Project Report
On
File Management System (Linux)

MASTERS OF COMPUTER APPLICATIONS

Submitted By:

Aman Kumar
(24MCA20154)

Project Guide:

Mr. Navdeep Singh
MCA Dept., (CU)

DEPARTMENT OF COMPUTER APPLICATION,
CHANDIGARH UNIVERSITY,
(NH05, Chandigarh-Ludhiana Highway, Gharuan, Mohali, Punjab, India)

SESSION 2024-26

DECLARATION

I, Aman Kumar, hereby declare that this project report titled "*File Management System*" is original work carried out by me under the supervision of Mr. Navdeep Singh. I further declare that this work has not been submitted to any other institute/university for the award of the degree of Master of Computer Applications.

Student Name: **Aman Kumar**
Roll No: **24MCA20154**

ACKNOWLEDGEMENT

I express my sincere gratitude to my project guide, Mr. Navdeep Singh, for invaluable guidance and support throughout this project. I also extend my thanks to Chandigarh University for the opportunity to undertake this project and to my classmates and family for their continuous encouragement.

Aman Kumar
24MCA20154

TABLE OF CONTENTS

1. Introduction

1.1 Scope of the System

The File Management System in Linux is a tool designed to simplify and automate essential file and directory operations in a Linux environment. It offers users a user-friendly, menu-driven interface for managing permissions, ownership, and general file tasks such as creating, deleting, and listing files and directories. This system is particularly useful for system administrators, developers, and Linux users who frequently interact with the file system. By consolidating these functions into a single script, the system provides a streamlined approach to managing files and directories, enhancing user productivity and reducing errors associated with manual command-line input.

1.2 Project Description

The primary objective of this project is to create an efficient File Permission Manager that makes it easier for users to interact with the file system. By leveraging a bash script, this project provides a simple, menu-driven interface to perform tasks that otherwise require specific knowledge of Linux commands and syntax.

1.2.1 About Existing System

In the traditional Linux environment, users interact with files and directories through individual commands such as `chmod` for permissions, `chown` for ownership, `ls` for listing, and more. While effective, this approach requires users to remember various command syntaxes, which can be error-prone, especially for beginners. Mistakes in these commands could lead to issues such as incorrect permissions, data loss, or even security vulnerabilities. Furthermore, the existing system does not provide a consolidated, user-friendly interface that integrates all these functions, making it inefficient for users managing multiple files and directories.

1.2.2 Implementation of Proposed System

The proposed system is implemented using a bash script that offers a menu-based approach. The script includes multiple functions, each corresponding to a specific file management task, such as changing permissions, modifying ownership, checking permissions, and creating or removing files and directories. This design simplifies the user's interaction with the file system, as they can perform all necessary tasks from a single interface without needing to remember complex command structures.

1.3 Advantages of the Project

- **User-Friendly Interface**: The menu-driven design allows users to navigate through options effortlessly, making file management accessible to users with minimal Linux experience.
- **Error Reduction**: By standardizing the command structure within functions, the system reduces the likelihood of user errors in command syntax.
- **Efficiency**: The integrated menu allows users to complete tasks more quickly compared to manually entering each command, saving time.

2. Project Category Tools & Environment

2.1 Project Category

This project falls under system utilities, specifically focusing on file and directory management in Linux. It provides a utility that enhances Linux's built-in command-line tools by presenting them in a user-friendly format.

2.2 Front-End Coverage

The front-end of the project is the command-line interface, which presents a menu listing various options. The menu enables users to select actions like changing permissions or creating files, each of which calls corresponding functions.

2.3 Back-End Coverage

The back-end of the project is implemented as a series of functions within a bash script. Each function executes Linux commands to perform tasks based on user input, such as changing file permissions with `chmod`, setting ownership with `chown`, and listing contents with `ls`.

2.4 Software and Hardware Requirements

Software:
OS: CentOS 7/8
Programming Language: Python 3.x
IDE: Visual Studio Code / PyCharm
Libraries: PyQt5 (for future GUI implementation)

Hardware:
Processor: Intel i5 or higher
RAM: 4 GB minimum
Disk Space: 50 GB minimum

3. Project Development Stages

3.1 Recognition of Needs

The need for this project stems from the complexities associated with Linux command-line file management. System administrators and users often require a simple way to manage file permissions and ownership. This project fulfills that need by providing an easy-to-use tool.

3.2 Feasibility Study

The feasibility analysis explored the effectiveness of a bash script for implementing the File Management System. Due to bash's robust file handling capabilities and compatibility with various Linux distributions, the bash script was found to be an efficient choice for building this system. Bash also offers the flexibility to extend functionality in future versions of the project.

