Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Fall, Year:2022), B.Sc. in CSE (Day)

File Controlling System of OS

Course title: Operating System Lab
Course Code: CSE 310 **Section:** 202 D2

Students Details

Name	ID
Md. Rajin Saleh	202002069

Submission Date : 10-01-2023
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 Introduction

File Controlling System of OS is the title of the project. A File Controlling System is a process of maintaining any kind of records in a proper manner like your document or your money records and this is the process to divide things in different stages and in writing from so that in future when needed it will be easy to get that particular record or easily said that it is an art of storing, naming, sorting and handling documents files in a systematic manner. So, in future it will be easy to retrieve data. It is one of the basic and important features of an operating system. Operating systems are used to manage files of computer systems. The project shows a "file controlling system of OS" which can perform different operations or tasks and have different functionalities. From this project, the outcomes or goals are to understand the basic concept of file controlling system of Operating system and to become familiar with a project implementation using shell scripting languages.

1.2 Motivation

The main motivation of this project is actually based on real life application which is file management problem and the operating system course and shell script is also motive me to involve in this project.

1.3 Problem Definition

1.3.1 Problem Statement

The problem statement is there are several file controlling operating which are doing in shell script. So, that here, I work with whiptail for creating menu and End of the task. The following are some of the tasks performed by file management of operating system of any computer system:

1. It helps to create new files in computer system and placing them at the specific

locations. item It helps in easily and quickly locating these files in computer system.

2. It helps to stores the files in separate folders known as directories. These directories help users to search file quickly or to manage the files according to their types or uses.
3. It helps the user to modify the data of files or to modify the name of the file in the directories etc.

File management helps users to organize their valuable documents in a systematic manner for better and efficient use of it.

1.3.2 Complex Engineering Problem

Here, I was working with a file controlling system of OS with several operations using shell scripting and bash command. Here, the challenge of work is to deal with Whiptail command in my project and many others new bash command which I wouldn't use before. So, the challenge of using whiptail command, I would be able to create a GUI look for this project.

1.4 Design Goals/Objectives

In this project, The objectives is about -

- To develop and understand basic the concept of file controlling system of OS.
- To know about how to make an operating system based project using shell script.
- To familiar with a project implementation using shell scripting languages with Whiptail.

Chapter 2

Design/Development/Implementation of the Project

2.1 Tools Technologies

Below given uses tools and technologies for the whole project -

1. Desktop / Laptop
2. Linux Operating System.
3. Shell Scripting Language.

2.2 Functionalities

The project contains some functionalities which are write in given below-

- List all Files and Directories.
- Create New Files.
- Delete Existing Files.
- Rename an Existing Files.
- Edit Files Content.
- Search for a file.
- View Content of File.
- Details of Particular File.
- Sort Files Content.
- Sort all Files in a Directories.

- Count Number of Directories.
- Count Number of Files.
- List only directories(folders).
- List files of particular extension.
- End the script

2.3 Project Paradigm

The file is actually the collection of associated information. This file-system prearranged into directory for efficient usage. Every directory has a number of files and other directories. The directory is defined as a bit which distinguish the entries that explained file and subdirectories in the recent directory. By theoretically we may change the file into a directory by changing its bit. A file system is considered as an element of an operating system that manage the storage space and operation of files on media like disks. The given below figure show a general hierarchy of the storage in an operating system-

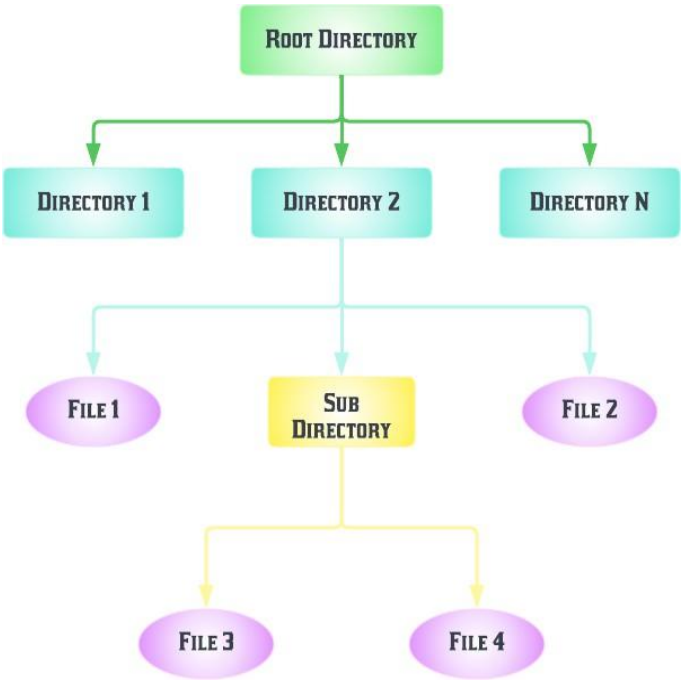


Figure 2.1: General hierarchy of the storage in an operating system.

2.4 Procedure

This script is a menu-based program that presents the user with a series of options for manipulating files and directories in a Unix-like operating system. The options are displayed using the whiptail command, which creates a visual menu for the user to select from. The user's selection is stored in the opt1 variable. The script then uses an if statement to check the value of opt1 and execute the corresponding block of code for the selected option. The options include:

1. List all Files and Directories.
2. Create New Files.
3. Delete Existing Files.
4. Rename an Existing Files.
5. Edit Files Content.
6. Search for a file.
7. View Content of File.
8. Details of Particular File.
9. Sort Files Content.
10. Sort all Files in a Directories.
11. Count Number of Directories.
12. Count Number of Files.
13. List only directories(folders).
14. List files of particular extension.
15. End the script

For each option, the script performs the requested action and provides feedback to the user through the command line or through additional whiptail dialogs.

2.4.1 Algorithm

The algorithm for this project is given below -

1. Display a welcome message to the user using the whiptail command.
2. Display a menu of options to the user using the whiptail command.
3. Store the user's selection in a variable.
4. Use an if statement to check the value of the user's selection.

5. For each option:
   (a) Perform the requested action.
   (b) Provide feedback to the user through the command line or additional whiptail dialogs.
6. Repeat from step 4 until the user selects the "End script" option.

2.5 Implementation

Code Snapshot

```
whiptail --title "Home Page" --msgbox "Welcome To File Controlling System of Operating System... " 10 65

opt1=$(
whiptail --title "Operating Systems" --menu "Make your choice" 16 60 9 \
  1 "List all Files and Directories."   \
  2 "Create New Files."  \
  3 "Delete Existing Files." \
  4 "Rename Files." \
  5 "Edit File Content." \
  6 "Search Files." \
  7 "View Content of File." \
  8 "Details of Particular File." \
  9 "Sort File Content." \
  10 "Sort Files in a Directory." \
  11 "List only Directories(Folders)." \
  12 "List Files of Particular Extension." \
  13 "Count Number of Directories." \
  14 "Count Number of Files." \
  0 "End script"  3>&2 2>&1 1>&3
)

if [ $opt1 -eq 1 ]
then
echo "Showing all files and directories...."

