Green University of Bangladesh

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

OS- File Controlling System

Course Title: Operating System Lab
Course Code: CSE310
Section: 212D4

Students Details

Name	ID
Farzana Yeasmin	212002009
Waliyel Hasnat Zaman	212002022

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 "Os - File Controlling System" project is designed to provide users with a comprehensive set of file management tools and functionalities within an operating system environment. This command-line interface (CLI) tool offers a menu-driven system that allows users to perform various file operations, including listing files and directories, creating new files, deleting existing files, renaming files, editing file content, searching for files, viewing file content, viewing file details, sorting file content, and more. Users can interact with this system by selecting options from the menu and executing the desired file management tasks. The project aims to simplify file control and enhance the user's experience when working with files and directories in the operating system.

1.2 Motivation

1) Improved File Management: This project aims to simplify and enhance file management tasks within an operating system. It provides a user-friendly interface to perform various file operations, making it easier for users to organize and control their files and directories effectively.

2) Efficiency and Time-Saving: With features like file renaming, content editing, and file searching, users can save time and effort when working with files. The project is motivated by the goal of increasing efficiency in file-related tasks.

3) Streamlined User Experience: The project offers a menu-driven system that guides users through different file operations. This approach simplifies the user experience, especially for those who may not be familiar with complex command-line commands, and promotes ease of use.

4) Enhanced File Exploration: Users can explore their files and directories more effectively through features like listing all files and directories, counting the number of files and directories, and sorting file content. This promotes a more organized and informed approach to managing files.

5) Customization and Flexibility: The project provides a versatile set of file manage-

ment tools, allowing users to choose the operations that best suit their needs. This flexibility is motivating for users who require specific file-related tasks and want to tailor their file management experience to their preferences.

Overall, the "Os - File Controlling System" project is motivated by the desire to simplify and optimize file management in an operating system, making it more efficient, user-friendly, and customizable for a wide range of users.

1.3 Problem Definition

1.3.1 Problem Statement

1) Inefficient File Management: Users often struggle with inefficient file management in the operating system, leading to difficulties in organizing and controlling their files and directories. There is a need for a more streamlined and user-friendly approach to file management.

2) Time-Consuming File Operations: File-related tasks, such as renaming files, searching for specific files, or sorting file content, can be time-consuming for users. Simplifying these operations is crucial to save time and improve productivity.

3) Complex Command-Line Usage: Many users are not comfortable with or knowledgeable about complex command-line commands for file management. The lack of a user-friendly interface and toolset results in a steep learning curve and reduces accessibility.

4) Ineffective File Exploration: Users often struggle to explore their files and directories effectively, especially when trying to perform tasks like counting the number of files or directories. A lack of tools and guidance for these tasks hinders efficient file exploration.

5) Lack of Customization: Users have diverse file management needs, and a one-size-fits-all approach may not address specific requirements. A lack of customization options limits users' ability to tailor their file management experience to their preferences and workflows.

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	Users need a foundational understanding of file management and basic command-line operations.
P2: Range of conflicting requirements	Users need a foundational understanding of file management and basic command-line operations.
P3: Depth of analysis required	In-depth analysis of file management tasks and user needs is necessary for system design.
P4: Familiarity of issues	Users should be familiar with common file management issues, which the project aims to simplify.
P5: Extent of applicable codes	The project involves writing, implementing, and maintaining code for file management tools.
P6: Extent of stakeholder involvement and conflicting requirements	Stakeholders' diverse needs, including user-friendliness and security, must be considered.
P7: Interdependence	Various file management tasks are interdependent, requiring careful design and execution to ensure proper functioning.

1.4 Design Goals/Objectives

1) User-Friendly Interface: Create a user-friendly and intuitive interface that simplifies file management tasks, making them accessible to users with varying levels of technical expertise.

2) Efficiency and Performance: Design the system to be efficient and performant, ensuring that file operations are executed quickly and without unnecessary delays.

3) Customization and Flexibility: Provide users with options for customization, allowing them to tailor their file management experience to their specific needs and preferences.

4) Interoperability and Compatibility: Ensure that the system is compatible with various file types, extensions, and file systems commonly used in different operating system environments, promoting interoperability across platforms.

