# Lab Report

| Only for course Teacher | | | | | | |
|---|---|---|---|---|---|---|
| | | Needs Improvement | Developing | Sufficient | Above Average | Total Mark |
| Allocate mark & Percentage | | 25% | 50% | 75% | 100% | 25 |
| Clarity | 2 | | | | | |
| Content Quality | 4 | | | | | |
| Spelling & Grammar | 2 | | | | | |
| Organization and Formatting | 2 | | | | | |
| | | | | | Total obtained mark | |
| Comments | | | | | | |

## Semester: Summer 25

**Student Name: Israt Jahan**

**Student ID: 232-35-063**

**Batch: 41** **Section: K2**

**Course Code: SE 233** **Course Name: Operating System &System Programming Lab**

**Course Teacher Name: Md.Mahbubur Rahman**

**Designation: Lecturer**

**Submission Date: 08/08/2025**

# File System Management Utility

# Abstract

This project aims to develop an interactive **File System Management Utility** using **Shell Scripting** that enables users to perform essential file operations directly from the terminal. The project provides functionalities such as listing files and directories, creating new files of different types, deleting and renaming files, editing content, searching, sorting file content, counting files and directories, and more. This tool simplifies file management tasks by providing a menu-driven interface, enhancing usability for system administrators and beginner-level users who prefer a command-line environment. The developed script successfully automates routine file operations, handles user errors, and ensures a user-friendly interaction loop.

## Table of Contents



# 1. Introduction

## 1.1. Background and Motivation

File management is a routine yet critical task in system administration. Traditional Linux command-line utilities require manual inputs and memorization of complex commands. For beginners, this can be error-prone and time-consuming. This project aims to bridge that gap by developing an **interactive shell-based file management system**, providing a guided menu-driven interface for performing common file tasks.

### 1.2. Problem Statement

System users, especially beginners, often lack a **simple interactive tool** for managing files via the command line. The absence of a guided interface can lead to **accidental data loss**, inefficient workflows, and redundant effort in executing routine tasks.

### 1.3. Purpose of the Project

The purpose of this project is to **develop a robust, user-friendly shell script** that automates common file management tasks (like create, delete, rename, edit, search, etc.) through a simple interactive interface. The tool will minimize human error and improve productivity.

## 2. Project Scope

### 2.1. In-Scope Features

- Interactive Menu for File Operations.
- File & Directory Listing.
- File Creation (C, SH, TXT).
- File Deletion & Renaming.
- File Content Editing & Viewing.
- Searching Files.
- Sorting Files & File Content.
- Counting Files & Directories.

### 2.2. Out-of-Scope Features

- Safe Delete with Recovery (Trash Concept).
- File Permission Management.
- Remote File Handling.

## 3. Objectives

- To design a bash script providing **14+ file management functionalities**.
- Ensure **interactive menus and input prompts** for user-friendliness.
- Handle invalid inputs and missing file errors gracefully.
- Provide **sorted views and statistical counts** of files and directories.

# 4. System Design and Architecture

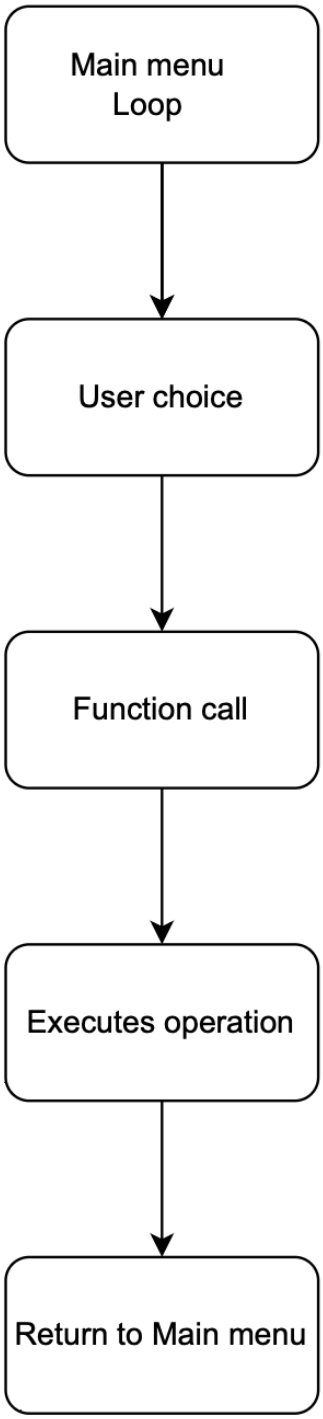
## 4.1. High-Level Design

The project follows a **Main Menu Loop**, displaying file operation options to the user. Upon selecting an option, corresponding functions are executed. After completion, control returns to the main menu until the user exits. Error handling is embedded within each function to ensure smooth operation.