```

```
--
26  {
27  for ((i = 0 ; i <= 100 ; i+=5)); do
28          sleep 0.1
29          echo $i
30      done
31  } | whiptail --gauge "Loading..." 6 50 0
32   ls
33   echo " "
34
35  elif [ $opt1 -eq 2 ]
36  then
37   echo "Enter File Name: "
38
39   read filename
40
41   touch filename
42   {
43  for ((i = 0 ; i <= 100 ; i+=5)); do
44          sleep 0.1
45          echo $i
46      done
47  } | whiptail --gauge "Creating..." 6 50 0
48  sleep 3
49   echo "File Created Successfully"
50   echo " "
51
52  elif [ $opt1 -eq 3 ]
53  then
```

```
54
55   echo "Enter name of File you want to Delete!"
56   read delfile
57
58   if [ -f "$delfile" ];
59   then
60     rm $delfile
61     {
62  for ((i = 0 ; i <= 100 ; i+=5)); do
63           sleep 0.1
64           echo $i
65       done
66  } | whiptail --gauge "Deleting..." 6 50 0
67  sleep 3
68     echo "Successfully Deleted."
69     echo " "
70
71   else
72     echo "File Does not Exist..Try again"
73     echo " "
74   fi
75
76
77  elif [ $opt1 -eq 4 ]
78  then
79
80   echo "Enter Old Name of File with Extension.."
81   read old
```

```
82
83  {
84  for ((i = 0 ; i <= 100 ; i+=5)); do
85          sleep 0.1
86          echo $i
87      done
88  } | whiptail --gauge "Checking File..." 6 50 0
89  sleep 3
90   if [ -f "$old" ];
91   then
92      echo "Now Enter New Name for file with Extension"
93      read new
94      mv $old $new
95      {
96  for ((i = 0 ; i <= 100 ; i+=5)); do
97          sleep 0.1
98          echo $i
99      done
100 } | whiptail --gauge "Renaming..." 6 50 0
101 sleep 3
102     echo "Successfully Rename."
103     echo "Now Your File Exist with $new Name"
104  else
105     echo "$old does not exist..Try again with correct
    filename."
106  fi
107 echo " "
108
```

```
109  elif [ $opt1 -eq 5 ]
110  then
111   echo "Enter File Name with Extension : "
112   read edit
113
114   {
115  for ((i = 0 ; i <= 100 ; i+=5)); do
116          sleep 0.1
117          echo $i
118      done