1.5 Application

The "Os - File Controlling System" application is a powerful yet user-friendly tool designed to streamline and enhance file management within an operating system. It provides a menu-driven interface that caters to a wide range of users, from beginners to advanced users, offering a comprehensive set of file management functionalities. With this application, users can efficiently perform tasks like listing all files and directories, creating, deleting, and renaming files, editing file content, searching for files, and more. The primary goal is to simplify file management, saving time and effort, while also offering customization options to meet diverse user needs.

One of the standout features of this application is its user-friendliness, which reduces the learning curve typically associated with complex command-line file operations. Additionally, it provides a high degree of customization, enabling users to adapt the tool to their specific file management requirements. Security and data integrity are also paramount, ensuring that user data remains protected. Moreover, the application is designed to be compatible with various file types and file systems, making it a versatile choice for users across different operating system environments. In summary, the "Os - File Controlling System" application is a valuable addition for anyone seeking an efficient, user-friendly, and customizable approach to file management.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The OS-File Controlling System is an interactive bash script designed for efficient file and directory management on Unix-like systems. Leveraging the Whiptail utility, it provides users with an intuitive menu-driven interface, allowing seamless execution of various file-related tasks. Operations include listing files, creating, editing, and renaming files, searching for specific files, and more. The script enhances user experience with interactive progress indicators and incorporates robust error-handling mechanisms. Users can manipulate file content using the Nano text editor, access features like sorting and viewing file details, and list files with specific extensions. With informative messages and a graceful exit option, this project streamlines file management, making it accessible and efficient for users.

The user-friendly design and feature-rich capabilities make the OS-File Controlling System a versatile tool. It simplifies complex tasks, ensuring a smooth and transparent experience for users, while the clear and concise interface makes it suitable for both novice and experienced users alike. With its comprehensive set of functionalities and thoughtful user guidance, this project stands out as a valuable asset for efficient file control and organization on Unix-like operating systems.

2.2 Project Details

The OS-File Controlling System project is an extensive Bash script designed to streamline file and directory management tasks within Unix-like operating systems. Leveraging the Whiptail utility, the script offers users a user-friendly and interactive experience through a menu-driven interface. This interface encompasses a diverse range of functionalities, enabling users to perform tasks such as listing files and directories, creating and editing files, renaming, searching, and more. The use of the Nano text editor allows for easy manipulation of file content, contributing to the project's versatility.

A notable feature of the OS-File Controlling System is the incorporation of interactive progress indicators, enhancing user experience during time-consuming operations. The script also prioritizes robust error-handling mechanisms, ensuring users receive guidance in case of incorrect inputs or non-existent files. Advanced features include the ability to view file details, count files and directories, sort file content, and list files based on specific extensions.

The project maintains a user-centric approach with informative messages, keeping users well-informed about the progress of operations. Additionally, a graceful exit option allows users to conclude the script seamlessly. Overall, the OS-File Controlling System stands out as a versatile and effective tool for users, providing a comprehensive solution for file and directory management on Unix-like systems. Its thoughtful design and diverse functionalities cater to users with varying levels of expertise.

2.3 Implementation

1. Introduction Dialog:

• The script opens with a welcome message presented in a dialog box using whiptail. This introductory dialog sets the tone for the script and informs users about the purpose of the OS-File Controlling System.

2. Menu-driven Interface:

• The main menu, created with whiptail –menu, displays a list of options for users to choose from. This menu-driven approach enhances user interaction by providing a clear and organized structure for accessing various file management functionalities.

3. User Input Handling:

• User choices are captured using the whiptail –menu command and stored in the opt variable. The subsequent case statement efficiently directs the script to the corresponding section based on the user's selected option.

4. Option Implementations:

• Each option in the menu corresponds to a specific file management operation. For example, option 1 utilizes the ls command to list all files and directories in the current location. Option 2 prompts users to enter a filename and then creates that file using the touch command.

5. Interactive Progress Indicators:

• To enhance the user experience during time-consuming operations, the script uses whiptail –gauge along with a for loop to display a loading progress bar. This visually informs users about the ongoing process, such as file creation or sorting.