## 4.2. Core Components

- `main_menu()`: Displays all user options and captures input.
- `list_files()`: Lists files and directories.
- `create_file()`: Creates .c, .sh, .txt files based on user choice.
- `delete_file()`: Deletes a file after checking existence.
- `rename_file()`: Renames files with old and new names.
- `edit_file()`: Opens files in `nano` for editing.
- `search_file()`: Searches for a file by name.
- `count_files_dirs()`: Counts files and directories.
- `sort_files()`: Sorts files alphabetically.

### 4.3. Data Flow


Main menu
Loop
User choice
Function call
Executes operation
Return to Main menu

# 5. Implementation Details

## 5.1. Core Functions and Logic

- **Main Loop**: A `while` loop that repeats until exit.
- **Case Selection**: A `case` block for handling menu options.
- **Error Handling**: Checks file existence using `[ -f "$file" ]` before operations.
- **Dynamic File Creation**: Uses `touch` with user-defined extensions.
- **Counting Files/Dirs**: Uses `ls -l` and `grep '^d'` or `grep '^-'`.

## 5.2. Key Shell Commands and Utilities Used

| Command | Usage |
|---|---|
| ls | List files and Directories |
| touch | Create new files |
| rm | Deletes files |
| mv | Rename files |
| cat | View file content |
| nano | Edit files |
| find | Search files |
| sort | Sort file content alphabetically |
| grep | Filter files or directories in counts |
| wc | Word count(used for counting files/dirs) |

## 5.3. Error Handling

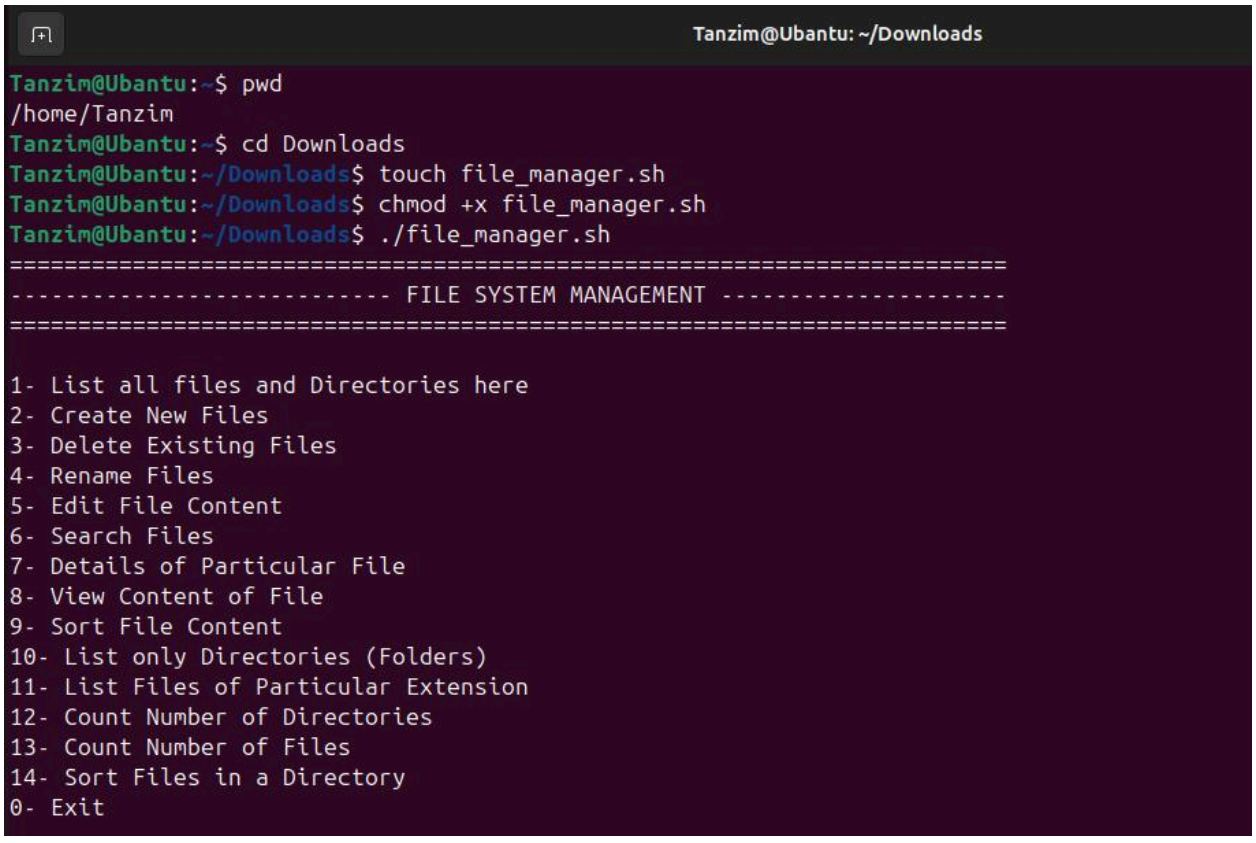
- Invalid option prompts with retry message.
- File existence checks before delete, edit, rename.
- Graceful exit using `exit` on option 0.
- Handles cases where no files of a particular extension exist.

