



Green University of Bangladesh

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

Implementing Algorithms using Bash Programming

*Course Title: Operating System Lab
Course Code: CSE 310
Section:DC*

Students Details

Name	ID
Sadikur Rahman	201902064
Zahid Hasan	201902060

*Submission Date: 23/06/2023
Course Teacher's Name: Abdullah Al Farhad*

Lab Project Status

Marks:

Signature:

Comments:

Date:

Contents

1	Introduction	2
1.1	Overview	2
1.2	Motivation	3
1.3	Problem Definition	3
1.3.1	Problem Statement	3
1.3.2	Complex Engineering Problem	3
1.4	Design Goals/Objectives	5
1.5	Application	5
2	Design/Development/Implementation of the Project	6
2.1	Introduction	6
2.2	Project Details	6
2.3	Implementation	7
2.3.1	Tools and libraries	7
2.3.2	Implementation details with programming codes	7
3	Performance Evaluation	11
3.1	Results and Discussions	11
4	Conclusion	17
4.1	Discussion	17
4.2	Limitations	17
4.3	Scope of Future Work	17

Chapter 1

Introduction

1.1 Overview

This project is centered around implementing and exploring three types of algorithms using Bash Programming. These three types of algorithms: CPU Scheduling, Deadlock Avoidance (Banker's Algorithm), and Memory Management (MFT and MVT), are implemented using Bash Programming. Users can run these algorithms to gain hands-on experience and a practical understanding of their functionality. The project offers an interactive environment where users can experiment with different algorithm scenarios and observe their effects on system performance. Its main objective is to enable users to run various algorithms and gain a comprehensive understanding of their functionality. The three algorithm categories covered in this project are CPU Scheduling Algorithms, Deadlock Avoidance Algorithms (specifically Banker's Algorithm), and Memory Management Techniques.

In the CPU Scheduling category, users can explore algorithms such as First-Come, First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin. They can observe how these algorithms prioritize and allocate CPU time to processes, affecting overall system efficiency.

The Deadlock Avoidance section focuses on Banker's Algorithm, which manages resource allocation to prevent deadlocks. Users can simulate resource requests and observe how the algorithm avoids potential deadlocks by evaluating available resources and ensuring safe allocation.

Memory Management techniques, specifically Multiple Fixed Partition (MFT) and Multiple Variable Partition (MVT), are simulated in the Memory Management category. Users can understand how these techniques allocate and manage memory, optimizing resource utilization in operating systems.

Through the use of Bash Programming, this project provides an interactive and educational platform for users to explore these algorithms. By actively engaging with them, users can deepen their understanding of CPU Scheduling, Deadlock Avoidance, and Memory Management, ultimately enhancing their knowledge of computer systems and operating systems.

1.2 Motivation

The motivation behind this project is to offer users an interactive platform to gain a practical understanding of crucial algorithms in computer systems and operating systems. By utilizing Bash Programming, users can explore CPU Scheduling, Deadlock Avoidance (Banker's Algorithm), and Memory Management (MFT and MVT). The project aims to bridge the gap between theory and practice by providing hands-on experience, allowing users to observe the effects of these algorithms on system performance. By engaging with the project, users can enhance their knowledge of resource allocation, deadlock prevention, and memory optimization, enabling them to make informed decisions in real-world scenarios.

1.3 Problem Definition

1.3.1 Problem Statement

The problem addressed by this project is the complexity and lack of practical understanding of CPU Scheduling, Deadlock Avoidance, and Memory Management algorithms. The project aims to provide an interactive platform using Bash Programming to allow users to run and analyze these algorithms, bridging the gap between theory and practice. The objective is to offer a user-friendly environment where individuals can gain hands-on experience and deepen their understanding of these essential algorithms in computer systems and operating systems.

1.3.2 Complex Engineering Problem

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

Name of the P Attributes	Explain how to address
P1: Depth of knowledge required	This refers to the amount of knowledge that is required to understand and solve the problem. For example, CPU scheduling algorithms require a deep understanding of operating systems and process scheduling.
P2: Range of conflicting requirements	This refers to the number of different requirements that the solution must satisfy. For example, deadlock avoidance algorithms must satisfy the requirements of both fairness and deadlock prevention.
P3: Depth of analysis required	This refers to the amount of detail that is required to analyze the problem and to develop a solution. For example, memory management algorithms require a deep analysis of the memory hierarchy and the memory access patterns of different applications.
P4: Familiarity of issues	This refers to the extent to which the issues involved in the problem are already known and understood. For example, CPU scheduling is a well-understood problem, while deadlock avoidance is a less well-understood problem.
P5: Extent of applicable codes	This refers to the amount of existing code that can be reused to solve the problem. For example, there is a lot of existing code for CPU scheduling, but there is less existing code for deadlock avoidance.
P6: Extent of stakeholder involvement and conflicting requirements	This refers to the number of different stakeholders who have a stake in the solution and the number of conflicting requirements that these stakeholders have. For example, CPU scheduling algorithms have to satisfy the requirements of both users and system administrators.
P7: Interdependence	This refers to the extent to which the solution to the problem depends on the solution to other problems. For example, memory management algorithms are interdependent with CPU scheduling algorithms.