119  } | whiptail --gauge "Checking File..." 6 50 0
120  sleep 3
121
122
123   if [ -f "$edit" ];
124   then
125  {
126  for ((i = 0 ; i <= 100 ; i+=5)); do
127          sleep 0.1
128          echo $i
129      done
130  } | whiptail --gauge "Opening..." 6 50 0
131  sleep 3
132      nano $edit
133   echo " "
134
135   else
136      echo "$edit File does not exist..Try again."
```

```
137   fi
138
139  elif [ $opt1 -eq 6 ]
140  then
141    echo "Search files here.."
142    echo "Enter File Name with Extension to search"
143    read f
144
145   if [ -f "$f" ];
146   then
147  {
148  for ((i = 0 ; i <= 100 ; i+=5)); do
149          sleep 0.1
150          echo $i
151      done
152  } | whiptail --gauge "Searching for $f file..." 6 50 0
153  sleep 5
154      echo "File Found."
155      find /home -name $f
156      echo " "
157
158   else
159      echo "File Does not Exist..Try again."
160   echo " "
161   fi
162
163  elif [ $opt1 -eq 7 ]
164  then
```

```
 echo "Enter File Name with Extension to see Detail : "
 read detail

{
for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Checking..." 6 50 0
sleep 3
 if [ -f "$detail" ];
 then
    {
for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Loading Properties..." 6 50 0
sleep 3
 stat $detail

 else
    echo "$detail File does not exist..Try again"
 fi
echo " "

elif [ $opt1 -eq 8 ]
then
```

```
193   echo "Enter File Name : "
194   read readfile
195
196   if [ -f "$readfile" ];
197   then
198   {
199  for ((i = 0 ; i <= 100 ; i+=5)); do
200          sleep 0.1
201          echo $i
202      done
203  } | whiptail --gauge "Showing..." 6 50 0
204  sleep 3
205     cat $readfile
206   else
207     echo "$readfile does not exist"
208     fi
209
210  echo " "
211
212  elif [ $opt1 -eq 9 ]
213  then
214   echo "Sort files content here.."
215   echo "Enter File Name with Extension to sort :"
216   read sortfile
217
218   if [ -f "$sortfile" ];
219   then
220   {
```

```
220   {
221  for ((i = 0 ; i <= 100 ; i+=5)); do