# 6. Results and Discussion

## 6.1. Achievement of Objectives

| Objective | Outcome |
|---|---|
| Provide 14+ file operations | Fully Implemented |
| Interactive Menus | Implemented using read and case |
| Error Handling | All invalid inputs managed with proper messages |
| Counting and Sorting files | Successfully implemented using grep,wc,sort |

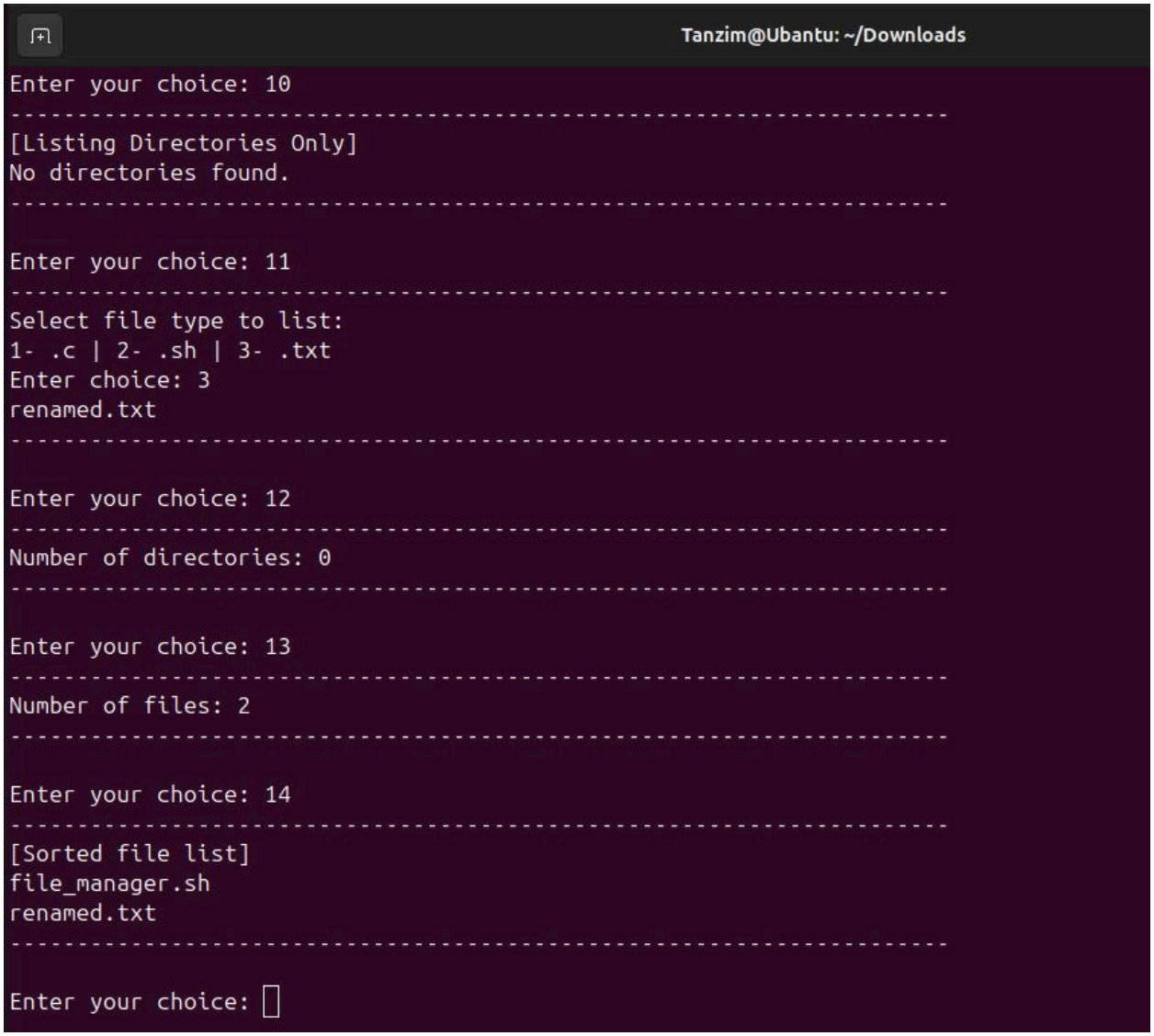
## 6.2. Screenshots and Usage Examples



```
Enter your choice: 1
------------------------------------------------------------------------
[Listing all files and directories]
total 4.0K
-rwxrwxr-x 1 Tanzim Tanzim 3.5K Aug  7 15:38 file_manager.sh
------------------------------------------------------------------------

Enter your choice: 2
------------------------------------------------------------------------
[Create New Files]
Select file type to create:
1- .c
2- .sh
3- .txt
Enter your choice (1-3): 1
Enter File Name: hello
hello.c created.
------------------------------------------------------------------------

Enter your choice: 2
------------------------------------------------------------------------
[Create New Files]
Select file type to create:
1- .c
2- .sh
3- .txt
Enter your choice (1-3): 3
Enter File Name: demo
demo.txt created.
------------------------------------------------------------------------
```

```
Enter your choice: 3
------------------------------------------------------------------------
Enter file name to delete: hello.c
Deleted hello.c.
------------------------------------------------------------------------

Enter your choice: 4
------------------------------------------------------------------------
Enter old file name: demo.txt
Enter new file name: renamed.txt
Renamed to renamed.txt
------------------------------------------------------------------------

Enter your choice: 5
------------------------------------------------------------------------
Enter file name to edit: renamed.txt
------------------------------------------------------------------------

Enter your choice: 6
------------------------------------------------------------------------
Enter file name to search: renamed.txt
[Searching for renamed.txt...]
./renamed.txt
------------------------------------------------------------------------
```

```
Enter your choice: 7
------------------------------------------------------------------------
