Green University of Bangladesh

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

Design and Development of Memory Management Simulator using Shell Script

---

Course Title: Operating System Lab
Course Code: CSE 310
Section: 212D4

Students Details

Name	ID
Zahidul Hasan Sajjad	212002030
Md.Rezaul Karim	212002044

Submission Date: 07/01/2024
Course Teacher's Name: Md. Jahidul Islam

[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

The Memory Management Simulator project is a shell-based tool designed to simulate key aspects of memory management in operating systems. It enables users to interactively allocate and deallocate memory for processes, offering support for various allocation algorithms. This educational tool provides a hands-on experience for students and professionals to explore different memory management strategies and gain practical insights into their impact on system performance.

1.2 Motivation

The motivations behind this project are:

1. Operational Insight: The Memory Management Simulator project is motivated by the desire to provide users with a hands-on experience in allocating and deallocating memory for processes, offering an operational understanding of these critical functions within an operating system.

2. Algorithmic Exploration: Supporting various memory allocation algorithms like First Come First Serve (FCFS) and Best Fit, the simulator encourages users to explore different strategies, fostering a deeper understanding of the diverse approaches to memory management.

3. User Customization: The motivation behind the project lies in enabling users to actively choose their preferred memory allocation and replacement algorithms, promoting customization and experimentation based on individual learning objectives.

4. Practical Skill Development: By offering an interactive and user-friendly environment, the project aims to contribute to the development of practical skills, empowering users to apply their knowledge to simulate and analyze memory management scenarios effectively.

1.3 Problem Definition

1.3.1 Problem Statement

The Memory Management Simulator project aims to address the challenge of limited practical resources for comprehending memory management in operating systems. The current gap lies in the absence of an accessible and user-friendly tool that allows individuals and professionals to actively engage in simulating and visualizing memory allocation and replacement strategies. Existing resources often lack the flexibility to choose and experiment with different algorithms interactively, hindering a deeper understanding of the dynamic nature of memory management. This project seeks to solve this problem by providing a hands-on, customizable, and educational platform.

1.3.2 Complex Engineering Problem

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	The project demands a profound understanding of operating systems, memory management concepts, and algorithmic design. In-depth knowledge of various memory allocation and replacement strategies is crucial for effective implementation.
P2: Range of conflicting requirements	The project involves navigating a diverse range of conflicting requirements, such as optimizing memory utilization while minimizing fragmentation, and balancing the need for simplicity in user interaction with the complexity inherent in memory management algorithms.
P3: Depth of analysis required	Analyzing and optimizing memory management algorithms requires a detailed examination of their performance characteristics. In-depth analysis is essential to identify strengths, weaknesses, and trade-offs associated with different strategies.
P4: Interdependence	There is a high degree of interdependence between different components of the system. Changes in memory allocation strategies can impact overall system performance, necessitating careful consideration of how alterations in one aspect may affect others.

1.4 Design Goals/Objectives

The objective of this project are :

1. To develop an intuitive and user-friendly interface to ensure ease of interaction, catering to both beginners and those familiar with memory management concepts.
2. To enable users to choose and implement different memory allocation and replacement algorithms, fostering a dynamic and customizable learning environment.
3. To implement a real-time visualization feature to allow users to observe the immediate effects of their actions, enhancing the understanding of memory management processesi
4. To design the simulator with a primary focus on its educational value, ensuring it serves as a practical tool for students and professionals to apply theoretical knowledge in simulated scenarios.
5. To build the project with a modular structure that allows for easy expansion and integration of additional memory management algorithms, accommodating diverse educational needs and evolving requirements.