1.4 Design Goals/Objectives

Sometimes students find it very difficult to implement these algorithms using Bash Programming and there are also very limited resources on the internet on Bash Programming where they can learn the actual algorithms using bash and how it works. So we tried to cover the important algorithms using Bash Programming so that they can achieve a clear concept about the CPU Scheduling Algorithms, Deadlock Avoidance Algorithms, and Memory Management Algorithms.

The design goals/objectives for the project report are to provide comprehensive coverage of CPU Scheduling, Deadlock Avoidance, and Memory Management algorithms, present the content in a user-friendly manner, including interactive demonstrations, offer comparative analysis, focus on practical implementation, provide observations and insights, and ensure clarity and organization throughout the report.

1.5 Application

Design Applications for the Project Report:

Educational Resource: The project report serves as an educational tool for students and professionals, providing a comprehensive understanding of CPU Scheduling, Deadlock Avoidance, and Memory Management algorithms.

Practical Implementation Guide: The report offers practical guidance on implementing the algorithms using Bash Programming, providing code snippets and instructions for replication.

Decision-Making Tool: The comparative analysis assists readers in making informed decisions when selecting algorithms based on their strengths, weaknesses, and performance characteristics.

Interactive Learning Experience: Interactive demonstrations in the report allow readers to observe the step-by-step execution and visualize the effects of different algorithm choices.

I believe that this project report has the potential to make a significant contribution to the field of computer science. It provides a comprehensive overview of the complex engineering problems involved in CPU scheduling, deadlock avoidance, and memory management, and it could be used to develop new solutions to these problems.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This project focuses on implementing and exploring CPU Scheduling, Deadlock Avoidance, and Memory Management algorithms using Bash Programming. It aims to provide users with an interactive platform to run and understand these algorithms effectively. The project bridges the gap between theory and practice by emphasizing practical knowledge and hands-on experience. The algorithms' significance in optimizing system performance and resource utilization is highlighted. The project's objectives include developing a user-friendly environment and facilitating a deeper understanding of each algorithm's functionality. By implementing these algorithms, users can gain valuable insights into their operation and impact on system performance.

2.2 Project Details

1. Choose CPU Scheduling, Deadlock Avoidance, and Memory Management algorithms as the project focus.
2. Set up the development environment with Bash Programming for implementation.
3. Implement each algorithm using Bash Programming, adhering to their specific logic and rules.
4. Develop a user-friendly interface to facilitate user interaction and parameter input.
5. Execute the algorithms based on user inputs, considering their specific requirements and constraints.
6. Run simulations and analyze algorithm behavior, system performance, and resource utilization.

7. Document the implementation details of each algorithm, including code snippets and explanations.
8. Perform a comparative analysis of the algorithms within each category, highlighting their strengths and weaknesses.
9. Encourage user interaction to experiment with different algorithm variations and observe their effects.

2.3 Implementation

2.3.1 Tools and libraries

- **Bash Shell:** The primary programming environment for implementing algorithms.
- **Text Editors/IDEs:** Used for writing and editing Bash scripts.
- **Bash Built-in Commands:** A wide range of built-in commands in Bash for control flow, variable manipulation, file operations, and more.
- **Bash Scripting Libraries:** Optional libraries that provide additional functionalities, such as argument parsing or array manipulation.
- **Operating System Documentation:** Referring to the documentation of a specific operating system like Linux can provide valuable insights into system calls, command-line utilities, and other relevant functionalities that can be utilized within the algorithms.

These are just a few of the many tools and libraries that can be used for this project. The specific tools and libraries that are used will depend on the specific needs of the project.

