Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Day)

Process Scheduling and Memory Management Techniques using Bash

Course Title: Operating System lab
Course Code: CSE 310
Section: 211-D2

Students Details

Name	ID
Md.Anas Khan	201902037
Nura Anha Tamanna	202902002

Submission Date: 06.01.24
Course Teacher's Name: Farjana Akter Jui

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

In the recent era of computing, Scheduling is the method by which threads, processes, or data flows are given access to system resources such as processor time, and communications bandwidth. This is usually done to load balance a system effectively or achieve a target quality of service. The basic reason behind the implementation of Scheduling Algorithms is that the modern computer systems of today rely on multi-tasking and multiplexing. Multi-tasking refers to the process of executing more than one thread at a time and multiplexing refers to the transmission of multiple flows simultaneously. Another reason is that modern systems rely heavily on I/O operations and they have to respond quickly whenever an I/O request is invoked. Besides, in operating system, the CPU plays the more important role of executing processes and controlling flow of the computing smoothly. Deadlocks are the main problems for operating systems tackle, whenever a deadlock occurs execution of processes stops completely. To overcome this deadlock situation there are two algorithms one for avoidance and one for detection. On the other hand, in computing, applications an operating system cannot survive without efficient memory management, especially if an application has to be under surve load for an undefined long time. Resources must be utilized efficiently to enhance performance. We have created a project that describes Process Scheduling Algorithms, Banker's Algorithm for deadlock avoidance, and various Memory Management Techniques in an Operating System.

1.2 Motivation

- With the increasing complexity of computing tasks and the ever-growing demand for system responsiveness, the optimization of process scheduling algorithms and memory management techniques becomes very important. This project presents an opportunity to explore and implement strategies that can enhance system performance.
- The challenges associated with CPU scheduling, deadlock avoidance, and memory allocation are pervasive in computing. Choosing this project offered the

chance to delve into problem-solving at the core of system design.

- Develop robust implementations of fundamental CPU scheduling algorithms, including First-Come-First-Serve (FCFS) and Shortest Job First (SJF), Priority Scheduling, and Round Robin. The goal is to create efficient scheduling mechanisms that optimize system performance and responsiveness.
- To prevent deadlock situations of a system, we were motivated to implement Banker's Algorithm and Deadlock Avoidance. This involves creating a mechanism for intelligent resource allocation to prevent situations that could lead to deadlock scenarios, enhancing system stability.
- Implement various memory management techniques, specifically Multi-programming with Fixed number of Tasks (MFT), Multi-programming with Variable number of Tasks (MVT), and contiguous memory allocation strategies such as First Fit, Best Fit, and Worst Fit. Evaluate the performance of each technique and their suitability in different scenarios.
- Explore the concept of multithreading within the context of First-Come-First-Serve (FCFS) and Shortest Job First (SJF) scheduling algorithms. Implement mechanisms for parallel processing and assess the impact on system throughput and responsiveness.
- By implementing multiple strategies, it becomes possible to evaluate their performance across various scenarios. This understanding is crucial for choosing the most appropriate strategy for a given scenario.
- The choice of Bash programming offered an opportunity to enhance scripting skills. Bash is a powerful and widely used scripting language in Unix-like operating systems, making it an ideal choice for developing solutions that involve system-level operations.

1.3 Problem Definition

1.3.1 Problem Statement

1. Develop Bash scripts to implement various process scheduling algorithms, including First-Come-First-Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, Round Robin, Banker's algorithm, and various Memory management techniques including First fit, Worst fit, Best fit. This involves translating algorithmic concepts into Bash code to simulate the behavior of each scheduling strategy.
2. Develop multithreading within the First Come First Serve (FCFS) and Shortest Job First (SJF) algorithms.
3. Need to investigate how these algorithms handle varying task priorities and execution times, and explore strategies based on workload characteristics.

4. Implement memory management techniques such as Multi-programming with a Fixed number of Tasks (MFT), Multi-programming with a Variable number of Tasks (MVT), and Contiguous Memory Allocation (Best Fit, Worst Fit, First Fit) using Bash scripts. Need to translate the conceptual understanding of these techniques into practical Bash code.

5. Develop Bash scripts to simulate and evaluate how memory management strategies perform under varying conditions, considering factors like process size, and memory availability.