1.5 Application

The Memory Management Simulator project finds practical applications in education, training, and system design. It serves as a valuable tool for teaching operating systems, providing students with hands-on experience in experimenting with different memory management algorithms. In corporate settings, it aids in internal training programs, allowing employees to enhance their understanding of memory management concepts. Additionally, system administrators can use the simulator for practical training in optimizing memory usage and addressing memory-related issues. The open-source nature of the project invites community contributions, fostering collaborative development and expanding its functionalities for broader use.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

In this section, we delve into the design, development, and implementation aspects of our Memory Allocation Simulation Project. Our aim is to provide a comprehensive view of the methodologies, technologies, and logical frameworks employed in bringing this project to fruition. We focus on the systematic approach we followed, ensuring that the simulation is both educational and reflective of real-world scenarios.

2.2 Project Details

2.2.1 User Interface

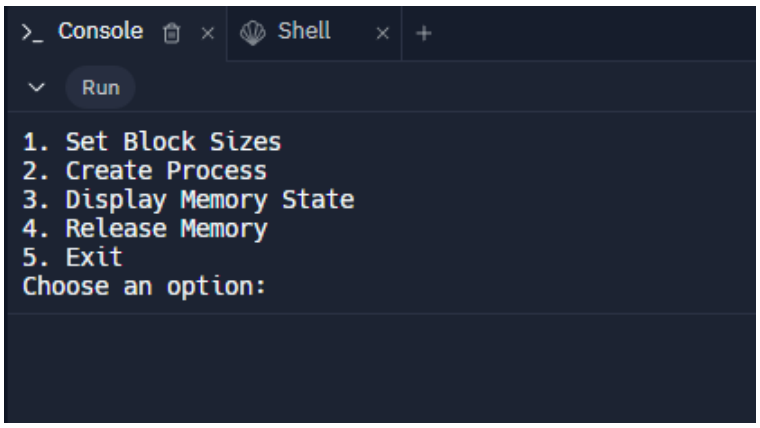

Figure 2.1: Main Menu

The user interface is an essential component of the Memory Management Simulator. It offers users a simple yet interactive menu-driven system where they can select various

memory management operations. Users can create processes, access memory pages, release memory, defragment memory, and view the current memory state.

2.2.2 Script Design and Structure

Our project is structured as a Bash script, chosen for its simplicity and ease of use. The script is designed to simulate a basic memory management system, encompassing various aspects like frame allocation, process management, and memory block structuring. We made use of arrays and control structures in Bash to mimic the behavior of memory allocation in computer systems.

2.3 Implementation

2.3.1 Building the Simulation Environment

The implementation phase involved setting up the simulation environment. We initialized key parameters such as page size, total memory, and the number of frames. This setup forms the backbone of our simulation, providing a framework within which different memory allocation strategies can be applied and analyzed.

2.4 Algorithms

2.4.1 Memory Allocation Strategies

In our project, we implemented three fundamental memory allocation algorithms: First-Fit, Best-Fit, and Worst-Fit.

Allocation Process

1. **First-Fit Algorithm:** This was the first strategy we implemented. It allocates the first block of memory large enough to accommodate the process. Our approach was to iterate through memory blocks and assign the process to the first available space that meets its requirements.
2. **Best-Fit Algorithm:** We then developed the Best-Fit algorithm, which involves searching for the smallest block that is big enough. Our implementation aimed to minimize wastage of memory space but required more computation to find the optimal block.
3. **Worst-Fit Algorithm:** Lastly, the Worst-Fit algorithm was incorporated. This strategy selects the largest available block for allocation. While this might seem counterintuitive, it is based on the idea of leaving the largest possible space free, which could be more useful for future allocations.

In implementing these algorithms, we paid special attention to how they impact memory fragmentation and utilization. Our goal was to not only simulate the allocation

process but also to provide insights into the advantages and drawbacks of each strategy. Through this, we hope to offer users a deeper understanding of memory management in computer systems.

In this section, we have outlined the core elements of our project's design, development, and implementation. Our focus has been on creating a tool that is not only functional but also educational, allowing users to gain practical insights into the complexities of memory allocation in computer systems.