222          sleep 0.1
223          echo $i
224      done
225  } | whiptail --gauge "Sorting..." 6 50 0
226  sleep 3
227      sort $sortfile
228   else
229      echo "$sortfile File does not exist..Try again."
230   fi
231
232  echo " "
233
234  elif [ $opt1 -eq 10 ]
235  then
236   echo "showing all Directories..."
237  {
238  for ((i = 0 ; i <= 100 ; i+=5)); do
239          sleep 0.1
240          echo $i
241      done
242  } | whiptail --gauge "Loading..." 6 50 0
243  sleep 3
244
245   ls -d */
246
247   echo " "
```

```
248  elif [ $opt1 -eq 11 ]
249  then
250   echo "Enter List type extenstion: "
251   read ext
252  {
253  for ((i = 0 ; i <= 100 ; i+=5)); do
254          sleep 0.1
255          echo $i
256      done
257  } | whiptail --gauge "Showing..." 6 50 0
258  sleep 3
259   ls *ext
260
261  echo " "
262
263  elif [ $opt1 -eq 12 ]
264  then
265
266  {
267  for ((i = 0 ; i <= 100 ; i+=5)); do
268          sleep 0.1
269          echo $i
270      done
271  } | whiptail --gauge "Loading..." 6 50 0
272  sleep 3
273
274  {
275  for ((i = 0 ; i <= 100 ; i+=5)); do
```

```
for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Counting..." 6 50 0
sleep 3

 echo "Number of Directories are : "
 echo */ | wc -w

echo " "


elif [ $opt1 -eq 13 ]
then
{
for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Counting..." 6 50 0
sleep 3
 echo "Number of Files are : "
 ls -l | grep -v 'total' | grep -v '^d' | wc -l

echo " "

elif [ $opt1 == 14 ]
```

```
then

 echo "Sort Files here.."
 echo "Your Request of Sorting file is Generated."
{
for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Shorting..." 6 50 0
sleep 3
 ls | sort

 echo " "
elif [ $opt1 == 0 ]
then
 {
for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Closing..." 6 50 0
echo "Successfully Exit..."
 exit 0

else
  echo "Invalid Input!.Try again...."
fi
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