6. Investigate the challenges and opportunities associated with dynamic memory allocation using Bash.

1.3.2 Complex Engineering Problem

The following Table 1.1 is completed according to our above discussion.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	A thorough understanding of process scheduling algorithms, memory management techniques, and multithreading in the context of FCFS and SJF algorithms in Bash programming is essential.
P2: Range of conflicting requirements	Balancing conflicting requirements involves implementing scheduling and memory management techniques that consider diverse workloads, prioritize tasks, and optimize resource utilization. Multithreading introduces complexity, requiring careful consideration of synchronization and thread management.
P3: Depth of analysis required	A deep analysis is required to assess the performance of scheduling algorithms, the suitability of memory management techniques in various scenarios, and the impact of multithreading on FCFS and SJF algorithms. Analytical tools and simulations will be utilized.
P4: Familiarity of issues	Familiarity with key issues related to operating systems, CPU scheduling, memory management, and Bash programming will be emphasized. Educational resources and documentation will ensure a solid understanding of these concepts.
P5: Extent of applicable codes	Extensive coding in Bash will be applied to implement various CPU scheduling algorithms such as FCFS, SJF, Priority Scheduling, Round Robin, Banker's algorithm, memory management techniques such as MVT, and also First fit, Best fit, Worst fit, and multithreading in FCFS and SJF. Practical coding exercises and examples will be provided to enhance coding skills.
P6: Extent of stakeholder involvement and conflicting requirements	Stakeholders, including system administrators, developers, and decision-makers, will be involved in understanding project goals, providing feedback during implementation, and addressing conflicting requirements. Continuous communication will be maintained to align the project with stakeholders' expectations.
P7: Interdependence	The interdependence between CPU scheduling and memory management will be addressed through an integrated solution.

1.4 Design Goals/Objectives

The objectives of this project are as follows:

- The primary goal is to develop and implement CPU scheduling algorithms (such as First-Come-First-Serve, Shortest Job First, Priority Scheduling, and Round Robin and the Banker's algorithm for deadlock avoidance in a shell scripting environment.
- Implement memory management techniques- MFT, MVT, and Contiguous Memory Allocation (Best Fit, Worst Fit, First Fit), using Bash scripts. This objective focuses on coding and simulating memory allocation strategies to gain hands-on experience with different memory management techniques.
- Evaluate and analyze the suitability of different memory allocation techniques for specific scenarios using Bash scripts.
- Develop Bash scripts incorporating multithreading principles to simulate concurrent execution in FCFS and SJF. Evaluate the impact of multithreading on overall system performance, responsiveness, and resource utilization.

1.5 Application

1. In cloud computing, where resources are shared among multiple users and applications, efficient process scheduling and memory management are critical. The project's implementation of CPU scheduling and memory allocation techniques in Bash can be adapted to enhance the performance of cloud-based systems, ensuring fair resource distribution and responsiveness.
2. In server environments, whether in data centers or cloud infrastructures, often deal with a multitude of concurrent processes. Implementing CPU scheduling algorithms and memory management techniques allows for effective resource allocation, minimizing idle time, and optimizing server performance.
3. Database servers often face varying workloads, requiring effective process scheduling to handle multiple queries and transactions. Memory management techniques play a role in optimizing the use of available memory for caching and data storage. The project's implementation can enhance the performance of database management systems.
4. Multithreading is essential for concurrent processing in modern applications. Implementing multithreading in First-Come-First-Serve (FCFS) and Shortest Job First (SJF) algorithms using Bash provides a practical approach for improving the responsiveness of applications that rely on these scheduling strategies.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The project is designed to simulate various process scheduling algorithms, including CPU scheduling (First Come First Served, Shortest Job First, Priority Scheduling, Round Robin), Banker's Algorithm and Deadlock Avoidance, and Memory Management techniques (Multiprogramming with Fixed Tasks (MFT), Multiprogramming with Variable Tasks (MVT), and First Fit, Best Fit, Worst Fit allocation strategies). The project is developed using Bash scripting, a versatile scripting language. Each algorithm is encapsulated within separate functions, promoting modularity and readability.

