
Internal Moderation:	Community of Practice
Copy Editor:	Kyle Keens
Total Marks:	100
Submission Week:	Week 6

This module is presented on NQF level 7.

5% will be deducted from the student's assignment mark for each calendar day the assignment is submitted late, up to a maximum of three calendar days. The penalty will be based on the official campus submission date.

Assignments submitted later than three calendar days after the deadline or not submitted will get 0%. [1]

Instructions to Students

1. Please ensure that your answer file (where applicable) is named as follows before submission: **Module Code – Assessment Type – Campus Name – Student Number.**
2. Remember to keep a copy of all submitted assignments.
3. All work must be typed.
4. Please note that you will be evaluated on your writing skills in all your assignments.
5. All work must be submitted through Turnitin. The full originality report will be automatically generated and available for the lecturer to assess. Negative marking will be applied if you are found guilty of plagiarism, poor writing skills, or if you have applied incorrect or insufficient referencing. (See the "instructions to students" book activity before this activity where the application of negative marking is explained.)
6. You are not allowed to offer your work for sale or to purchase the work of other students. This includes the use of professional assignment writers and websites, such as Essay Box. You are also not allowed to make use of artificial intelligence tools, such as ChatGPT, to create content and submit it as your own work. If this should happen, Eduvos reserves the right not to accept future submissions from you.

Section A

Learning Objective

1. Evaluate an OS with relation to management systems based on policies and algorithms.
2. Develop an understanding to model business and other non-software systems.
3. Investigate alternatives designs of OS.

Question 1 30 Marks

Study the scenario and complete the question(s) that follow(s):

Operating-System Structure

A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and be modified easily. A common approach is to partition the task into small components, or modules, rather than have one single system. Each of these modules should be a well-defined portion of the system, with carefully defined interfaces and functions. You may use a similar approach when you structure your programs: rather than placing all of your code in the main() function, you instead separate logic into a number of functions, clearly articulate parameters and return values, and then call those functions from main().

Source: Silberschatz's Operating System Concepts, 10th Edition Pages (644-645).

1.1 Critically evaluate the motivations for structuring modern operating systems using a modular approach rather than as monolithic systems. Include theoretical underpinnings and practical benefits related to system complexity, development processes, and scalability.

(8 Marks)

1.2 Analyse how modularisation improves the reliability, maintainability, and extensibility of an operating system. Support your explanation with reference to software engineering principles and relevant real-world examples.

(8 Marks)

1.3 Conduct a detailed comparative analysis of monolithic kernels, layered systems, microkernels, and modular kernel architectures. For each:

- Provide an in-depth description of its structure and operation.
- Critically evaluate the advantages, including impact on system performance, security, and development.
- Assess the disadvantages and limitations, focusing on trade-offs inherent in each design.
- Support your discussion with real-world operating system examples.

(Allocate 8 marks for description, 7 marks for advantages, and 7 marks for disadvantages and trade-offs)

(Total: 14 Marks)

End of Question 1

Question 2 30 Marks

Study the scenario and complete the question(s) that follow(s):

File-System Mounting

Just as a file must be opened before it can be used, a file system must be mounted before it can be available to processes on the system. More specifically, the directory structure may be built out of multiple file-system-containing volumes, which must be mounted to make them available within the file-system name space. The mount procedure is straightforward. The operating system is given the name of the device and the mount point—the location within the file structure where the file system is to be attached. Some operating systems require that a file-system type be provided, while others inspect the structures of the device and determine the type of file system. Typically, a mount point is an empty directory. For instance, on a UNIX system, a file system containing a user's home directories might be mounted as /home; then, to access the directory structure within that file system, we could precede the directory names with /home, as in /home/jane. Mounting that file system under /users would result in the path name /users/jane, which we could use to reach the same directory. Next, the operating system verifies that the device contains a valid file system. It does so by asking the device driver to read the device directory and verifying that the directory has the expected format. Finally, the operating system notes in its directory structure that a file system is mounted at the specified mount point. This scheme enables the operating system to traverse its directory structure, switching among file systems, and even file systems of varying types, as appropriate.

Source: Silberschatz's Operating System Concepts, 10th Edition Pages (644-645).

2.1 Critically explain the file-system mounting process in modern operating systems, highlighting the steps involved and the role of the mount point within the global directory structure**.**

(5 Marks)

2. 2 Discuss why mounting is essential in modern multi-volume file systems and analyse how the operating system verifies the integrity and compatibility of a file system before mounting it. Include potential consequences if verification fails.