As far as the simulation setup or simulation environment is concerned, it is important to consider the following factors:

- Hardware: The basic computer or server was the simulation be run on it was a virtual machine. The system specifications (e.g. CPU, RAM, storage)core i5 6th gen, Ram 4 GB and Storage 512 GB.
- Operating system: It was a Unix-like system (e.g. Linux, MacOS). Ubuntu linux 22.04.01 version was the platform and the same OS be used for all machines involved in the simulation
- Dependencies: The simulation doesn't needs to used any third party software.
- Networking: The simulation doesn't involve in the networking.
- Data: The data will be stored in the memory and it was used for only this project.

  By considering these and other factors, I can establish a simulation setup and environment that is appropriate for project.

3.2 Results Analysis/Testing

The Result's Figure is given in below -

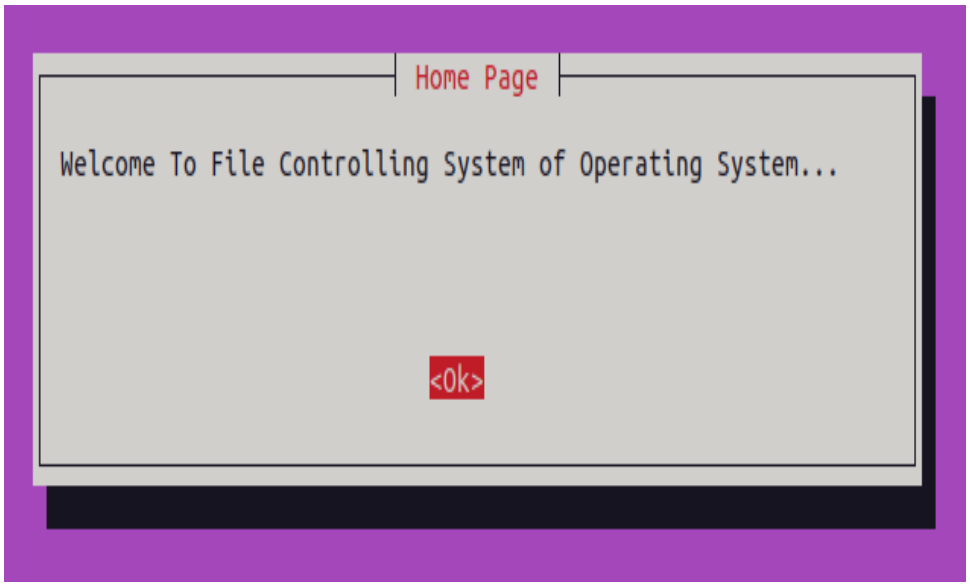
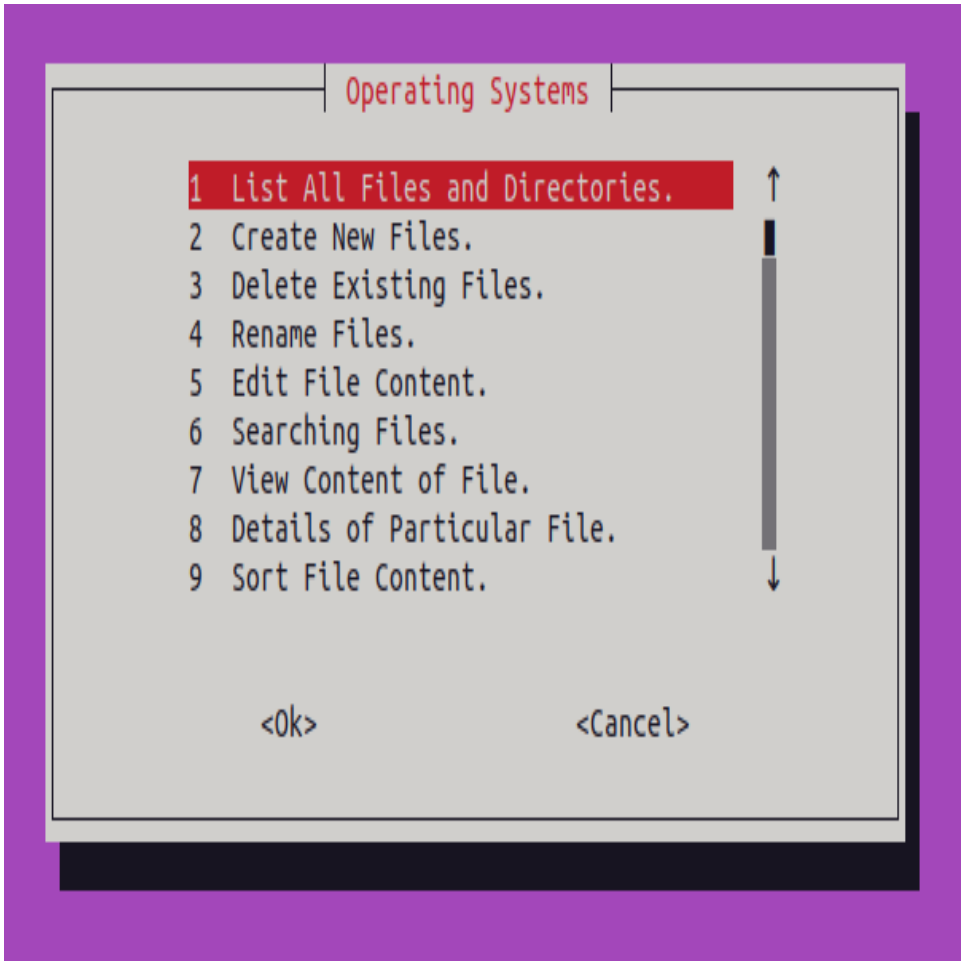


Figure 3.1: After run the program, the home page popup box open



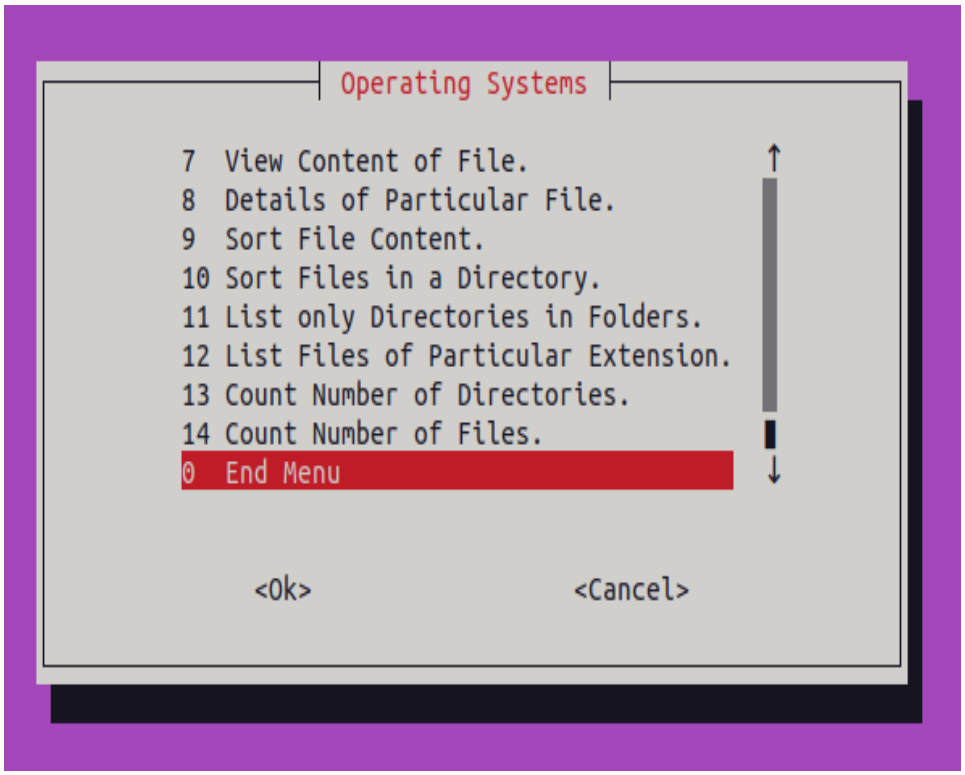


Figure 3.2: After Ok click, The menu popup box open.

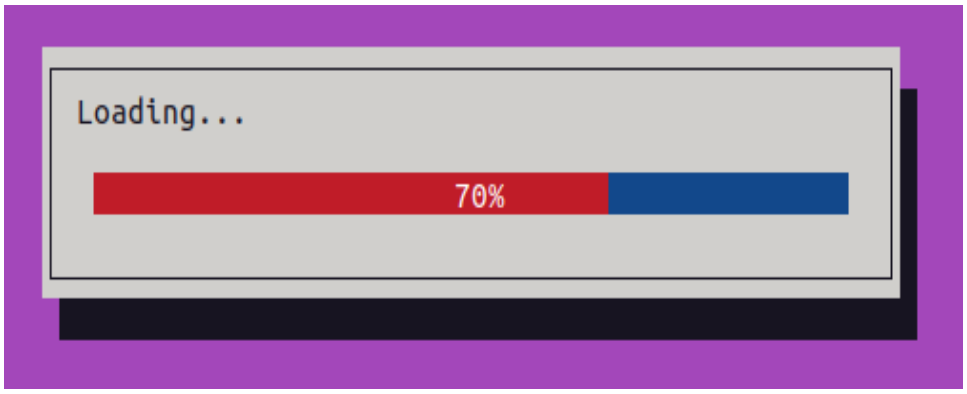


Figure 3.3: After click on option 1, loading for list showing.

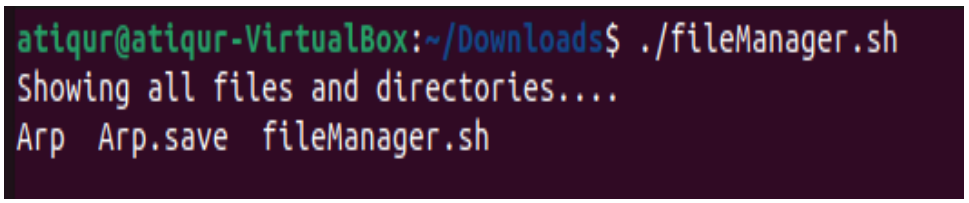


Figure 3.4: Showing all the list.

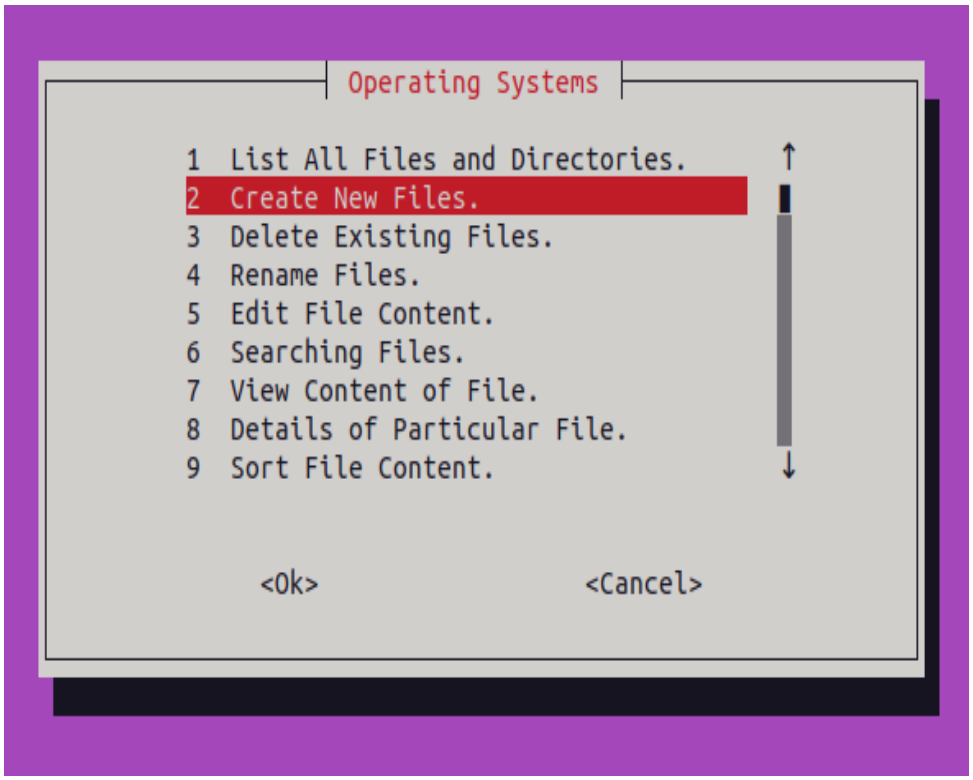


Figure 3.5: Creating option.

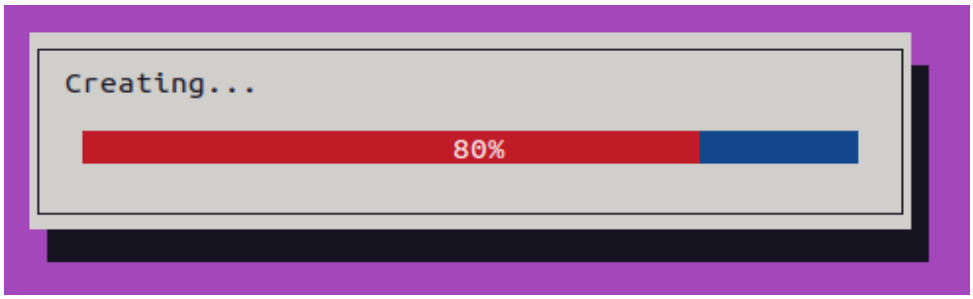


Figure 3.6: Creating load..

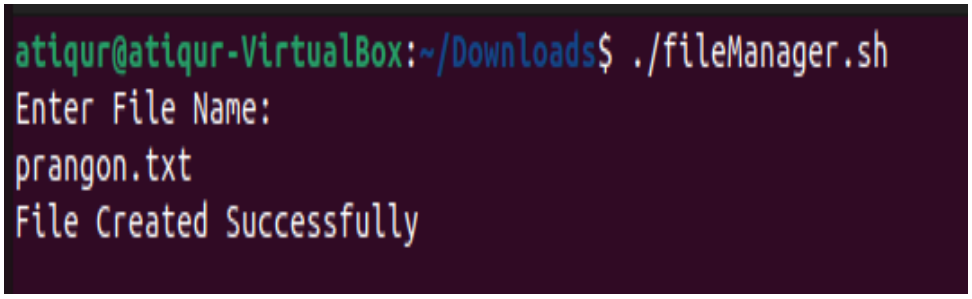


Figure 3.7: Created a new file.

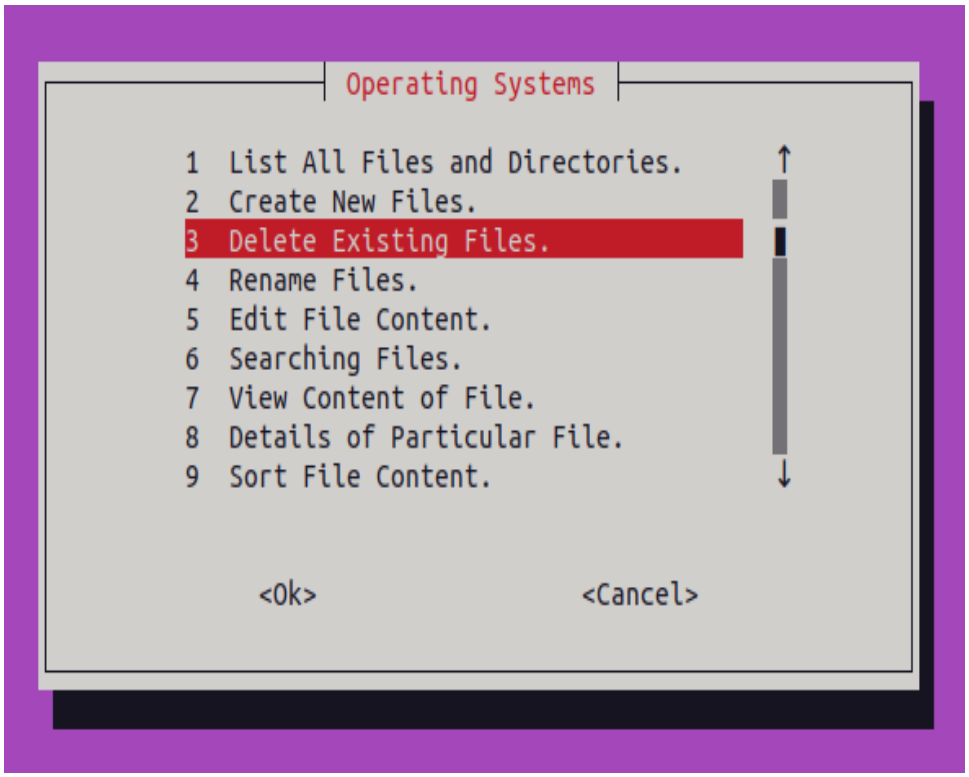


Figure 3.8: Deleting Option.

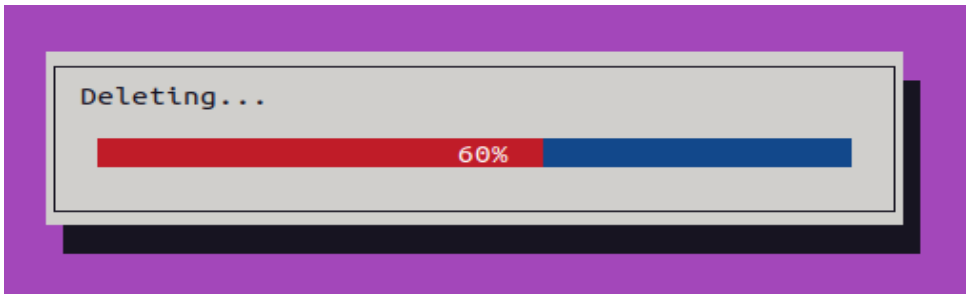


Figure 3.9: Deleting load.

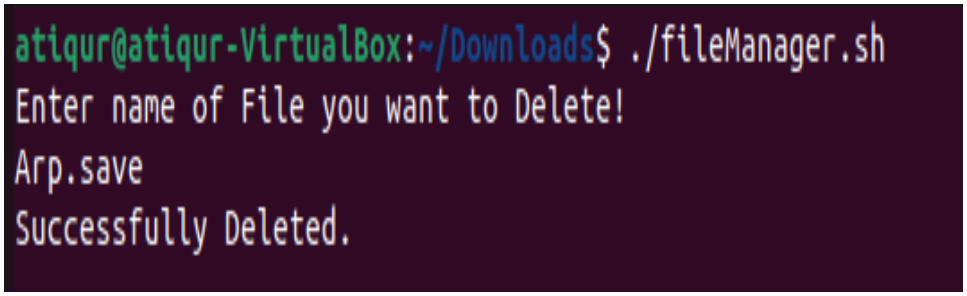


Figure 3.10: Deleting File.

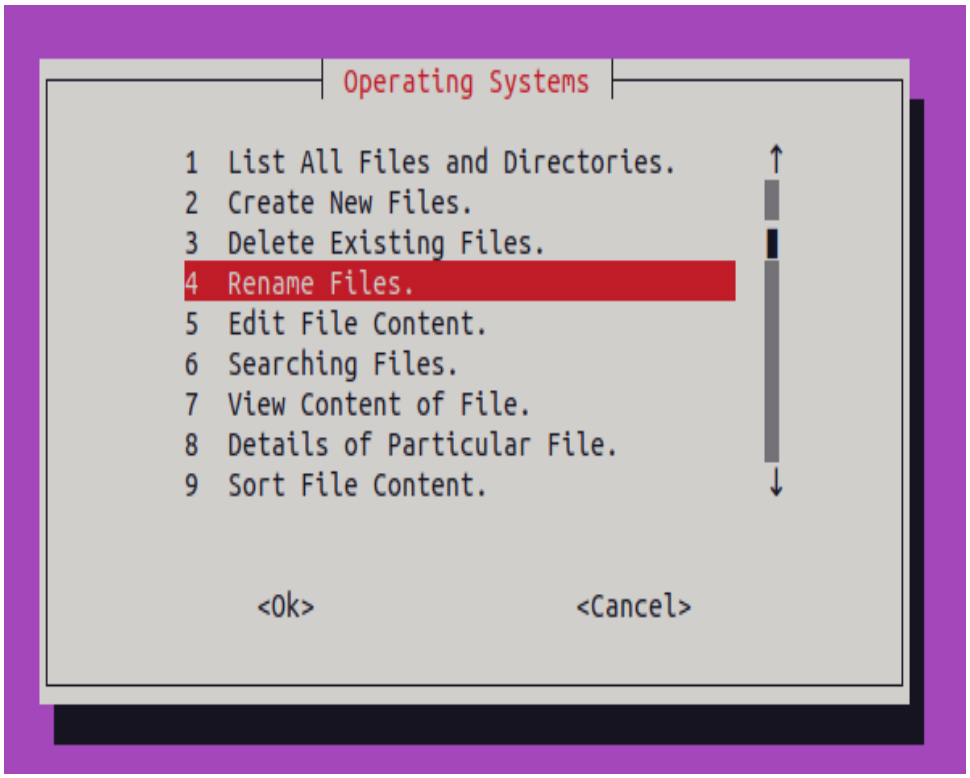


Figure 3.11: Rename option.

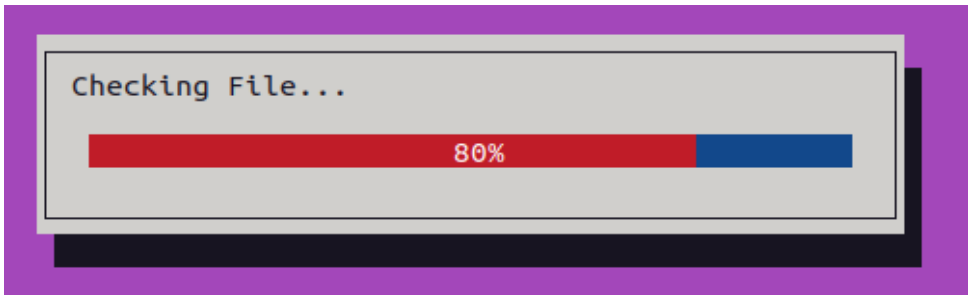


Figure 3.12: Checking existing file.

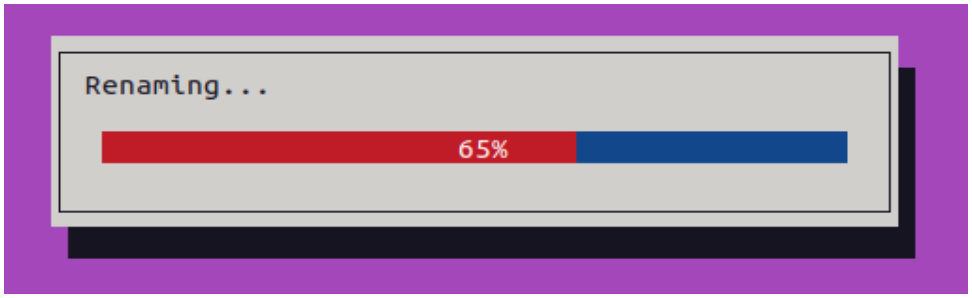


Figure 3.13: Rename loading.