2.2 FlowChart

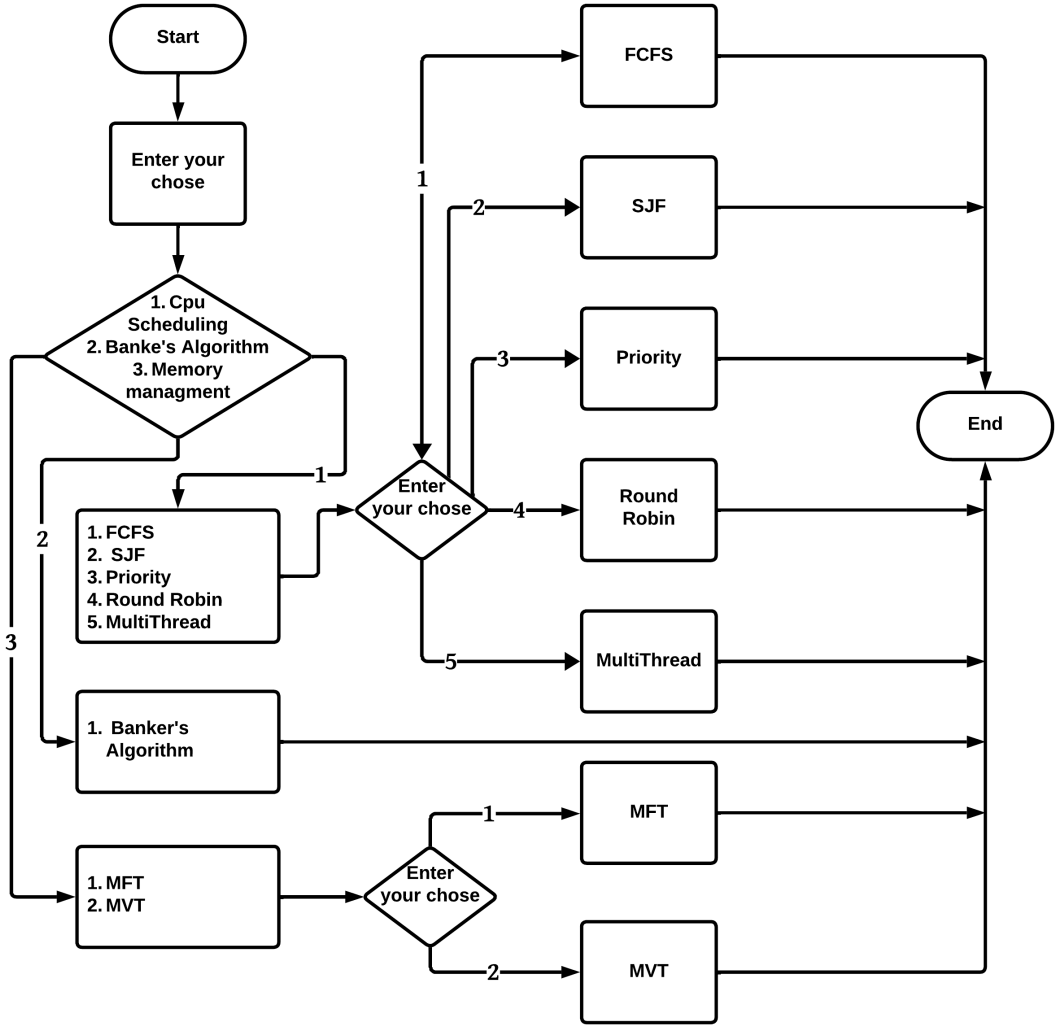


Figure 2.1: Process Scheduling Flowchart

2.3 Implementation

Here's an overview of our project:

1. **Modular Code Structure:** The project adopts a modular approach, with each algorithm and technique implemented as separate Bash functions. This promotes code readability, ease of maintenance, and the ability to focus on specific components independently.

2. **User Interaction:** The script interacts with the user through a command-line interface. It prompts the user to input parameters such as the name and number of processes, burst times, number of resources, memory sizes, block sizes etc.

3. **CPU Algorithms:**

   (a) **First Come First Served (FCFS):** Processes are stored in a queue, and execution occurs based on their arrival order. Waiting times and turnaround times are calculated and displayed.

   (b) **Shortest Job First (SJF):** Processes are sorted based on burst time, with the shortest job scheduled first. Waiting times and turnaround times are calculated and displayed.

   (c) **Priority Scheduling:** Processes are assigned priority values, and the scheduler selects and executes processes based on their priority. Waiting times and turnaround times are calculated and displayed.

   (d) **Round Robin:** Time-slicing is implemented here, and processes are scheduled in a circular manner also displayed specifying which process is fully completed or which is partially completed within their burst times. Waiting times and turnaround times are also calculated and displayed.

   (e) **Multithreading:** Multithreading is implemented in FCFS and SJF algorithms. Processes are simulated to execute concurrently, providing a glance into parallel processing.

4. **Banker's Algorithm and Deadlock Avoidance:** The script simulates the Banker's algorithm for deadlock avoidance. It checks the safety of allocating resources to processes, avoiding unsafe states.

5. **Memory Management Techniques:**

   (a) **Multiprogramming with Fixed Tasks (MFT):** Fixed partitions are allocated to processes, and calculate internal fragmentation and external fragmentation. Memory utilization and fragmentation details are displayed.

   (b) **Multiprogramming with Variable Tasks (MVT):** Variable-sized partitions are allocated, also calculate internal fragmentation. Memory utilization and fragmentation details are displayed.

   (c) **First Fit, Best Fit, Worst Fit Allocation:** Memory is allocated to processes using different allocation strategies. Allocation details, internal fragmentation, and memory usage are displayed for each strategy.

(d) **Suggest Best Fit Function:** The Best Fit function compares the internal fragmentation values obtained for each fit (First Fit, Best Fit, Worst Fit) strategy. It identifies the strategy with the minimum internal fragmentation as the optimal choice for the given scenario. Based on the comparison, the function suggests the strategy (First Fit, Best Fit, Worst Fit) that results in the least internal fragmentation. This suggestion is displayed to the user, providing insights into the most efficient memory allocation strategy for the current workload. If the calculation of internal fragmentation for each fit (First Fit, Best Fit, Worst Fit) remains the same then this function will display to the user that he/she can choose any algorithm as the best fit.

6. **Simulation and Output:** Simulated delays are introduced to mimic real-time execution.