Chapter 3

Performance Evaluation

3.1 Setting Up the Simulation

In setting up the simulation, our goal was to closely mimic the dynamics of memory allocation in a computer system. Using Bash, we crafted a script that establishes a virtual environment, simulating memory frames and processes. Constants such as PAGE_SIZE and TOTAL_MEMORY were defined, along with arrays to represent memory frames and block sizes. This configuration allowed us to explore different memory block setups (fixed or variable) and to test various memory allocation strategies: first-fit, best-fit, and worst-fit.

3.1.1 Execution of the Simulation

For the execution phase, we designed an interactive process where users can configure memory blocks, create processes, and manage memory allocation and deallocation. We provided options to select from three allocation strategies, each influencing how memory is distributed among processes. We meticulously tracked the state of memory frames after each operation to gain insights into the evolution of memory usage and allocation patterns.

3.2 Results Analysis/Testing

3.2.1 Efficiency of Allocation Strategies

Our initial analysis focused on the efficiency of the three allocation strategies. We evaluated efficiency in terms of memory utilization and the capacity to allocate memory to processes within the constraints of total memory. We observed that the first-fit algorithm tends to operate faster but can lead to greater fragmentation over time. In contrast, the best-fit and worst-fit algorithms were scrutinized for their effectiveness in optimizing memory use and reducing fragmentation, though they might incur longer allocation times.

```
1. Set Block Sizes
2. Create Process
3. Display Memory State
4. Release Memory
5. Exit
Choose an option: 1
Choose block type (fixed, variable):
fixed
1. Set Block Sizes
2. Create Process
3. Display Memory State
4. Release Memory
5. Exit
Choose an option: 2
Enter the memory size of the process:
200
Choose the allocation algorithm (first, best, worst):
first
Creating Process P1 with 7 pages using first Fit algorithm...
Page 0 of Process P1 is in Frame 0
Page 1 of Process P1 is in Frame 1
Page 2 of Process P1 is in Frame 2
Page 3 of Process P1 is in Frame 3
Page 4 of Process P1 is in Frame 4
Page 5 of Process P1 is in Frame 5
Page 6 of Process P1 is in Frame 6
```

Figure 3.1: First Fit Allocation

3.2.2 Fragmentation and Process Accommodation

Our attention then turned to how each allocation strategy influences memory fragmentation. We quantified fragmentation by the number of unallocated or partially allocated frames after a series of memory operations. Additionally, we assessed the system's ability to accommodate new processes under these conditions, which provided us with valuable insights into the sustainability of each allocation strategy over time.

3.2.3 Response to Variable Block Sizes

In the final part of our analysis, we examined the system's response to variable block sizes. This aspect was crucial to understand the flexibility of the memory allocation system in adapting to diverse memory demands. We evaluated the performance of each allocation strategy under varying conditions, emphasizing their adaptability and overall effectiveness.

3.3 Results Overall Discussion

In our comprehensive discussion, we integrated findings from various tests and analyses. This involved a comparative review of the allocation strategies in terms of efficiency, fragmentation, and adaptability to fluctuating memory needs. We also deliberated on possible enhancements to the script and the implications of our findings for the development of advanced memory management algorithms in operating systems. We

```
1. Set Block Sizes
2. Create Process
3. Display Memory State
4. Release Memory
5. Exit
Choose an option: 1
Choose block type (fixed, variable):
variable
Enter size for Block 0:
100
Enter size for Block 1:
324
Enter size for Block 2:
300
Enter size for Block 3:
300
```

Figure 3.2: Variable Size block data

concluded with recommendations for future improvements to the simulation environment, aiming for broader and more detailed testing and analysis.