3.3 System Analysis

Data Flow Diagram (DFD)

The DFD outlines the flow of data within the system:

1. **Level 0 (Context Level)**: The user interacts with the system through a single interface (menu), providing input that flows into different sub-functions.
2. **Level 1 (Detailed DFD)**: Each option in the menu represents a function call (e.g., changing permissions or creating a file), which processes user inputs and generates outputs like success/failure messages.

Entity-Relationship (ER) Diagram

The ER diagram shows the relationship between entities:

- **User**: The primary entity, responsible for initiating commands.
- **File/Directory**: These entities represent the objects managed by the system. Each file or directory has attributes like permissions and ownership.

3.4 Structure Charts, Data Tables

Structure Charts

- **Main Menu**: The primary menu displays options and interacts with the user.
- **Sub-Functions**: Each menu option links to specific functions (e.g., `change_permissions`, `create_file`), which execute the respective file management commands.

Data Tables

- **File Paths**: Store the user-specified path for files and directories.

- **Permissions and Ownership Data**: Store permissions and ownership details input by the user for file management tasks.

3.5 System Development

The system was developed by first defining core functions such as `change_permissions`, `change_ownership`, `check_permissions`, and `create_file`. These functions were then integrated into a loop-driven menu system. Error handling was incorporated to manage incorrect inputs and ensure smooth execution of commands.

3.6 Testing, Implementation & Maintenance

- **Testing**: Each function was tested independently on various Linux distributions, ensuring compatibility and reliability. For example, the `change_permissions` function was tested with different permission levels to confirm its functionality.
- **Implementation**: The system is deployed as a bash script, executable directly from the Linux terminal.
- **Maintenance**: Maintenance involves reviewing and updating the script for compatibility with newer Linux utilities, adding user-requested features, or fixing bugs.

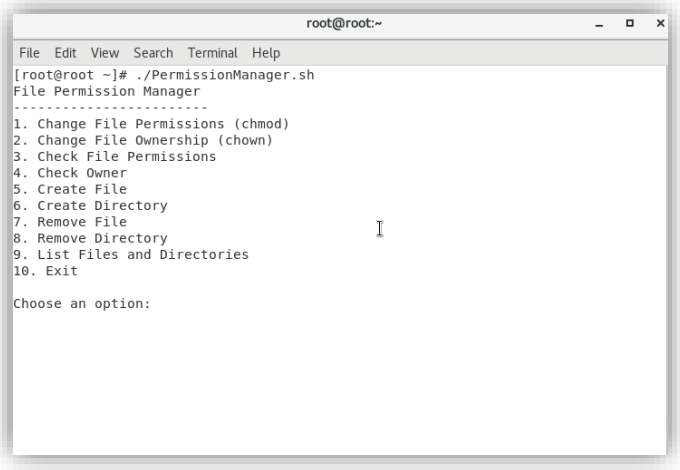
Chapter 4: Sample Reports

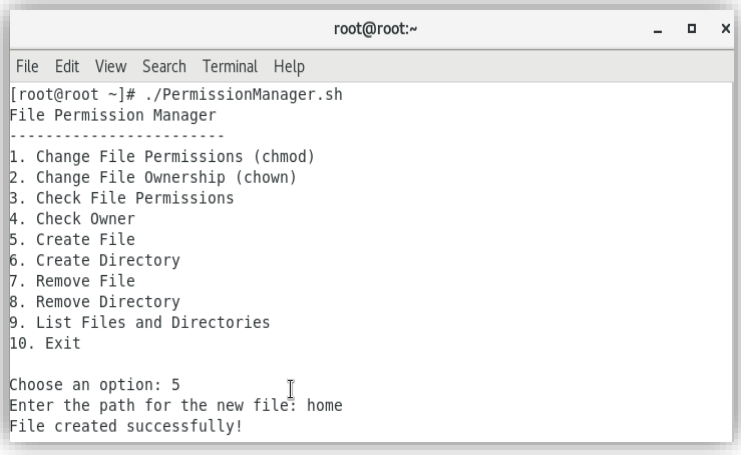
(Sample CODE)