7. **Input both numeric values and strings:** To enhance user interaction and flexibility, we implement the menu option in a way that allows users to input both numeric values and strings for operations. Users can input numeric values corresponding to the menu items or enter the name of the algorithm directly. It provides users with a more intuitive and flexible way to interact with the system.

8. **Exit and Back to Main Menu Option:** After choosing any algorithm to operate if a user wants to exit from that selecting algorithm or go back to the main menu, definitely he can do it. In our project, we also implemented this system where when the user selects the "Exit" option, the program performs necessary cleanup tasks and then terminates. And when the user selects the "Back to Main Menu" option, the program navigates back to the main menu easily.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

1. **Bash Environment** : Ensure that Bash is installed on a system. Bash is usually pre-installed on Unix-based systems. For Windows, anyone can use tools like Git Bash or Cygwin.
2. **Text Editor:** Choose a text editor for writing and editing Bash scripts. Popular choices include Visual Studio Code, Atom, or Sublime Text. Ensure the selected editor supports Bash syntax highlighting for better code readability.
3. **Operating System:** Make sure you are working on an operating system that supports Bash scripting. Common choices are Linux distributions (e.g., Ubuntu, Fedora) or macOS.

3.2 Results Analysis/Testing

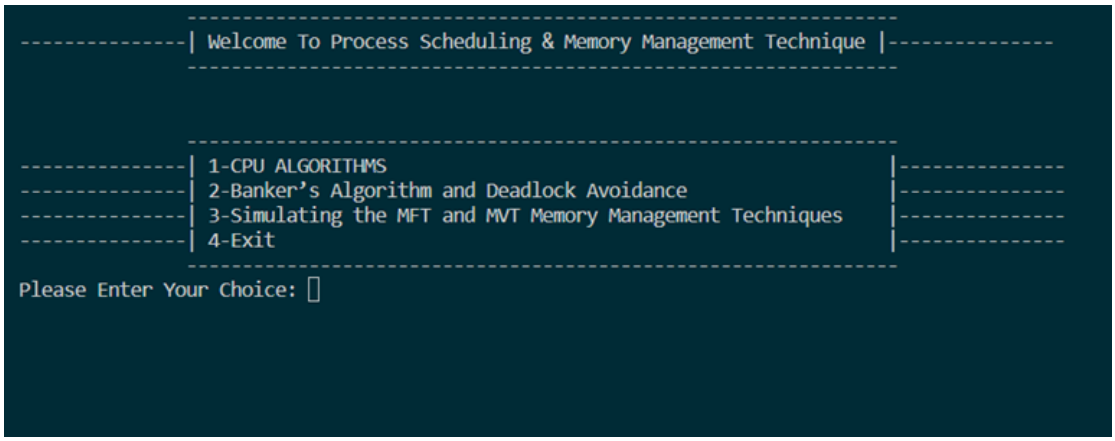


Figure 3.1: Main Menu

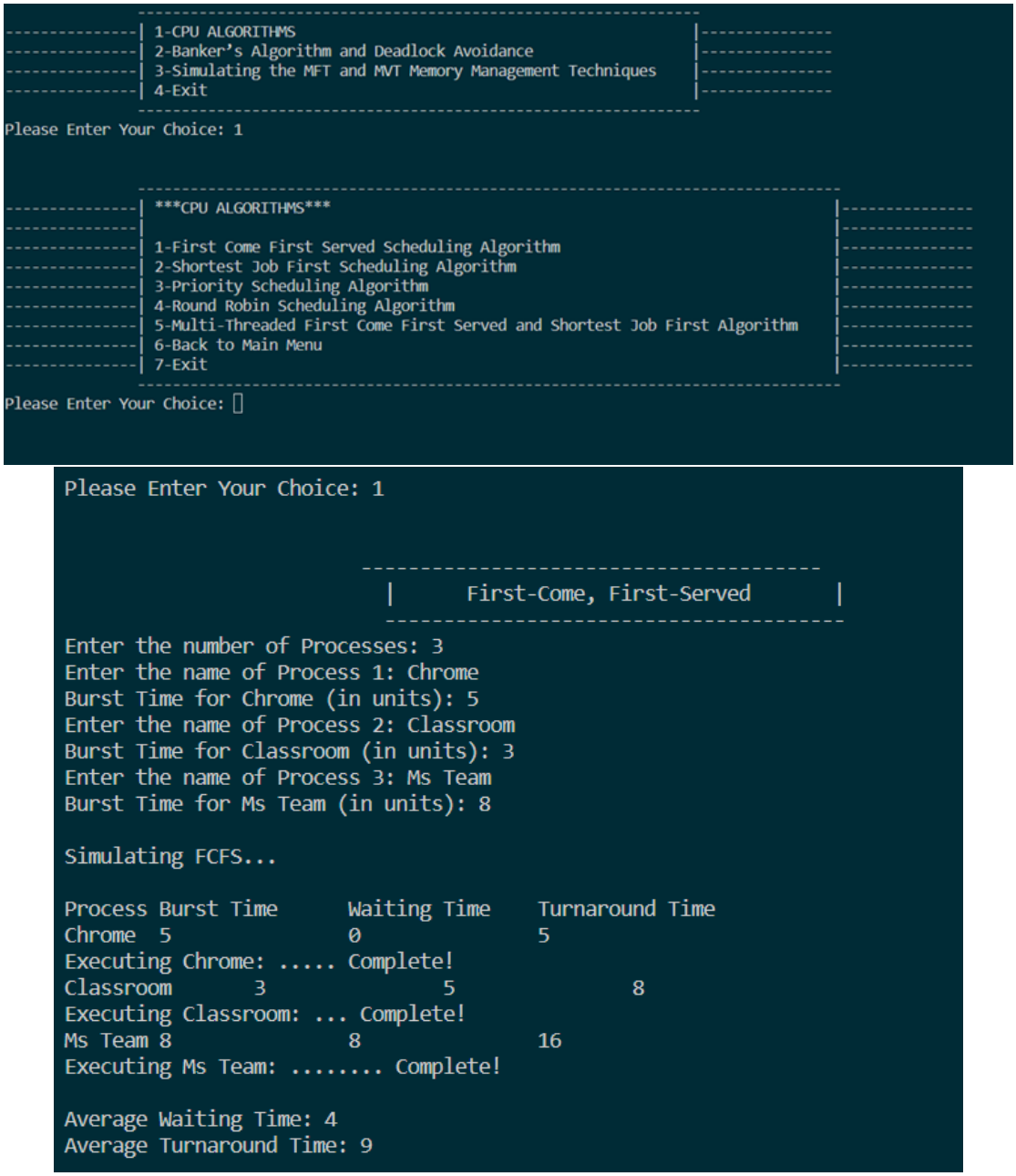
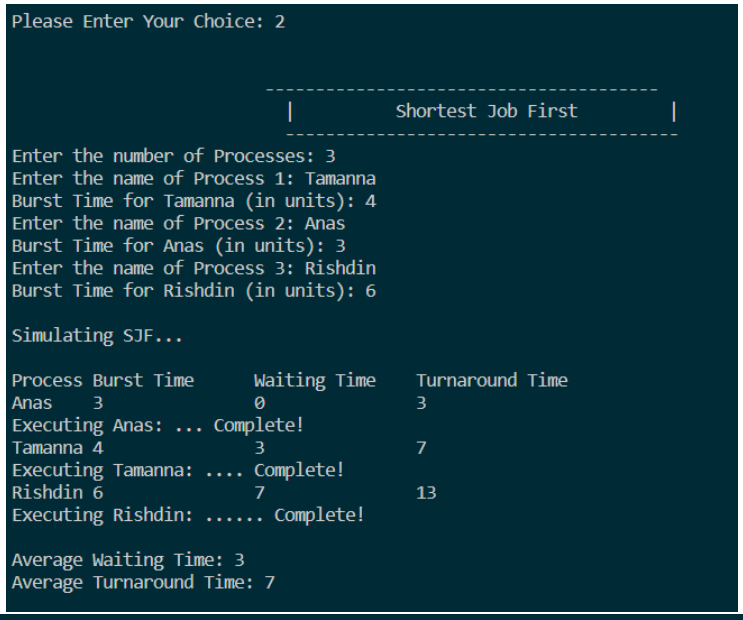