Enter file name to view details: renamed.txt
  File: renamed.txt
  Size: 27              Blocks: 8          IO Block: 4096   regular file
Device: 8,2     Inode: 1312003     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  Tanzim)   Gid: ( 1000/  Tanzim)
Access: 2025-08-07 15:40:55.624375781 +0000
Modify: 2025-08-07 15:41:26.612480433 +0000
Change: 2025-08-07 15:41:26.612480433 +0000
 Birth: 2025-08-07 15:39:50.099146519 +0000
------------------------------------------------------------------------

Enter your choice: 8
------------------------------------------------------------------------
Enter file name to view content: renamed.txt
Hello,this is a test file.
------------------------------------------------------------------------

Enter your choice: 9
------------------------------------------------------------------------
Enter file name to sort content: renamed.txt
File content sorted and saved in renamed.txt.
------------------------------------------------------------------------

Enter your choice: 10
------------------------------------------------------------------------
[Listing Directories Only]
No directories found.
------------------------------------------------------------------------
```


Tanzim@Ubantu: ~/Downloads
Enter your choice: 10
[Listing Directories Only]
No directories found.
Enter your choice: 11
Select file type to list:
1- .c | 2- .sh | 3- .txt
Enter choice: 3
renamed.txt
Enter your choice: 12
Number of directories: 0
Enter your choice: 13
Number of files: 2
Enter your choice: 14
[Sorted file list]
file_manager.sh
renamed.txt
Enter your choice:


Enter your choice: 14
[Sorted file list]
file_manager.sh
renamed.txt
Enter your choice: 0
Good Bye!
Tanzim@Ubantu:~/Downloads$

# 7. Conclusion and Future Work

### 7.1. Conclusion

This project successfully developed a **shell-based interactive file management system**. It simplifies command-line operations, provides an intuitive user interface, and handles common file operations effectively. Through this project, a deeper understanding of **bash scripting, conditional structures, loops, and file handling commands** was gained.

### 7.2. Challenges Faced

- Handling invalid inputs dynamically without script termination.
- Managing file type filters when no such files existed.
- Keeping UI clean and responsive using minimal bash utilities.