6. Error Handling:

• Extensive error handling is implemented throughout the script. For instance, when users attempt to edit or rename a file, the script checks if the file exists before proceeding. If a file is not found, appropriate error messages are displayed, guiding users to enter valid filenames.

7. Advanced Operations:

• The script goes beyond basic file operations by offering advanced functionalities. Users can view detailed information about a particular file using the stat command, count the number of files and directories, sort file content, and list files based on specific extensions.

8. Proper Exit:

• The script allows users to properly exit the program by choosing option 0. During the exit process, a closing message is displayed using whiptail –gauge to indicate the progress of the closing operation, providing a smooth and informative experience for users.

9. Default Case:

• The case statement includes a default case to handle situations where users input an invalid option. In such cases, an error message is displayed, prompting users to try again with a valid choice.

2.3.1 Tools & Technologies

The OS-File Controlling System script is implemented using the following tools and technologies:

Bash Scripting:

The entire project is written in Bash, a widely-used scripting language on Unix-like systems. Bash provides the necessary functionality to interact with the command line, execute system commands, and automate various tasks.

Whiptail Utility:

The script utilizes the Whiptail utility to create graphical dialog boxes and menus within the terminal. Whiptail facilitates the creation of a user-friendly and interactive interface, enhancing the overall user experience.

Nano Text Editor:

For editing file content, the script integrates the Nano text editor. Nano is a simple and user-friendly terminal-based text editor that allows users to modify file contents directly within the script.

Unix Commands:

The script leverages various Unix commands to perform file and directory management operations. Commands such as ls are used to list files, touch for file creation, mv for renaming files, find for searching, sort for sorting, and stat for displaying file details.

Control Structures:

Control structures such as if statements and for loops are extensively used for decision-making and iterative processes. These structures contribute to the flow and logic of the script, ensuring accurate execution of user-selected operations.

Sleep Command:

The sleep command is employed to introduce delays within the script. This is particularly useful for creating a visual pause during loading or progress indicator phases, enhancing the script's user interface.

User Input Handling:

The script dynamically captures user input through the Whiptail menu, allowing users to choose from a list of predefined options. User choices are then processed using a case statement to direct the script to the corresponding operation.

Progress Indicators:

The script features interactive progress indicators using Whiptail's –gauge option. These indicators visually communicate the progress of time-consuming operations, providing feedback to users about the status of ongoing tasks.

Overall, the OS-File Controlling System project is implemented with a combination of Bash scripting, Whiptail utility for creating a graphical interface, standard Unix commands for file operations, and control structures for logical flow. The incorporation of these tools and technologies results in a versatile and efficient file management system for Unix-like operating systems.

