



Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Science and Engineering

Semester: (Spring, Year:2024), B.Sc. in CSE (Evening)

File System Management Simulation

Course Title: Operating System Lab

Course Code: CSE-402

Section: 231-EA

Student Details

| Name | | ID |
|------|-------------------|-----------|
| 1. | Md Khorshed Alom | 231015001 |
| 2. | Most. Sumya Akter | 231015015 |

Submission Date : 16-05-2025

Course Teacher's Name : Md Romzan Alom

(Lecturer)

Dept. of CSE,GUB

Lab Report Status

Marks: Signature:.....

Comments:..... Date:.....

Contents

| | |
|---|---------------|
| 1 Introduction | 3 |
| 1.1 Overview | 3 |
| 1.2 Motivation | 3 |
| 1.3 Problem Definition | 3 |
| 1.3.1 Problem Statement | 3 |
| 1.4 Design Goals/Objectives | 3 |
| 1.5 Application | 4 |
| 2 Design/Development/Implementation of the Project | 5 |
| 2.1 Introduction | 5 |
| 2.2 Project Details | 5 |
| 2.2.1 Frontend Interface | 5 |
| 2.3 Implementation | 6 |
| 2.3.1 The Workflow | 6 |
| 2.3.2 Tools & Libraries..... | 6 |
| 3 Performance Evaluation | 8 |
| 3.1 Simulation Environment/ Simulation Procedure | 8 |
| 3.1.1 Backend Setup | 8 |
| 3.1.2 Frontend Simulation..... | 8 |
| 3.2 Results Analysis/Testing | 8 |
| 3.2.1 Source Code | 8 |
| 3.2.2 UI/UX | 8 |
| 3.3 Results Overall Discussion..... | 9 |
| 3.3.1 Complex Engineering Problem Discussion..... | 9 |
| 4 Conclusion | 10 |
| 4.1 Discussion | 10 |
| 4.2 Limitations | 10 |
| 4.3 Scope of Future Work | 10 |

Chapter 1

Introduction

1.1 Overview

The File System Management Simulation is a terminal-based utility developed using Bash shell scripting for Unix/Linux systems. It offers a menu-driven interface that allows users to perform common file operations such as creating, reading, writing, copying, moving, and deleting files. Additionally, users can manage directories by creating, listing, and removing them. The simulation also supports modifying file permissions, providing hands-on experience with Unix/Linux file system management. Designed for educational purposes, it helps users understand the intricacies of file operations and system administration tasks. The project enhances proficiency in Bash scripting and system management concepts. By interacting with this simulation, users can gain practical knowledge applicable to real-world Unix/Linux environments.

1.2 Motivation

The File System Management script offers a user-friendly, menu-driven interface for managing files and directories in a Unix/Linux environment. By utilizing a simple terminal-based menu, users can perform common tasks such as creating, deleting, and renaming files without needing to recall complex shell commands. This approach streamlines file management, making it accessible even for those with limited command-line experience. The script enhances productivity by automating routine operations and reducing the potential for errors. It's an invaluable tool for both beginners and seasoned users seeking efficient file system management.

1.3 Problem Definition

1.3.1 Problem Statement

Performing frequent file operations through CLI commands can be repetitive and error-prone, especially for novice users. There is a need for a script that streamlines these operations with prompts and guided input. Such a script can automate tasks like creating, deleting, and renaming files, reducing the likelihood of mistakes. By providing clear instructions and feedback, it enhances user experience and efficiency. This approach makes file management more accessible, even for those with limited command-line experience.

1.4 Design Goals/Objectives

The main objectives of a File System Management (FSM) include:

- Provide an interactive, easy-to-use CLI utility.
- Enable basic file operations such as create, delete, rename, and view.
- Support advanced features like file content sorting and extension-based filtering.
- Ensure input validation and display user-friendly messages.

1.5 Application

This script is a valuable tool for Linux users, particularly beginners, aiming to simplify file management tasks. It serves as an educational resource, aiding users in understanding and performing basic file operations through a guided, menu-driven interface. Additionally, the script proves beneficial for system administrators by automating routine file management tasks, thereby enhancing efficiency and reducing the likelihood of errors.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The File System Management Simulation is implemented as a Bash shell script that operates within a terminal environment. Upon execution, it presents a looped menu offering various file management options. Users can interactively select tasks such as creating, deleting, or modifying files and directories. The script continues to display the menu until the user chooses to exit, providing a straightforward and efficient method for managing file operations.

2.2 Project Details

2.2.1 Front-end Interface

The interface is a text-based menu presented to the user at runtime. The user inputs numbers corresponding to actions such as:

- i. Listing files
- ii. Creating .txt, .sh, or .c files
- iii. Viewing file content
- iv. Counting files/directories

2.3 Implementation

2.3.1 The Workflow

- i. The script begins with a while loop running until option 0 (Exit) is chosen.

- ii. Based on user input (read opt1), a case-like structure using if-elif determines the action.
- iii. Commands used: ls, touch, rm, mv, nano, cat, stat, sort, wc, and find.