(5 Marks)

2.3 Compare and contrast the reasons for storing operating systems in firmware versus disk storage. Evaluate the advantages and constraints of each approach, citing examples from embedded and general-purpose computing environments.

(4 Marks)

2.4 Design a system boot process that supports multiple operating systems on the same hardware. Explain the role of the bootstrap program and describe how it manages the selection and loading of the desired OS. Discuss possible user interactions and fallback mechanisms.

(5 Marks)

2.5 Identify and analyse five core services provided by an operating system. For each service, explain how it simplifies user interaction with the hardware and system resources, and why it cannot be fully implemented by user-level programs alone.

(7 Marks)

2.6 Describe and critically evaluate three general methods of passing parameters to the operating system during system calls, considering efficiency, security, and complexity.

(4 Marks)

End of Question 2

Question 3 20 Marks

Study the scenario and complete the question(s) that follow(s):

Error Detection and Correction

Error detection and correction are fundamental to many areas of computing, including memory, networking, and storage. Error detection determines if a problem has occurred — for example a bit in DRAM spontaneously changed from a 0 to a 1, the contents of a network packet changed during transmission, or a block of data changed between when it was written and when it was read. By detecting the issue, the system can halt an operation before the error is propagated, report the error to the user or administrator, or warn of a device that might be starting to fail or has already failed. Memory systems have long detected certain errors by using parity bits. In this scenario, each byte in a memory system has a parity bit associated with it that records whether the number of bits in the byte set to 1 is even (parity = 0) or odd (parity = 1). If one of the bits in the byte is damaged (either a 1 becomes a 0, or a 0 becomes a 1), the parity of the byte changes and thus does not match the stored parity. Similarly, if the stored parity bit is damaged, it does not match the computed parity. Thus, all single-bit errors are detected by the memory system. A double-bit-error might go undetected, however. Note that parity is easily calculated by performing an XOR (for "eXclusive OR") of the bits. Also note that for every byte of memory, we now need an extra bit of memory to store the parity.

Source: Silberschatz's Operating System Concepts, 10th Edition Pages (498-499)

3.1 Critically analyse the purpose and mechanisms of error detection in computing systems. Discuss how parity works, its detection capabilities and limitations, and describe an advanced method commonly used for error correction in memory.

(10 Marks)

3.2 Outline the requirements for solving the critical-section problem in concurrent systems. Evaluate Peterson's solution against these requirements and discuss its practical limitations in modern computing environments.

(10 Marks)

End of Question 3

Question 4 20 Marks

Scenario

You are a senior system administrator for a cloud-based banking platform that supports concurrent transaction processing, report generation, and auditing. Each of these operations runs as a **separate process**, requiring dynamic allocation of critical system resources such as **CPU cores (A)**, **memory blocks (B)**, **database connections (C)**, and **I/O channels (D)**.

Due to the high volume of transactions, ensuring system **safety** and **deadlock avoidance** is critical. The operating system implements the **Banker's Algorithm** to manage resource allocation. Every process declares its maximum required resources at the start of execution. During runtime, processes request additional resources, which the system evaluates using safety checks.

You are tasked with verifying the safety of the current system state and evaluating whether a resource request from one of the processes can be granted without causing the system to enter an unsafe state.

System Configuration

- **Processes:** 4 (P0, P1, P2, P3)
- **Resource Types:** 4 (A = CPU, B = Memory, C = DB Connections, D = I/O)

Allocation Matrix

Process	**ABCD**
P0	0 1 0 3
P1	2 0 0 1
P2	3 0 2 1
P3	2 1 1 0

Maximum Matrix

Process	**ABCD**
P0	2 2 2 3
P1	3 2 1 2
P2	3 1 2 2
P3	4 2 1 2

Available Resources

ABCD
2 1 1 1

Practical Task
1. Calculate the Need Matrix

(4 marks)

Use the formula:
Need = Maximum - Allocation

2. Determine whether the current system is in a safe state

(10 marks)

Using the **Banker's Algorithm**, perform the following:

- Identify which processes can complete with current available resources.
- Simulate resource release upon completion.
- Determine a **safe sequence**, if it exists.
- Show all working steps.

3. Evaluate a Resource Request from P1: [1, 2, 1, 1]

(6 marks)

Using the Banker's Algorithm:

- Verify that the request does not exceed P1's declared need.

- Check if resources are available.
- If so, simulate allocation and determine whether the system remains in a safe state.
- Justify whether the request **should be granted or denied**.

End of Question 4