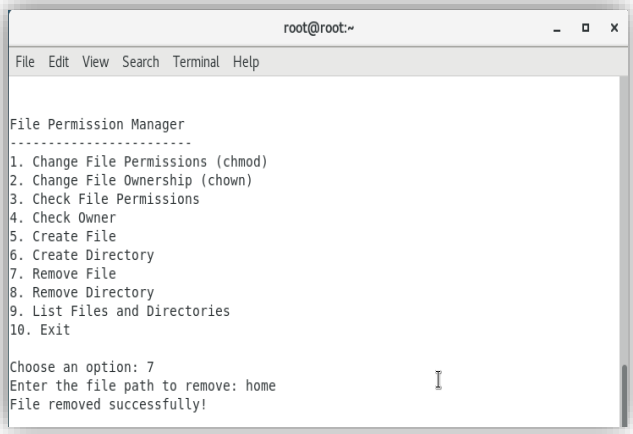
```
#!/bin/bash

# File Permission Manager Script

# Function to display menu
display_menu() {
   echo "File Permission Manager"
   echo "-----------------------"
   echo "1. Change File Permissions (chmod)"
   echo "2. Change File Ownership (chown)"
   echo "3. Check File Permissions"
   echo "4. Check Owner"
   echo "5. Create File"
   echo "6. Create Directory"
   echo "7. Remove File"
   echo "8. Remove Directory"
   echo "9. List Files and Directories"
   echo "10. Exit"
   echo ""
   echo -n "Choose an option: "
}

# Function to change file permissions
change_permissions() {
   echo -n "Enter the file/directory path: "
   read file_path
   echo -n "Enter the permission (e.g., 755 or u+rwx): "
   read permission
   sudo chmod $permission $file_path
   if [ $? -eq 0 ]; then
      echo "Permissions changed successfully!"
   else
      echo "Failed to change permissions."
   fi
}

# Function to change file ownership
change_ownership() {
   echo -n "Enter the file/directory path: "
```

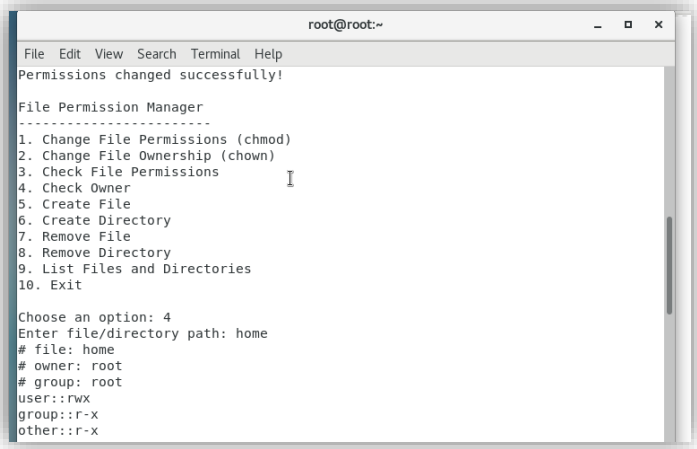
```
    read file_path
    echo -n "Enter the new owner (username): "
    read owner
    echo -n "Enter the new group (optional, leave blank for none): "
    read group

    if [ -z "$group" ]; then
    sudo chown $owner $file_path
    else
      sudo chown $owner:$group $file_path
    fi

    if [ $? -eq 0 ]; then
       echo "Ownership changed successfully!"
    else
       echo "Failed to change ownership."
    fi
}

# Function to check file permissions
check_permissions() {
    echo -n "Enter the file/directory path: "
    read file_path
    ls -l $file_path
}

check_owner(){
    echo -n "Enter file/directory path: "
    read path
    getfacl $path
}

# Function to create a file
create_file() {
    echo -n "Enter the path for the new file: "
    read file_path
    touch $file_path
    if [ $? -eq 0 ]; then
       echo "File created successfully!"
    else
       echo "Failed to create file."
```

Output:


root@root:~
File Edit View Search Terminal Help
[root@root ~]# ./PermissionManager.sh
File Permission Manager
1. Change File Permissions (chmod)
2. Change File Ownership (chown)
3. Check File Permissions
4. Check Owner
5. Create File
6. Create Directory
7. Remove File
8. Remove Directory
9. List Files and Directories
10. Exit
Choose an option:


root@root:~
File Edit View Search Terminal Help
[root@root ~]# ./PermissionManager.sh
File Permission Manager
1. Change File Permissions (chmod)
2. Change File Ownership (chown)
3. Check File Permissions
4. Check Owner
5. Create File
6. Create Directory
7. Remove File
8. Remove Directory
9. List Files and Directories
10. Exit
Choose an option: 5
Enter the path for the new file: home
File created successfully!


root@root:~
File Edit View Search Terminal Help
File Permission Manager
1. Change File Permissions (chmod)
2. Change File Ownership (chown)
3. Check File Permissions
4. Check Owner
5. Create File
6. Create Directory
7. Remove File
8. Remove Directory
9. List Files and Directories
10. Exit
Choose an option: 7
Enter the file path to remove: home
File removed successfully!