2.3.1 Tools & Libraries

- i. **Bash Shell**
- ii. **nano** – text editor
- iii. **Vs code** – code editor
- iv. **Linux CLI Utilities** – ls, touch, rm, mv, cat, stat, etc.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

3.1.1 Backend setup

- i. **OS:** Ubuntu 20.04+
- ii. **Environment:** GNU Bash 5+
- iii. **Permissions:** Script should be made executable (chmod +x system.sh)

3.1.2 Frontend Simulation

- i. **Run via terminal:** ./system.sh
- ii. User follows on-screen menu for operations

3.2 Result Analysis/Testing

3.2.1 Source Code

Here some to the code of FMS:

```

$ system.sh x
File-System-Management-OS-Project-main > $ system.sh
1  #!/bin/bash
2  i="0"
3  while [ $i -lt 100 ]
4  do
5      echo "=====
6      echo "-----FILE SYSTEM MANAGEMENT-----"
7      echo "=====
8      echo " "
9      echo "1- List all files and Directories here"
10     echo "2- Create New Files"
11     echo "3- Delete Existing Files"
12     echo "4- Rename Files"
13     echo "5- Edit File Content"
14     echo "6- Search Files"
15     echo "7- Details of Particular File"
16     echo "8- View Content of File"
17     echo "9- Sort File Content"
18     echo "10- List only Directories(Folders)"
19     echo "11- List Files of Particular Extension"
20     echo "12- Count Number of Directories"
21     echo "13- Count Number of Files"
22     echo "14- Sort Files in a Directory"
23     echo "0- Exit"
24     echo " "

```

Fig 1 – FSM code

```

23  read opt1
24  if [ $opt1 == 1 ]
25  then
26      echo "List all files & Directories.."
27      echo "Showing all files & directories.."
28      sleep 4
29      echo "loading..."
30      echo "----- OutPut -----"
31      ls
32      echo " "
33
34  elif [ $opt1 == 2 ]
35  then
36      echo "Create New Files here.."echo "Which type of file you want to create !"
37      echo "1- .c"
38      echo "2- .sh"
39      echo "3- .txt"
40      echo "Enter your choice from 1 - 3"
41      read filechoice
42
43      if [ $filechoice == 1 ]
44      then
45          echo "Enter File Name without .c Extension"
46          read filename

```

Fig 2 – FSM code

```

136     echo "Delete existing files here.. "
137     echo "Enter name of File you want to Delete!"
138     echo "Note: Please Enter full Name with Extension."
139     read delfile
140     echo "-----OutPut-----"
141
142
143     if [ -f "$delfile" ];
144     then
145         rm $delfile
146         echo "Successfully Deleted."
147         echo " "
148     else
149         echo "File Does not Exist..Try again"
150         echo " "
151     fi
152
153 elif [ $opt1 == 7 ]
154 then
155     echo "Details of files.."
156     echo "Enter file name with extension.."
157     read detail
158     echo "-----output-----"
159     echo "Checking for file"
160     sleep 3
161

```

Fig 3 – FSM code

Output result:

```

0-Exit

5
Search files here..
Enter File Name with Extension to search
mkneel.py
-----OutPut-----
File Does not Exist..Try again.

=====

```

Fig 4 – FSM file search output


```
9
Sort files content here..
Enter file name with extension..
mkneel.txt
-----output-----
Sorting file content.

1 2 5 3 9 8 6
7 3 2 8 4 0 10
9 7 8 2 1 1 0
```

Fig 5 – FSM file sort output

3.2.2 UI/UX

- i. Clear menu presentation with separators
- ii. Delay (sleep) added for realism and better UX
- iii. User-friendly prompts and messages

```
project\File-System-Management-OS-Project> bash system.sh
=====
----- File System Management -----

1-List all files & Directories
2-Create New files
3-Rename Files
4-Modify File Content
5-Search Files
6-Remove Existing Files
7-Details of Files
8-View content of file here..
9-Sort files content here..
10-List of all directories..
11- List Files of Particular Extension
12- Count Number of Directories
0-Exit

█
```

Fig 6 – UI/UX

3.3 Results Overall Discussion

3.3.1 Complex Engineering Problem Discussion

Though simple in nature, combining multiple shell utilities and implementing robust condition handling mimics real-world engineering practices like:

- i. Input parsing
- ii. Command execution based on context
- iii. Iterative interaction design

Chapter 4

Conclusion

4.1 Discussion

This project successfully demonstrates how Bash scripting can automate and simplify common file system tasks. It provides a foundation for larger administrative tools.

4.2 Limitations

- i. Lacks graphical interface
- ii. Limited to current directory operations
- iii. No logging or permission checks

4.3 Scope of Future Work

The scope of future work for a File System Management Simulator includes:

- i. Add GUI using Zenity or dialog
- ii. Extend to support batch operations and nested directories
- iii. Implement logging and undo features
- iv. Integrate user authentication or file permission checks

Reference:

1. <https://www.gnu.org/software/bash/manual/bash.html>
2. <https://ryanstutorials.net/bash-scripting-tutorial/>
3. <https://www.geeksforgeeks.org/basic-linux-commands/>
4. <https://www.tutorialspoint.com/unix/unix-file-management.htm>