```
atiqur@atiqur-VirtualBox:~/Downloads$ ./fileManager.sh
Enter Old Name of File with Extension..
Arp
Now Enter New Name for file with Extension
Atiqur.txt
Successfully Rename.
Now Your File Exist with Atiqur.txt Name
```

Figure 3.14: Rename successfully.

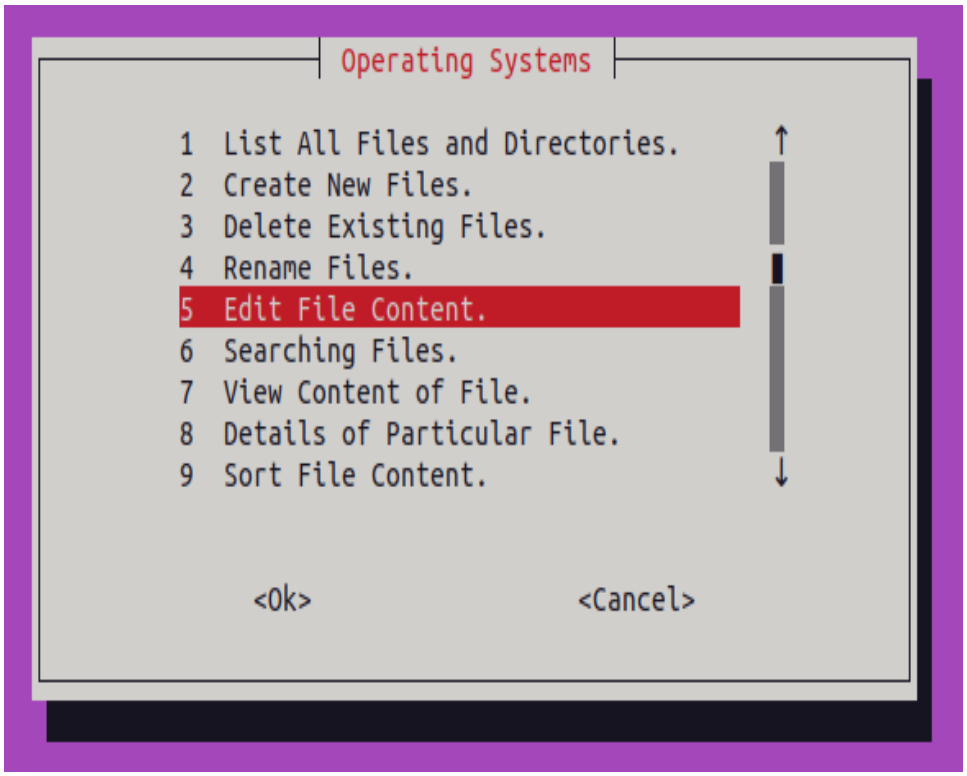


Figure 3.15: Edit File option.

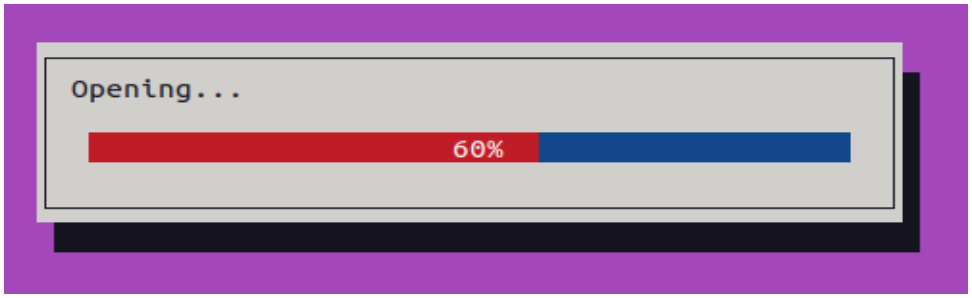


Figure 3.16: Opening file for Edit.

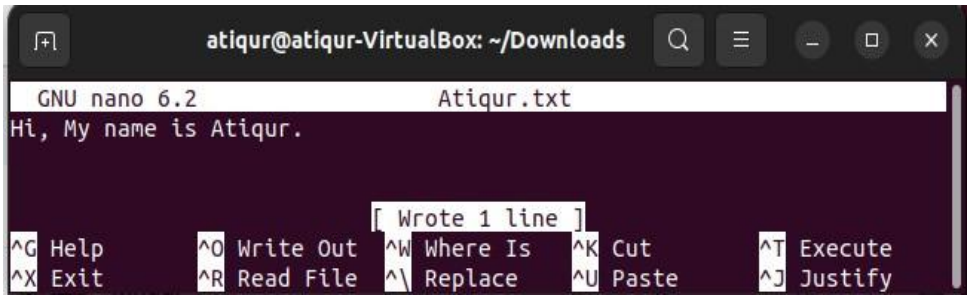


Figure 3.17: Editing the file in nano editor.

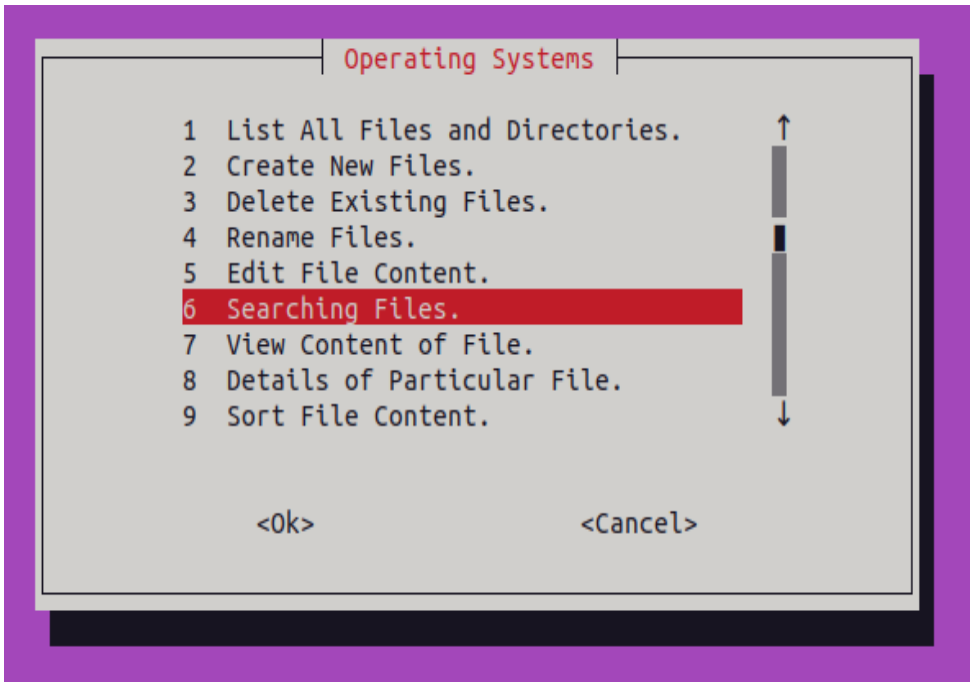


Figure 3.18: Searching option.

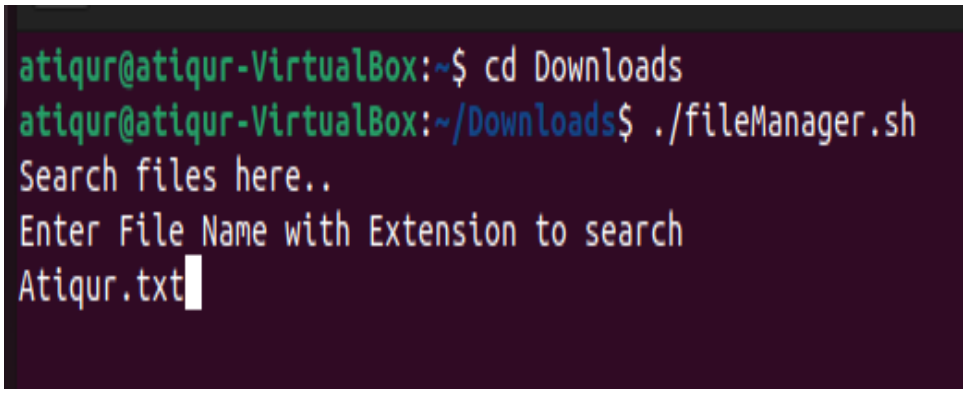


Figure 3.19: File name for search.

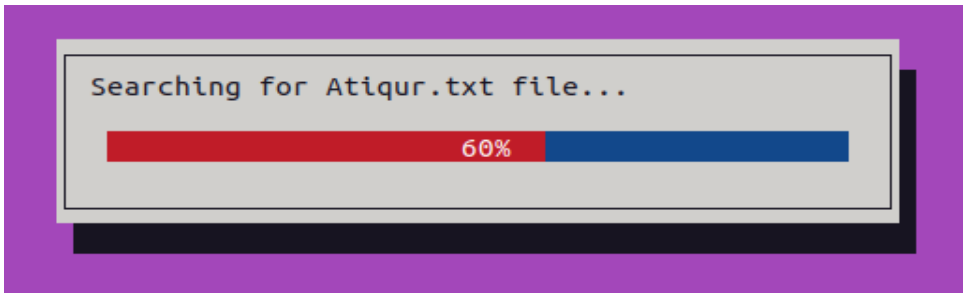


Figure 3.20: Searching for the given file name.

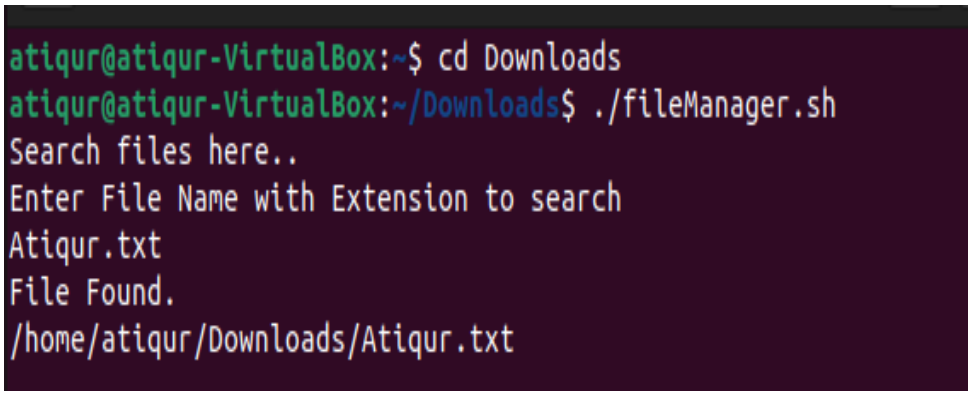


Figure 3.21: File Found.

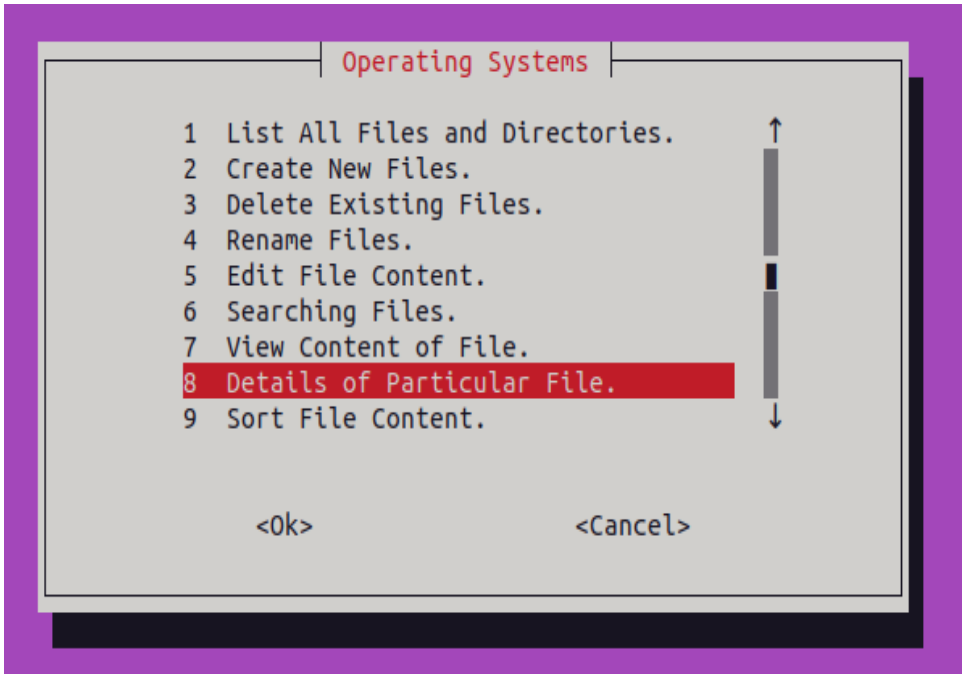


Figure 3.22: Details of Particular File Option.