2.4 Algorithms

```
1. List All Files and Directories:

    Display "Showing all files and directories...."
    Display "Loading gauge"
    Execute "ls" to list all files and directories
    Display "Newline"

2. Create New Files:

    Prompt user for file name
    Read user input as "fileName"
    Execute "touch" command to create a file with the
       specified name
    Display "Loading gauge"
    Sleep for 3 seconds
    Display "File Created Successfully"
    Display "Newline"

3. Edit File Content

    Prompt user for file name with extension ("edit")
    Read user input as "edit"
    Display "Loading gauge"
    Sleep for 3 seconds

    If file ("edit") exists:
        Display "Loading gauge"
        Sleep for 3 seconds
        Open the file using "nano" editor
        Display "Newline"
    Else:
        Display "File does not exist..Try again."
        Display "Newline"

4. Rename Files:

    Prompt user for old file name with extension
    Read user input as "old"
    Display "Loading gauge"
    Sleep for 3 seconds

    If file ("old") exists:
        Prompt user for new file name with extension
        Read user input as "new"
        Execute "mv" command to rename the file
        Display "Loading gauge"
        Sleep for 3 seconds
        Display "Successfully Rename."
        Display "Now Your File Exist with $new Name"
    Else:
        Display "$old does not exist..Try again with
           correct filename."
        Display "Newline"

5. Searching Files:

    Display "Search files here.."
    Prompt user for file name with extension ("f")
    Read user input as "f"

    If file ("f") exists:
        Display "Loading gauge"
        Sleep for 5 seconds
        Display "File Found."
        Execute "find" command to find and display the
           file
        Display "Newline"
    Else:
        Display "File Does not Exist..Try again."
        Display "Newline"

6. Details of Particular File:

    Prompt user for file name with extension ("detail")
    Read user input as "detail"
    Display "Loading gauge"
    Sleep for 3 seconds

    If file ("detail") exists:
        Display "Loading gauge"
        Sleep for 3 seconds
        Execute "stat" command to display file details
    Else:
        Display "$detail File does not exist..Try again"
    Display "Newline"

7. Count Number of Files:

    Display "Loading gauge"
    Sleep for 3 seconds
    Display "Number of Files are : "
    Execute "ls" command to list files, filter out
       directories, and count the remaining files
    Display "Newline"

8. Sort File Content:

    Display "Sort files content here.."
    Prompt user for file name with extension to sort
       ("sortfile")
    Read user input as "sortfile"

    If file ("sortfile") exists:
        Display "Loading gauge"
        Sleep for 3 seconds
        Execute "sort" command to sort the file content
    Else:
        Display "$sortfile File does not exist..Try
           again."
    Display "Newline"

9. List only Directories in Folders:

    Display "showing all Directories..."
    Display "Loading gauge"