root@root:~
File Edit View Search Terminal Help
Permissions changed successfully!
File Permission Manager
1. Change File Permissions (chmod)
2. Change File Ownership (chown)
3. Check File Permissions
4. Check Owner
5. Create File
6. Create Directory
7. Remove File
8. Remove Directory
9. List Files and Directories
10. Exit
Choose an option: 4
Enter file/directory path: home
file: home
owner: root
group: root
user::rwx
group::r-x
other::r-x

5. Future Enhancement

- **Graphical User Interface (GUI)**: A GUI can be developed for users who prefer visual over command-line interfaces.
- **Logging System**: Adding a logging mechanism to track changes to file permissions and ownership, which is useful for audit and security purposes.
- **Extended Permissions and ACL Support**: Implement advanced permissions and ACL support to handle more complex file management needs.
- **Enhanced Error Handling**: Adding more robust error handling to guide users in case of incorrect input or operation failures.

6. Conclusion

The **File Management System in Linux** project effectively simplifies complex file-related tasks by providing a user-friendly, menu-driven interface through a bash script. By automating common operations like managing file permissions, ownership, and handling file creation or removal, this system saves time and reduces the likelihood of errors. The system is designed to address the needs of system administrators, developers, and general users by offering a more intuitive way to manage files and directories in a Linux environment.

This project provides several benefits, including streamlined operations for users who might be unfamiliar with the precise syntax required by various Linux file management commands. It consolidates multiple functionalities into one accessible interface, making it an essential tool for individuals working in Linux environments. Furthermore, the system's simplicity ensures that it can be easily integrated into existing workflows without the need for additional resources or significant training.

The implementation of this system demonstrates the power of bash scripting in Linux to create efficient, lightweight utilities for routine tasks. The script enables users to manage file permissions and ownership with ease, enhancing both productivity and system security. Additionally, the project offers flexibility for future expansion, such as the development of a graphical user interface (GUI) or the addition of advanced file control mechanisms like ACL (Access Control Lists).

Overall, this **File Management System** serves as a valuable tool that bridges the gap between complex Linux commands and user-friendly accessibility. By leveraging the existing capabilities of the Linux operating system, it provides an effective and reliable solution for managing files, making it a powerful utility for Linux users of all experience levels.

7. Bibliography

1. **Linux Documentation Project**, "Bash Scripting Guide."
   The Linux Documentation Project is a comprehensive resource for learning the ins and outs of Linux, including detailed guides on bash scripting and system administration. This guide has been instrumental in understanding the core principles behind bash scripting and how it can be used to automate tasks in the Linux environment.
2. **GNU Bash Manual**, "Bash Features."
   The GNU Bash manual offers a detailed overview of the Bash shell's features, including its scripting capabilities. This resource was used to implement several of the script's functions, such as conditionals, loops, and input handling, ensuring the script operates smoothly and effectively.
3. **Linux Man Pages: Commands `chmod`, `chown`, `ls`, `getfacl`, `touch`, `mkdir`, `rm`, `rmdir`.**
   The official manual pages for the Linux commands used in the system provide essential syntax and options. These references helped clarify the specific usage of commands like `chmod` for changing permissions, `chown` for changing ownership, `ls` for listing files, and others, which form the backbone of the file management functions in this project.
4. **Bash Guide for Beginners**, by Machtelt Garrels, **Bash Scripting** (2003).
   This book offers an excellent introduction to bash scripting, covering basic concepts such as variables, functions, and control structures. The project made use of these foundational concepts to build the script's functionality in a modular and reusable manner.
5. **"The Linux Command Line"**, by William E. Shotts Jr. (No Starch Press, 2012).
   This book provides a comprehensive introduction to using the Linux command line. It was particularly helpful for understanding how to handle file management operations in Linux using command-line tools, such as `chmod`, `chown`, and `ls`, which are critical for the script's operations.
6. **Linux Access Control Lists (ACLs)**, Red Hat Customer Portal, https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_system_administrators_guide/sect-selinux-access-control-lists.
   This online resource provided useful insights into advanced file permissions and ACLs, which can enhance the security and flexibility of file management. While not explicitly implemented in this version of the system, it offers ideas for potential future expansions of the project.
7. **Bash Command Line Syntax and Structure**, Linux.org, https://www.linux.org/.
   This website provides valuable resources on the command line, bash syntax, and scripting. It was referenced to ensure proper implementation of command-line syntax and structure in the bash script, ensuring compatibility and correctness in execution.