Figure 3.2: CPU Algorithm: FCFS Algorithm



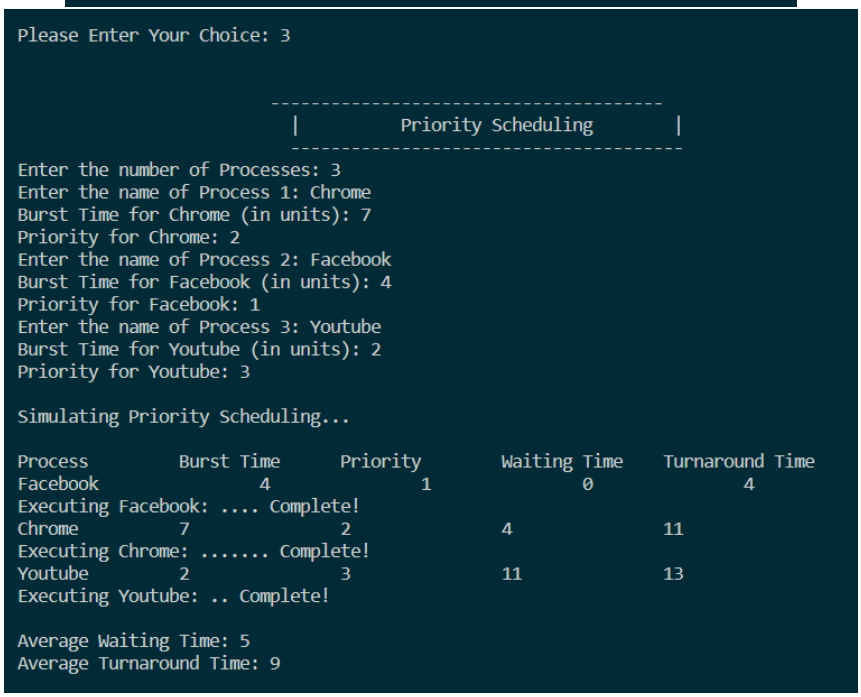


Figure 3.3: CPU Algorithm: SJF and Priority Scheduling Algorithm

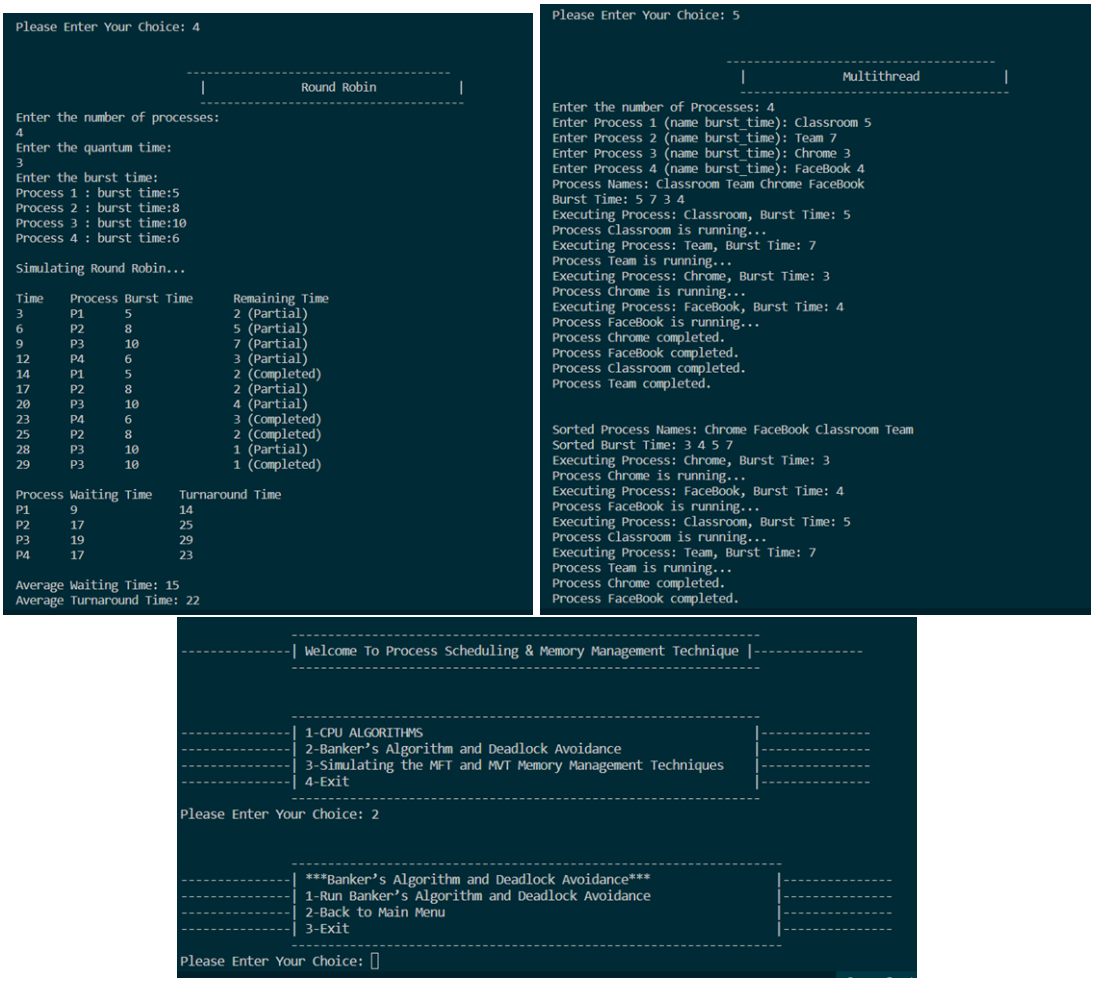


Figure 3.4: CPU Algorithm:(Round Robin Algorithm), Multithreading, and Banker's Algorithm