```
atiqur@atiqur-VirtualBox:~/Downloads$ ./fileManager.sh
Enter File Name with Extension to see Detail :
Atqiur.txt
```

Figure 3.23: Enter the file name for details.

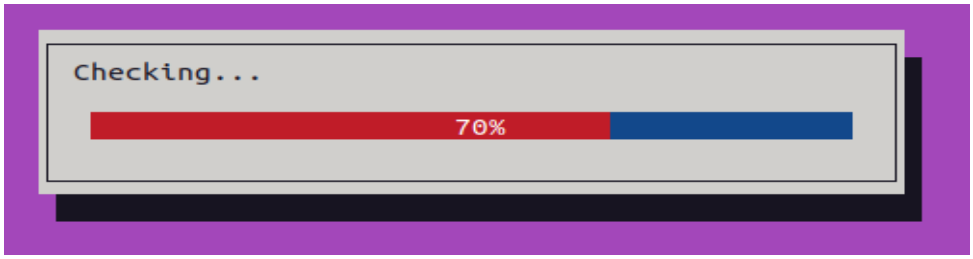


Figure 3.24: Checking existing file for details.

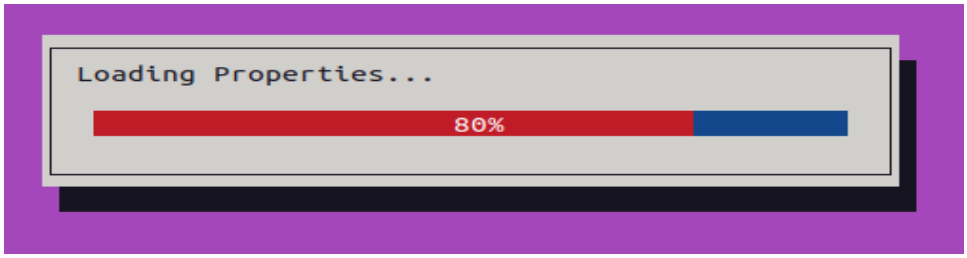


Figure 3.25: Loading file properties.