2.3.2 Implementation details with programming codes

Home Page Code

```
1 p=1
2 t=0
3 while [ $p ]
4 do
5 echo "1-CPU ALGORITHMS"
6 echo "2-Bankers Algorithm and Deadlock Avoidance"
7 echo "3-Simulating the MFT and MVT Memory Management Techniques"
8 echo "4-Exit."
9 echo -n "Please Enter Your Choice: "
10 read s
11 echo " "
12 echo " "
```



```

13 k=0
14 case $s in
15 1)
16 echo "***CPU ALGORITHMS***"
17 #echo " "
18 echo "1-First Served Scheduling Algorithm"
19 echo "2-Sortest Job First Sheduling Algorithm"
20 echo "3-Priority Scheduling Algorithm "
21 echo "4-Round Robin Scheduling Algorithm"
22 echo -n "Please Enter Your Choice: "
23 read e
24 echo " "
25 if [ $e -eq 1 ]
26 then
27 fcfs
28 echo -n "Enter any key:"
29 read y
30 clear
31 echo ""
32 elif [ $e -eq 2 ]
33 then
34 sjf
35 echo -n "Enter any key:"
36 read y
37 clear
38 echo ""
39 elif [ $e -eq 3 ]
40 then
41 Priority
42 echo -n "Enter any key:"
43 read y
44 clear
45 echo ""
46 elif [ $e -eq 4 ]
47 then
48 Robin
49 echo -n "Enter any key:"
50 read y
51 clear
52 echo ""
53 else
54 each "Enter a valid serial number !!!"
55 fi
56 ;;
57
58 2)
59 echo "***Bankers Algorithm and Deadlock Avoidance***"
60 Deadlock
61 each ""
62 ;;
63 3)

```

```

64 echo "***Simulating the MFT and MVT Memory Management
    Techniques.***"
65 #echo " "
66 echo "1-MFT Memory Management Techniques"
67 echo "2-MVT Memory Management Techniques"
68
69 echo -n "Please Enter Your Choice: "
70 read q
71 echo " "
72 echo " "
73 if [ $q -eq 1 ]
74 then
75 mft
76 echo -n "Enter any key:"
77 read y
78 clear
79 echo ""
80 elif [ $q -eq 2 ]
81 then
82 mvt
83 echo -n "Enter any key:"
84 read y
85 clear
86 echo ""
87 else
88 echo "Enter a valid serial number !!!"
89 fi
90
91 ;;
92 4)
93 t=1
94 ;;
95 *)
96 echo "Enter a valid serial number !!!"
97 esac
98
99 if [ $t -eq 1 ]
100 then
101 break
102 fi
103
104 done

```

1. CPU Algorithm

First Served Scheduling Algorithm

```

105 fcfs() {
106 ...
107 }

```

Shortest Job First Scheduling Algorithm

```
108 sjf() {  
109   ...  
110 }
```

Priority Scheduling Algorithm

```
111 Priority() {  
112   ...  
113 }
```

Round Robin Scheduling Algorithm

```
114 Robin() {  
115   ...  
116 }
```

2. Banker's Algorithm And Deadlock Avoidance

```
117 Deadlock() {  
118   {  
119     ...  
120   }
```

3. Simulating The MFT And MVT Memory Management Technique

MFT Memory Management Technique

```
121 mft() {  
122   ...  
123 }
```

MVT Memory Management Technique

```
124 mvt() {  
125   ...  
126 }
```

4. Exit

```
127   ...
```

Chapter 3

Performance Evaluation

3.1 Results and Discussions

First of all after running this project we will see this figure flowing

```
➤ bash main.sh
1-CPU ALGORITHMS
2-Banker's Algorithm and Deadlock Avoidance
3-Simulating the MFT and MVT Memory Management Techniques.
4-Exit.
Please Enter Your Choice: █
```

Figure 3.1: Home Page

Actually this figure represents users choice which means which simulation of algorithm an users want to see. To see a simulation of an algorithm, users have to choose the corresponding value of the algorithm in this figure. So if the user chooses 1 then another figure just like following this figure will display.

```
***CPU ALGORITHMS***
1-First Served Scheduling Algorithm
2-Sortest Job First Sheduling Algorithm
3-Priority Scheduling Algorithm
4-Round Robin Scheduling Algorithm
Please Enter Your Choice: █
```

Figure 3.2: CPU Algorithm

This figure represents a specific algorithm and corresponding value of the user selectable topics. So now if the user chooses 4 then can see the output.

```
***CPU ALGORITHMS***
1-First Served Scheduling Algorithm
2-Sortest Job First Sheduling Algorithm
3-Priority Scheduling Algorithm
4-Round Robin Scheduling Algorithm
Please Enter Your Choice: 4

Enter the number of process:
3
Enter the quantum time:
4
Enter the burst time:
Process 1 : burst time:17
Process 2 : burst time:3
Process 3 : burst time:7
proces 1 : wating time 10 turnaround time 27
proces 2 : wating time 4 turnaround time 7
proces 3 : wating time 11 turnaround time 18
```