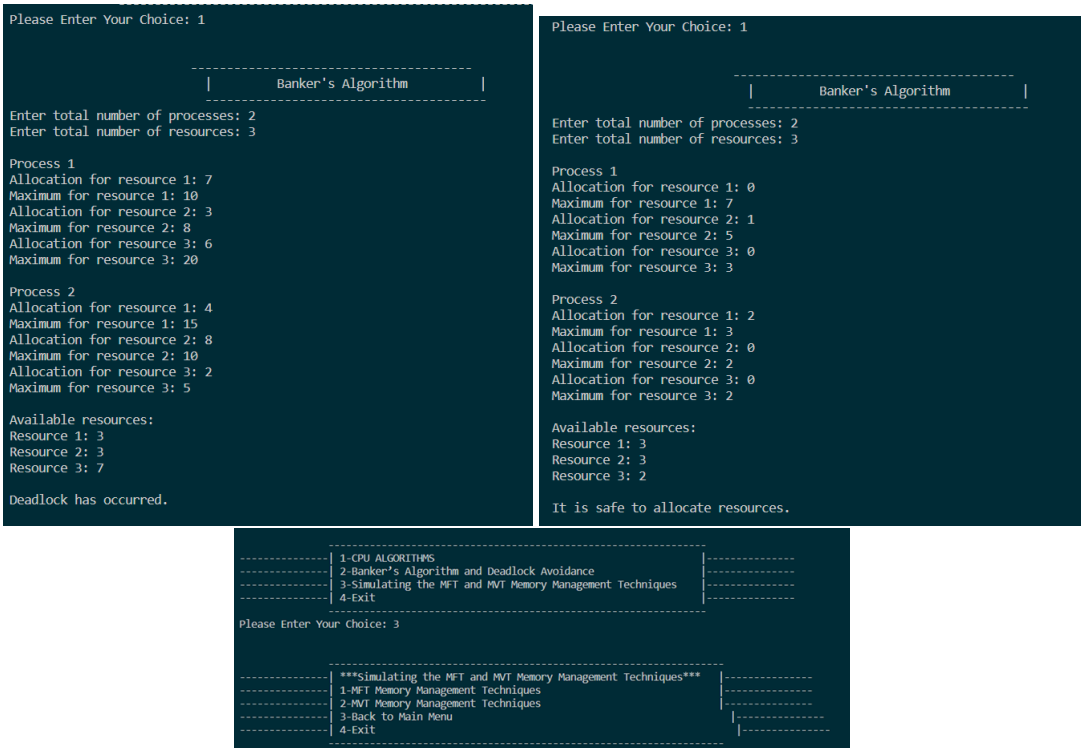


Figure 3.5: Banker's Algorithm:(Deadlock detection, Safety detection), and Memory Management