```
atiqur@atiqur-VirtualBox:~/Downloads$ ./fileManager.sh
Enter File Name with Extension to see Detail :
Atiqur.txt
  File: Atiqur.txt
  Size: 23              Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d      Inode: 655705      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  atiqur)   Gid: ( 1000/  atiqur)
Access: 2023-01-10 01:45:55.462351700 +0600
Modify: 2023-01-10 01:35:18.161524559 +0600
Change: 2023-01-10 01:35:18.161524559 +0600
 Birth: 2023-01-09 22:00:24.017198307 +0600
```

Figure 3.26: File Properties Displayed.

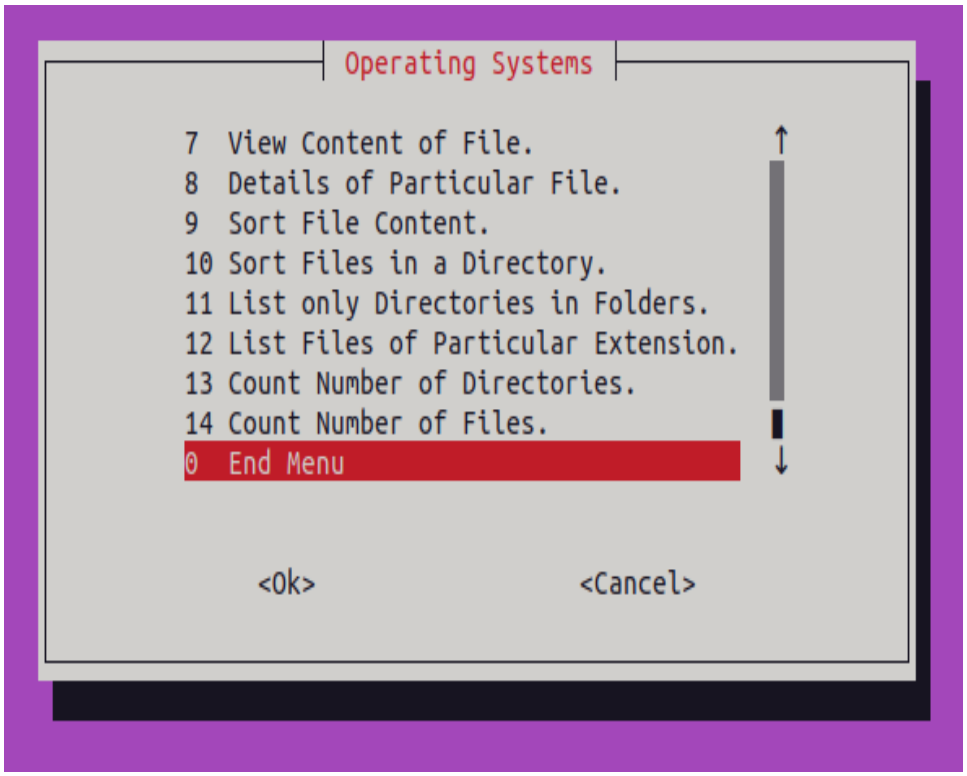


Figure 3.27: End Option.

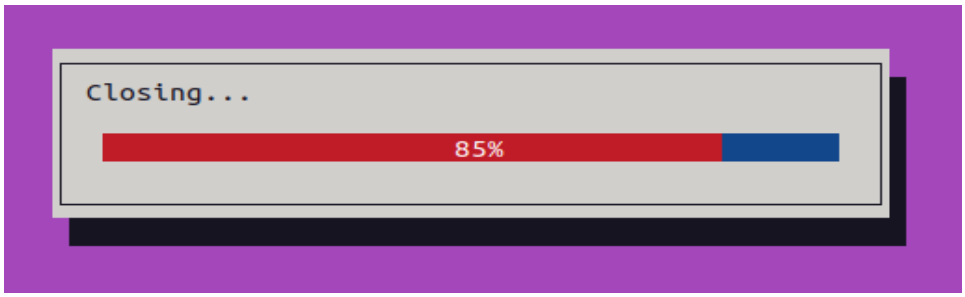


Figure 3.28: Closing.

Figure 3.29: Close successfully.

3.3 Results Overall Discussion

Here In the output section, there are some functions output snapshots are given. From the output section see that, the result is accurately found with expected output. The project's program is executed successfully without any bugs or errors. So, I agree that this project program can be used to for a file controlling system in operating system.

3.4 Achievement

By completing this project, I achieved all the objectives with the main concept of this project which is developing a file controlling system and also get the desired outputs.

3.5 Challenge face

in this project, I faced some challenges when doing the tasks which are given below -

- When working with Whiptail command for welcome message and creating menu found some error which is recovered after trying many times with research about whiptail.
- When working with different operations of file system then facing maintaining problems which is little bit challenge for me.

Chapter 4

Conclusion

4.1 Discussion

The project is about "A File Controlling System" which is a menu-based program for manipulating files and directories in a Unix-like operating system. The project provides a variety of options for listing, creating, deleting, renaming, and editing files, as well as searching, sorting, and viewing the contents of files. The project uses the whiptail command to create a user interface in the terminal, and relies on a variety of Unix commands and Bash scripting techniques to perform the various file operations. here are different basic functions that users can perform on files. All these functionalities are discussed above in the form of code as well as in simple natural language. So everyone having the basic knowledge of computer can use this file management system to perform different functions on files. It can motivate us to work in the future with daily life problems and motivates us to work with Shell scipt languages and operating system. It does not solve any new problems as file controlling based softwares already exists. What this project will do is that, it will reveal the underlying working principle and replicate all the existing file control based softwares. This project is enough to be an exhibition. So, that we can said that our project is completed.

4.2 Limitations

Here are some potential limitations of this project:

- Compatibility: The script relies on the whiptail command, which may not be available or may behave differently on all Unix-like systems. This could limit the compatibility of the script with certain systems or configurations.
- Security: The script allows the user to delete, rename, and edit files, which could potentially pose a security risk if the user has access to sensitive files or if the script is not run with proper safeguards. It is important to consider the security implications of the script and take appropriate measures to protect against unauthorized access or tampering.
- Customization: The script provides a fixed set of options for manipulating files

and directories, which may not be suitable for all use cases. It may be difficult or impossible to add or modify the options provided by the script without modifying the code itself.

- Error handling: The script does not include robust error handling or input validation, which could lead to unpredictable behavior or errors if the user provides invalid input or if the script encounters unexpected conditions.
- Performance: The script may not be efficient or perform well for large amounts of data or for operations that require significant processing power. This could limit the scalability of the script and its ability to handle large or complex tasks.

These are just a few examples of potential limitations of the project.

4.3 Scope of Future Work

Here are some potential areas for future work on this project:

- Compatibility: In future work Could be consider testing the script on a wider range of systems and configurations to ensure compatibility and identify any issues that need to be addressed. Could also consider providing alternatives or workarounds for systems that do not support the whiptail command or that have different behavior for this command.
- Security: In future could be implementing additional security measures to protect against unauthorized access or tampering with the script. This could include measures such as user authentication, file permissions, and input validation.
- Customization: In future could be consider adding options or functionality to the script to make it more flexible and adaptable to different use cases. This could include options for specifying different parameters or input files, or for interacting with external programs or services.
- Error handling: In future could be adding more robust error handling and input validation to the script to improve its reliability and prevent errors or unintended behavior. This could include checks for invalid input, missing files, or other common error conditions.
- Performance: Work could be on optimizing the performance of the script, particularly for large or complex tasks. This could involve optimizing the code, using faster algorithms, or using multiple machines or processors to parallelize the work.

These are just a few examples of potential areas for future work on the project. There may be other areas for improvement depending on your specific goals and requirements.

4.4 References

1. Amit Shukla. (2017). File Management System. Results Retrieved from https://www.includehelp.com/operating-systems/file-management-in-operatingsystem.aspx.(2022, December).

2. Linuxhint.(2019). Use of stat command. https://linuxhint.com/linu_stat_command (2022, December)

3. Ostechnix. How To Create GUI Dialog Boxes In Bash Scripts With Whiptai. Results Retrieved from https://ostechnix.com/create-gui-dialog-boxes-in-bash-scripts-with-whiptail/(2023,January).

4. Logicweb. (2004). Linux Bash Commands: A-Z (Beginner's Cheat Sheet). Results Retrieved from https://www.logicweb.com/knowledgebase/linux/linux-bash-commands-a-z/(2023, January)