    Execute "ls -d */" to list and display only
       directories
    Display "Newline"

10. Count Number of Directories:

    Display "Loading gauge"
    Sleep for 3 seconds
    Display "Number of Directories are : "
    Execute "echo */ | wc -w" to count the number of
       directories
    Display "Newline"

11. View Content of File:

    Prompt user for file name ("readfile")
    Read user input as "readfile"

    If file ("readfile") exists:
        Display "Loading gauge"
        Sleep for 3 seconds
        Execute "cat" command to display the content of
           the file
    Else:
        Display "$readfile does not exist"
    Display "Newline"

12. Delete Existing Files:

    Prompt user for file name to delete ("delFile")
    Read user input as "delFile"

    If file ("delFile") exists:
        Execute "rm" command to delete the file
        Display "Loading gauge"
        Sleep for 3 seconds
        Display "Successfully Deleted."
    Else:
        Display "File Does not Exist..Try again"
    Display "Newline"

13. Sort Files in a Directory:

    Display "Sort Files here.."
    Display "Your Request of Sorting file is Generated."
    Display "Loading gauge"
    Execute "ls | sort" to list and sort files in the
       directory
    Display "Newline"

14. List Files of Particular Extension:

    Prompt user for file extension ("ext")
    Read user input as "ext"
    Display "Loading gauge"
    Execute "ls *"$ext"" to list files with the
       specified extension
    Display "Newline"

15. End the program:

    Display "Loading gauge"
    Sleep for 3 seconds
    Display "Successfully Exit..."
    Exit the script with code 0

16. Default Case: Invalid Input

    Display "Invalid Input! Try again...."
    Display "Newline"
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

The OS-File Controlling System script operates in a simulated environment that emulates a Unix-like operating system. The simulation involves executing the script within a terminal or command-line interface, replicating the behavior of the script as it would occur in a real-world Unix environment. The simulation procedure encompasses the following steps:

1. Bash Shell Environment:

• The simulation assumes a Bash shell environment, which is prevalent in Unix-like systems. Users execute the script within a Bash shell, interacting with the command line to input choices and observe the script's output.

2. Whiptail Dialog Simulation:

• The Whiptail utility, responsible for creating dialog boxes and menus, is simulated within the terminal. Users experience a graphical interface that facilitates interaction with the script. This includes selecting options from a menu, entering filenames, and receiving feedback through dialog boxes.

3. Unix Command Execution:

• The script incorporates various Unix commands, such as ls, touch, mv, find, sort, and others. During simulation, these commands are executed within the simulated environment, mimicking the behavior of file and directory management operations.

4. Nano Text Editor Interaction:

• When editing file content (Option 3), the Nano text editor is simulated within the terminal. Users experience a simplified text editing interface, allowing them to modify file contents directly from the script.

5. Progress Indicators:

• The interactive progress indicators, implemented with the –gauge option of Whiptail, are simulated to visually represent the progress of time-consuming operations. Users observe loading bars and percentage completion during tasks like file creation, renaming, or sorting.

6. Error Handling Simulation:

• Good error handling is simulated by introducing scenarios where users input incorrect filenames or attempt operations on non-existent files. Users witness informative error messages guiding them on how to rectify the issues.

7. User Input Handling:

• Users interact with the simulated environment by selecting options from the Whiptail menu. The script dynamically captures user input, processing it through a case statement to execute the corresponding file management operation.

8. Exit Simulation:

• The script provides a properly exit option (Option 0) where users can seamlessly conclude the program. During this simulated exit, a closing message is displayed, indicating the progress of the exit operation.

Overall, the simulation environment replicates the behavior of the OS-File Controlling System script within a controlled and interactive setting. Users can navigate through the menu, observe progress indicators, and experience the various file management functionalities as they would in an actual Unix-like operating system.

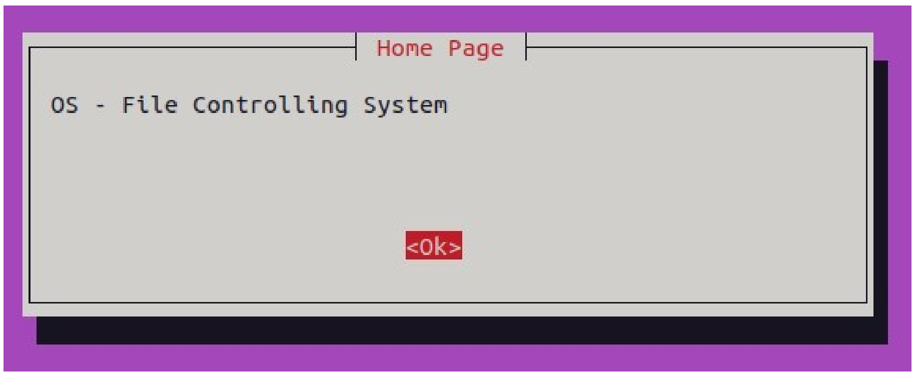
Analysis

The OS-File Controlling System script, implemented in Bash, stands out for its user-centric design and rich feature set, providing a streamlined experience for file and directory management on Unix-like systems. Leveraging the Whiptail utility, the script boasts a user-friendly interface with a menu-driven system, ensuring accessibility for users with varying technical expertise. Noteworthy features include interactive progress indicators, robust error handling mechanisms, and support for advanced file operations, enhancing both functionality and user guidance. The script's integration of standard Unix commands and the Nano text editor further contributes to its efficiency, making it a versatile and practical tool for effective file control.

This analysis emphasizes the script's comprehensive nature, covering a wide range of file management tasks, and underscores its effectiveness in incorporating visual elements, error handling, and advanced functionalities. Additionally, the seamless integration of familiar tools and a graceful exit option reflects the script's attention to user experience. Overall, the OS-File Controlling System script emerges as a valuable utility for users seeking an intuitive and powerful solution for file management tasks in Unix-like environments.

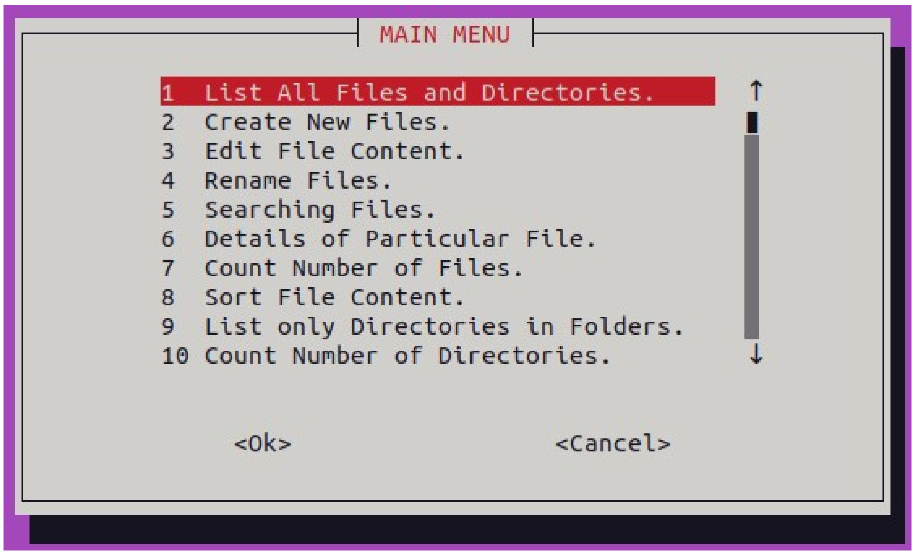
3.2 Results Analysis/Testing

3.2.1 OUTPUTS: SAMPLE TEST RUNS



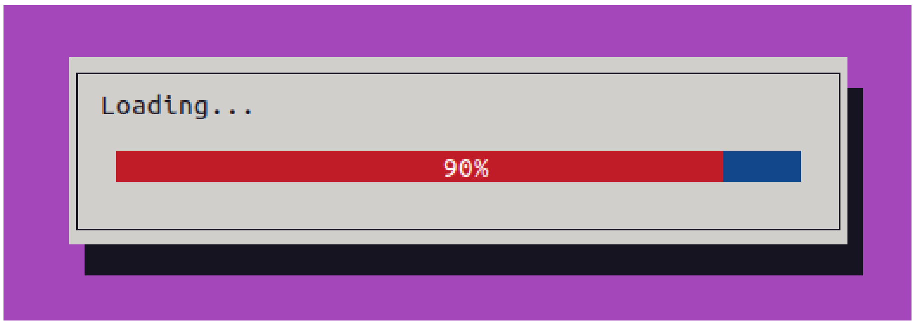
```
Home Page