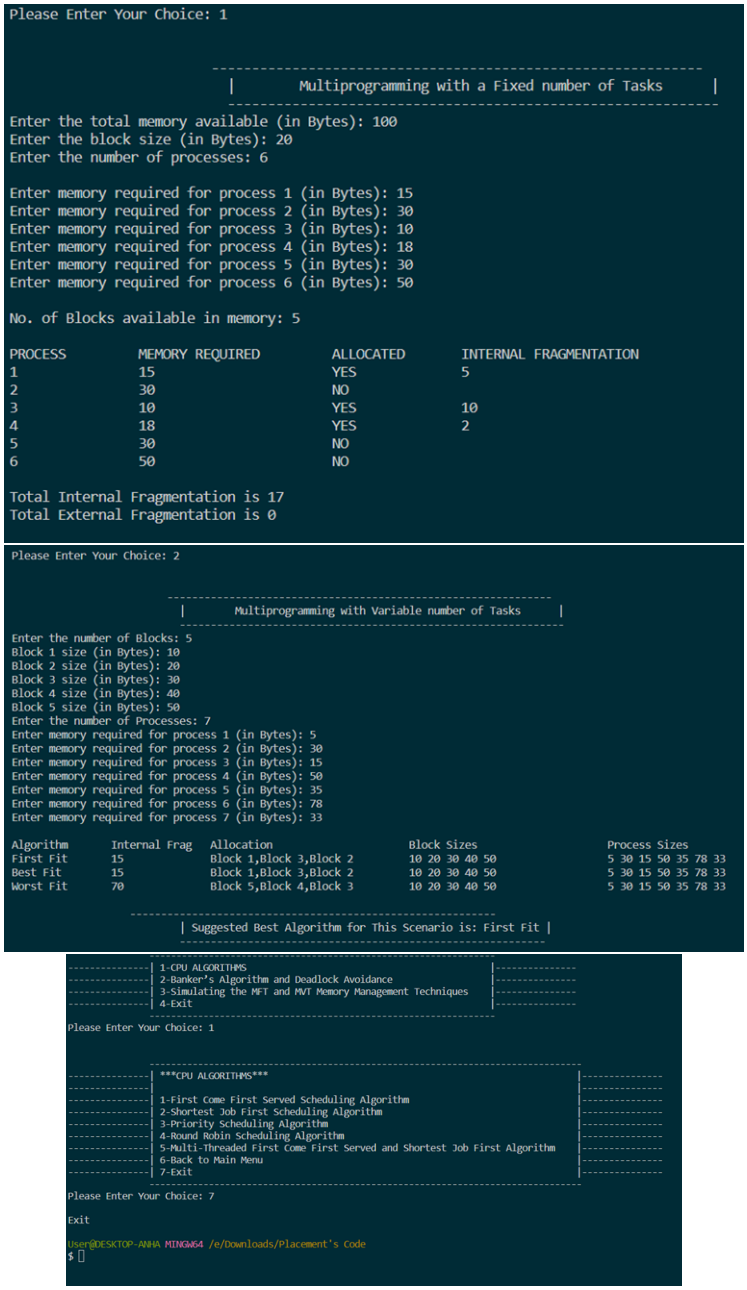


Figure 3.6: Memory Management:(MFT, MVT), and Exit

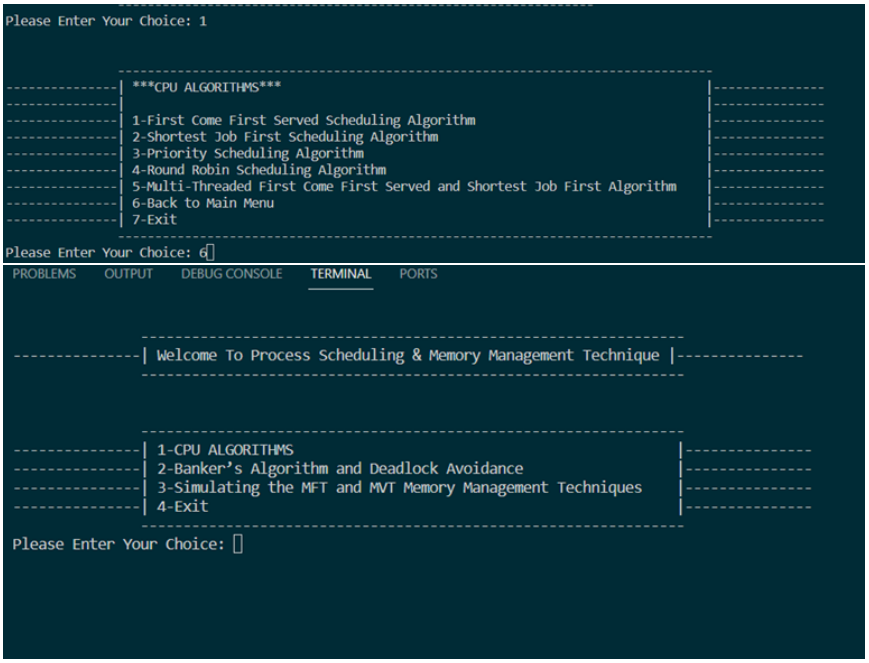


Figure 3.7: Back-To-Main Menu

3.3 Results Overall Discussion

The results of the "Process Scheduling and Memory Management Techniques using Bash programming" reflect the successful implementation of various process scheduling algorithms and memory management techniques.

Chapter 4

Conclusion

4.1 Discussion

In conclusion, this project has achieved its primary objectives by successfully implementing a robust system for Process Scheduling Algorithms and Memory Management Techniques using Bash programming. The core functionalities of the project, including CPU scheduling algorithms, Banker's algorithm, and various memory management strategies (MFT, MVT, First fit, Best fit, Worst fit), were effectively designed and implemented. The implementation of the scheduling algorithms, such as First-Come-First-Serve (FCFS) and Shortest Job First (SJF), provides users with a versatile toolset for handling diverse workloads. The inclusion of multithreading capabilities within FCFS and SJF algorithms adds an additional layer of complexity, allowing for the exploration of concurrent processing scenarios. Memory management techniques, including MFT (Fixed Partitioning), MVT (Variable Partitioning), and different allocation strategies (First fit, Best fit, Worst fit), were implemented to address the challenges of memory allocation and fragmentation. The project exhibits a well-considered balance between flexibility and efficiency, catering to varying demands in memory management scenarios. The incorporation of a menu-driven interface enhances user interaction, providing a user-friendly means to choose and execute specific algorithms or memory management techniques. The seamless transition between numeric and string inputs in the menu exemplifies the project's adaptability and user-focused design, offering a convenient and intuitive experience.

4.2 Scope of Future Work

The current "Process Scheduling Algorithm and Memory Management Techniques using Bash Programming" project lays a solid foundation for future enhancements and expansions. Several properties for future work can further enrich and extend the project's capabilities:

- Expand the project to simulate real-world scenarios, considering factors such as process priorities, dynamic resource demands, and system constraints. Implement scenarios reflecting modern computing environments to make the simula-

tion more realistic and applicable.

- Investigate the impact of memory-related vulnerabilities and implement safeguards against unauthorized access or data breaches.
- Allowing players to customize the game board's theme or appearance adds a personal touch to each match.
- Consider more advanced scheduling algorithms and memory optimization techniques to keep pace with evolving computing requirements.

By achieving these properties future work on this project will be enriched successfully. Iterative development cycles, focused on user satisfaction and engagement, will contribute to the sustained success and growth of this project.

Bibliography

[1] *Vehicular Parkig System Using IoT With FCFS and Round Robin Scheduling Algorithms*. DRam, Bhupendra Pundir, Manisha. (2023). 10.21203/rs.3.rs-3242122/v1.

[2] *Deep reinforcement learning based task-scheduling algorithm in cloud computing*. Multimedia Tools and Applications