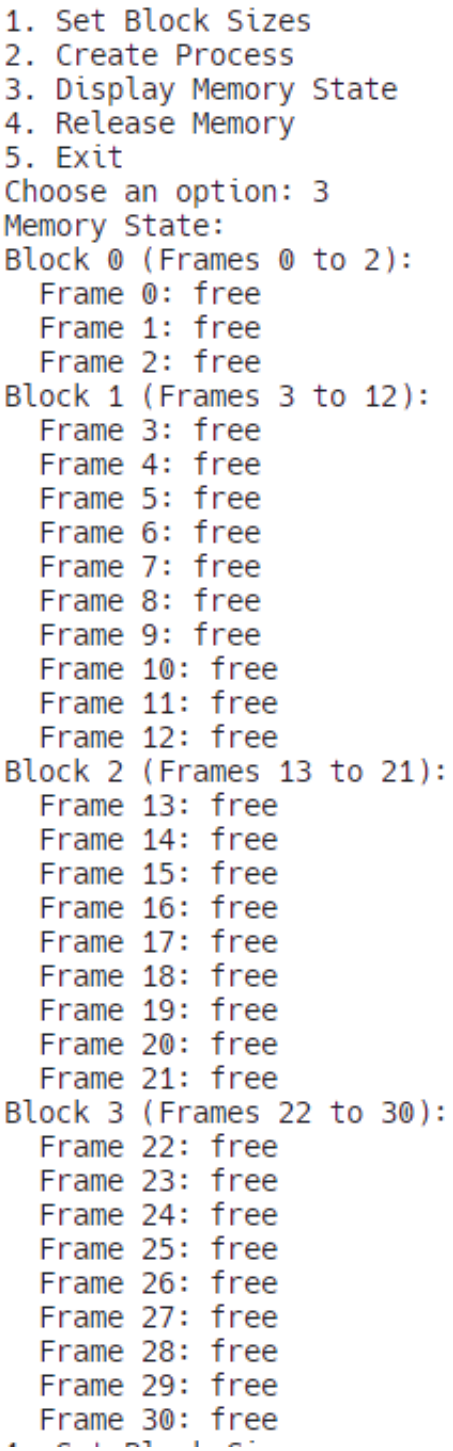

Figure 3.3: Memory state of variable size

Chapter 4

Conclusion

4.1 Discussion

In concluding our project on the Memory Allocation Simulation, we reflect on the significant insights and knowledge we have gained. Through the development of this simulation, we have explored the complexities and nuances of memory management in computer systems. Our implementation of various allocation strategies—First-Fit, Best-Fit, and Worst-Fit—has allowed us to understand their implications in terms of efficiency and resource utilization. This project has not only solidified our theoretical understanding but also provided us with practical experience in simulating and analyzing key aspects of operating systems.

4.2 Limitations

1. **Simplicity of Simulation Environment:**The project's environment is simplified for educational purposes and may not fully represent the complexities of real-world memory allocation.
2. **Predefined Memory Sizes:** The simulation works within set memory sizes, lacking the dynamic allocation capabilities found in actual operating systems.
3. **Basic Performance Metrics:** The metrics used for evaluating memory allocation strategies are fundamental and might not cover all relevant factors in more advanced or practical scenarios..

4.3 Scope of Future Work

Looking forward, there is substantial scope for extending and enhancing this project:

1. **Dynamic Memory Allocation:** We envision an advanced version of the simulation that can handle dynamic memory allocation, more closely mimicking real-world scenarios.

2. **Graphical User Interface (GUI):** Developing a GUI would make the simulation more accessible and user-friendly, particularly for those less familiar with command-line interfaces

3. **Integration with Real Operating Systems:** An ambitious extension could be integrating our simulation with a real operating system to observe memory allocation in real-time.

4. **Educational Modules:** We plan to develop educational modules around this tool, making it a more effective teaching aid for students learning about operating systems.

In conclusion, this project has been a fruitful endeavor in understanding and demonstrating the principles of memory allocation. We believe that the knowledge gained and the potential for future enhancements make this project a valuable asset in both educational and research contexts in the field of computer science.

References

1. Stack Overflow. (n.d.). How to generate a memory shortage using Bash script. Retrieved from https://stackoverflow.com/questions/20200982/

2. PhoenixNAP. (n.d.). Memory Management. Retrieved from https://phoenixnap.com/kb/memory-management

3. Silberschatz, A., Gagne, G., & Galvin, P. B. (Year). *Operating System Concepts*. Publisher.