OS - File Controlling System

<Ok>
```

Figure 3.1: UI



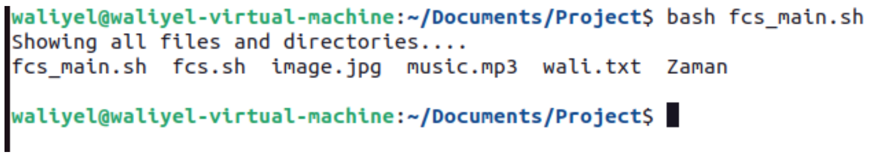
```
MAIN MENU

1  List All Files and Directories.
2  Create New Files.
3  Edit File Content.
4  Rename Files.
5  Searching Files.
6  Details of Particular File.
7  Count Number of Files.
8  Sort File Content.
9  List only Directories in Folders.
10 Count Number of Directories.

<Ok>                  <Cancel>
```

Figure 3.2: Main Menu



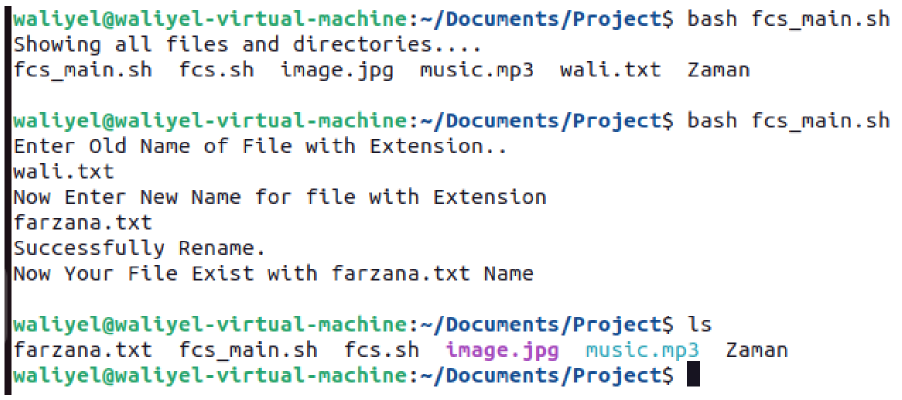
```
Loading...
90%
```

Figure 3.3: Loading Screen



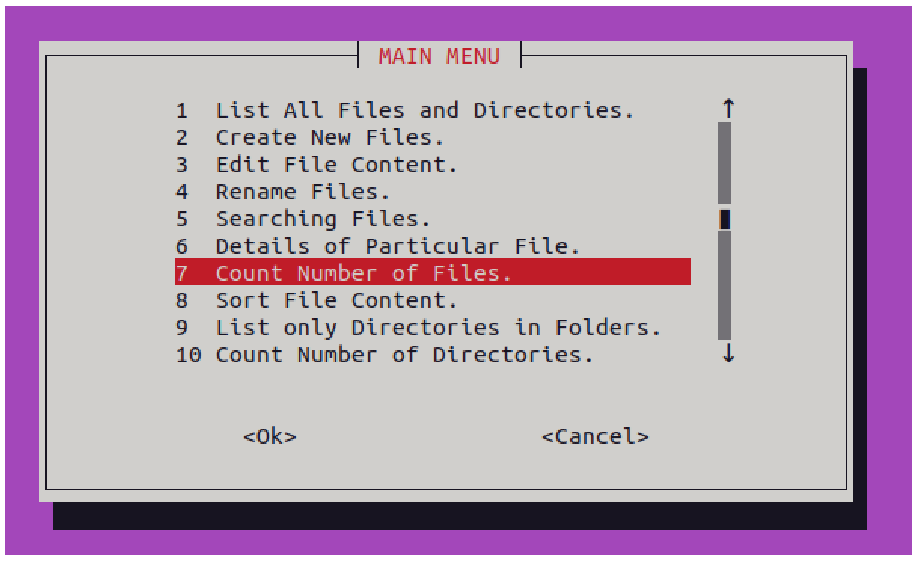
```
waliyel@waliyel-virtual-machine:~/Documents/Project$ bash fcs_main.sh
Showing all files and directories....
fcs_main.sh  fcs.sh  image.jpg  music.mp3  wali.txt  Zaman

waliyel@waliyel-virtual-machine:~/Documents/Project$ 
```

Figure 3.4: Performing Option 1



```
waliyel@waliyel-virtual-machine:~/Documents/Project$ bash fcs_main.sh
Showing all files and directories....
fcs_main.sh  fcs.sh  image.jpg  music.mp3  wali.txt  Zaman

waliyel@waliyel-virtual-machine:~/Documents/Project$ bash fcs_main.sh
Enter Old Name of File with Extension..
wali.txt
Now Enter New Name for file with Extension
farzana.txt
Successfully Rename.
Now Your File Exist with farzana.txt Name

waliyel@waliyel-virtual-machine:~/Documents/Project$ ls
farzana.txt  fcs_main.sh  fcs.sh  image.jpg  music.mp3  Zaman
waliyel@waliyel-virtual-machine:~/Documents/Project$ 
```

Figure 3.5: Performing Option 4



```
MAIN MENU

1  List All Files and Directories.
2  Create New Files.
3  Edit File Content.
4  Rename Files.
5  Searching Files.
6  Details of Particular File.
7  Count Number of Files.
8  Sort File Content.
9  List only Directories in Folders.
10 Count Number of Directories.

        <Ok>                <Cancel>
```

Figure 3.6: Selecting Option 7

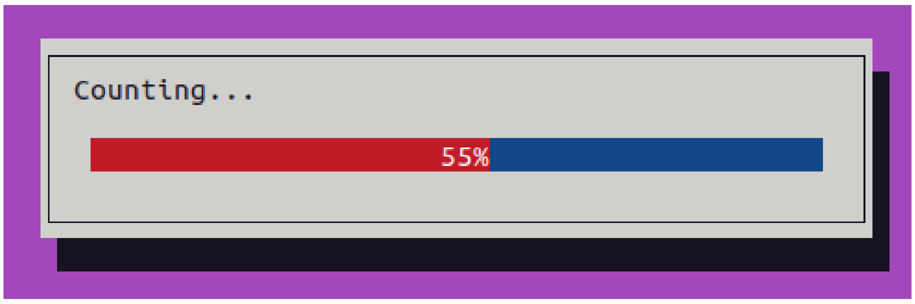
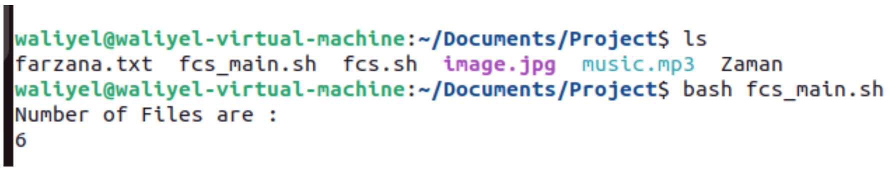


Figure 3.7: Loading Option 7