### 7.3. Future Enhancements

- Implement a **safe delete (trash bin concept)**.
- Add **file permission management** features.
- Create a **GUI wrapper** using tools like `zenity`.
- Add **logging mechanisms** for all operations.

# 8. References


- Bash Manual Pages (`man bash`)
- GNU Core Utilities Documentation
- TutorialsPoint Bash Scripting Guide


# 9. Appendix

## 9.1. Full Source Code

```
#!/bin/bash

# Show the menu only once at the beginning
```

```
echo "================================================================================"

echo "---------------------------- FILE SYSTEM MANAGEMENT ----------------------"

echo "================================================================================"

echo

echo "1- List all files and Directories here"

echo "2- Create New Files"

echo "3- Delete Existing Files"

echo "4- Rename Files"

echo "5- Edit File Content"

echo "6- Search Files"

echo "7- Details of Particular File"

echo "8- View Content of File"

echo "9- Sort File Content"

echo "10- List only Directories (Folders)"

echo "11- List Files of Particular Extension"

echo "12- Count Number of Directories"

echo "13- Count Number of Files"

echo "14- Sort Files in a Directory"

echo "0- Exit"

echo


# Infinite loop for choice input
```

```
while true
do
	read -p "Enter your choice: " opt1
	echo "----------------------------------------------------------------------"

	case $opt1 in
	1)
		echo "[Listing all files and directories]"
		ls -lh
		;;
	2)
		echo "[Create New Files]"
		echo "Select file type to create:"
		echo "1- .c"
		echo "2- .sh"
		echo "3- .txt"
		read -p "Enter your choice (1-3): " filechoice
		case $filechoice in
			1) read -p "Enter File Name: " filename; touch "$filename.c"; echo "$filename.c created." ;;
			2) read -p "Enter File Name: " filename; touch "$filename.sh"; echo "$filename.sh created." ;;
			3) read -p "Enter File Name: " filename; touch "$filename.txt"; echo "$filename.txt created." ;;
			*) echo "Invalid input." ;;
```

```
esac
                ;;

        3)
                read -p "Enter file name to delete: " delfile
                [ -f "$delfile" ] && rm "$delfile" && echo "Deleted $delfile." || echo "File does not
exist."
                ;;

        4)
                read -p "Enter old file name: " old
                if [ -f "$old" ]; then
                        read -p "Enter new file name: " new
                        mv "$old" "$new"
                        echo "Renamed to $new"
                else
                        echo "File does not exist."
                fi
                ;;

        5)
                read -p "Enter file name to edit: " edit
                [ -f "$edit" ] && nano "$edit" || echo "File does not exist."
                ;;
```

```
6)
        read -p "Enter file name to search: " f
        echo "[Searching for $f...]"
        find . -name "$f"
        ;;

7)
        read -p "Enter file name to view details: " detail
        [ -f "$detail" ] && stat "$detail" || echo "File does not exist."
        ;;

8)
        read -p "Enter file name to view content: " readfile
        [ -f "$readfile" ] && cat "$readfile" || echo "File does not exist."
        ;;

9)
        read -p "Enter file name to sort content: " sortfile
        if [ -f "$sortfile" ]; then
                sort "$sortfile" -o "$sortfile"
                echo "File content sorted and saved in $sortfile."
        else
                echo "File does not exist."
        fi
```

```
        ;;

10)
        echo "[Listing Directories Only]"
        ls -d */ 2>/dev/null || echo "No directories found."
        ;;

11)
        echo "Select file type to list:"
        echo "1- .c | 2- .sh | 3- .txt"
        read -p "Enter choice: " extopt
        case $extopt in
                1) ls *.c 2>/dev/null || echo "No .c files found." ;;
                2) ls *.sh 2>/dev/null || echo "No .sh files found." ;;
                3) ls *.txt 2>/dev/null || echo "No .txt files found." ;;
                *) echo "Invalid input." ;;
        esac
        ;;

12)
        count=$(ls -l | grep '^d' | wc -l)
        echo "Number of directories: $count"
        ;;
```

```
        13)
                count=$(ls -l | grep '^-' | wc -l)
                echo "Number of files: $count"
                ;;

        14)
                echo "[Sorted file list]"
                ls | sort
                ;;

        0)
                echo "Good Bye!"
                exit 0
                ;;

        *)
                echo "Invalid input. Try again."
                ;;
        esac

        echo "------------------------------------------------------------------"
        echo
done
```