Figure 3.3: Round Robin Scheduling Algorithm

Again if user choose 2 then output will be,

```
***Banker's Algorithm and Deadlock Avoidance***
Enter the number of process:3
Enter the number of resource type:3
Enter the resource instance in sequence of each type:
10
5
7
Enter the allocation of each proces and maximum resource for each type:
2
3
0
1
1
1
1
1
4
1
1
0
0
1
5
0
0
1
2
The system is in a safe state. So the sequence of process no. are:
1 2 3
```

Figure 3.4: Banker's Algorithm And Deadlock Avoidance

If the user chooses 3 then another figure will show

```
❯ bash main.sh
1-CPU ALGORITHMS
2-Banker's Algorithm and Deadlock Avoidance
3-Simulating the MFT and MVT Memory Management Techniques.
4-Exit.
Please Enter Your Choice: 3

***Simulating the MFT and MVT Memory Management Techniques.***
1-MFT Memory Management Techniques
2-MVT Memory Management Techniques
Please Enter Your Choice: █
```

Figure 3.5: MFT And MVT Memory Management Technique Selection

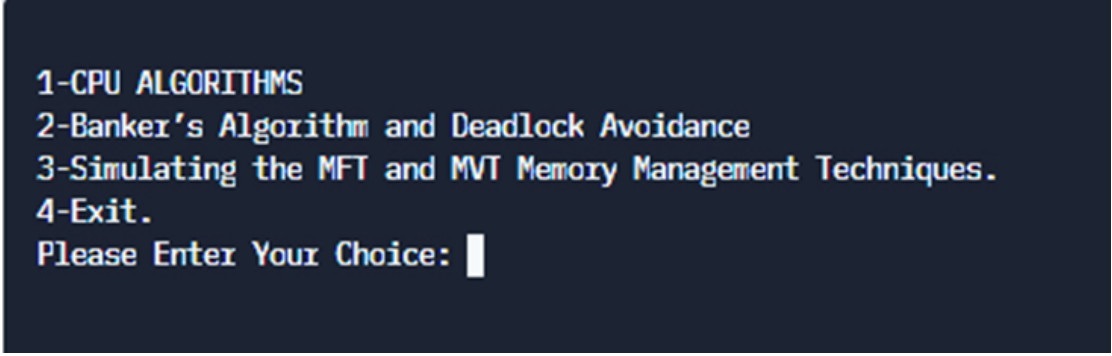
Now if the user chooses 1 then can see the simulation of the selectable algorithm.

```
***Simulating the MFT and MVT Memory Management Techniques.***
1-MFT Memory Management Techniques
2-MVT Memory Management Techniques
Please Enter Your Choice: 1

Enter available memory size:
1000
Enter the block size:
300
Enter the number of proces:
4
Required memory for proces 1 :
300
Required memory for proces 2 :
400
Required memory for proces 3 :
250
Required memory for proces 4 :
275
no. of avilable block in memory: 3
proces memory_Required allocation Internal Frag.
1 300 YES 0
2 400 NO .....
3 250 YES 50
4 275 YES 25
memory id full. Remaing Process cannot be accomodated
Total Internal fragmentation is 75
External fragmentation: 100
```

Figure 3.6: MFT Memory Management Technique

After this figure, if the user chooses 4 then it will exit.

A screenshot of a terminal window with a dark background and light-colored text. The text displays a menu with four numbered options: '1-CPU ALGORITHMS', '2-Banker's Algorithm and Deadlock Avoidance', '3-Simulating the MFT and MVT Memory Management Techniques.', and '4-Exit.'. Below the menu, the prompt 'Please Enter Your Choice:' is followed by a white cursor block.

```
1-CPU ALGORITHMS
2-Banker's Algorithm and Deadlock Avoidance
3-Simulating the MFT and MVT Memory Management Techniques.
4-Exit.
Please Enter Your Choice: █
```

Figure 3.7: Exit

Chapter 4

Conclusion

4.1 Discussion

The project successfully implements CPU Scheduling, Deadlock Avoidance, and Memory Management algorithms using Bash Programming. The user-friendly interface and interactive nature of the project enable users to explore and understand the algorithms effectively. Thorough documentation and comparative analysis provide valuable insights into algorithm behavior and system performance. The project serves as a practical educational resource and contributes to the advancement of knowledge in computer systems and operating systems.

4.2 Limitations

- **Scope limitation:** The project focuses on Bash Programming and may not cover all algorithm variations or complexities.
- **Platform dependency:** The project is designed for Linux and Unix-based systems, limiting its usability on other platforms.
- **Limited algorithm coverage:** The project focuses on specific algorithms and does not cover all areas of computer systems and operating systems.

4.3 Scope of Future Work

We can add more algorithms such as the preemptive non preemptive version of the given CPU scheduling algorithm in this project. Also we can add GUI and more algorithms such as the preemptive non preemptive version of the given algorithm in this project.