```
waliyel@waliyel-virtual-machine:~/Documents/Project$ ls
farzana.txt  fcs_main.sh  fcs.sh  image.jpg  music.mp3  Zaman
waliyel@waliyel-virtual-machine:~/Documents/Project$ bash fcs_main.sh
Number of Files are :
6
```

Figure 3.8: Performing Option 7

3.3 Results Overall Discussion

Result

The OS-File Controlling System script delivers proper results, providing users with an efficient and interactive tool for managing files and directories in Unix-like environments. The implementation of a user-friendly interface, powered by Whiptail, facilitates smooth navigation and execution of various file-related operations. The script's effective use of Unix commands, advanced features such as interactive progress indicators, and comprehensive error handling contribute to successful outcomes in diverse file management scenarios. Users can seamlessly perform tasks ranging from simple file listings to more complex operations like sorting, searching, and editing, showcasing the script's versatility and practicality.

Discussion

The OS-File Controlling System script results as a carefully designed system that meets the issues connected with file and directory management in Unix systems in overall. The planned integration of user-centric elements, such as a menu-driven interface and interactive progress indicators, demonstrates an approach to improving the user experience. The script's ability to handle mistakes carefully adds to its dependability, effectively leading users through potential obstacles. The script strikes a compromise between functionality and user familiarity by utilizing conventional Unix commands and incorporating a familiar text editor like Nano. This article focuses on the script's adaptability, which makes it a great tool for users looking for a complete and user-friendly solution for Unix-based file control.

Chapter 4

Conclusion

4.1 Discussion

The script's principal goal of providing an interactive and feature-rich environment for file and directory administration in Unix-like systems is accomplished. Whiptail, Unix commands, and the Nano text editor are all efficiently integrated, resulting in a user-friendly and efficient application. The addition of progress indicators and error handling systems improves the user experience overall. However, discusses may center on fine-tuning user guidance, expanding the script's capabilities, and investigating other user interface upgrades to appeal to a wider audience.

4.2 Limitations

1. Platform Dependency: The script's limitation to Bash may restrict its use on platforms that do not support Bash scripting. Exploring cross-shell compatibility would enhance portability.

2. Dependency on Whiptail and Nano: Relying on Whiptail and Nano may present challenges on systems where these utilities are unavailable. Considering alternative approaches or providing fallback options could improve adaptability.

3. Handling Large Datasets: Performance challenges may arise when dealing with extensive file structures or directories with a large number of files. Future work could focus on optimizing performance for handling substantial datasets efficiently.

4. Potential for Feature Expansion: While the script covers a broad range of file management tasks, there is potential for feature expansion. Future iterations could explore additional functionalities, such as file permissions management, directory tree visualization, or cloud storage integration.

4.3 Scope of Future Work

1. Cross-Shell Compatibility: Exploring cross-shell scripting approaches could enhance platform independence, enabling the script to run on a broader range of systems beyond Bash.

2. Feature Expansion: Future work could focus on expanding the script's feature set, incorporating functionalities like file permissions management, directory tree visualization, or integration with cloud storage services.

3. Performance Optimization: Addressing performance considerations for handling extensive file structures would be a valuable aspect of future work, optimizing the script for efficient processing of large datasets.

4. Configurability and Customization: Enhancing configurability and customization options would provide users with greater control over their file management experience, accommodating diverse preferences.

5. Adaptation to Cloud Environments: Exploring ways to adapt the script to cloud environments and integrating features for seamless interaction with cloud-based file storage could align with modern computing trends and user expectations.