جامعة الجلالة
GALALA UNIVERSITY

Efficient File Management System

For Enhanced Data Organization

CSE233 || Operating Systems

Under supervision of: Prof. Safa A. El-Askary

Project Team members

Student ID	Student Name
222102487	Aser Mohamed Ali
222100822	Sherif Farag
222101717	Merna Muhammed
222100306	Amina Elshabandy
222101590	Mostafa ibrahim
222100363	Basel Hammed

Fall 2024

➢ *Abstract:*

Efficient file management *is critical in modern computing for enhanced data organization and system optimization. This project aims to develop a user-friendly and efficient file management system using Ubuntu Server. The system implements operations such as listing files, creating, deleting, renaming, editing, searching, and sorting files, along with directory-specific functionalities. By leveraging shell scripting and C programming, the system ensures robust and scalable file management.*

➢ Introduction and Description:

File management systems play a pivotal role in operating systems, providing an interface between the user and storage devices. This project integrates Ubuntu Server, a widely used Linux-based platform, to demonstrate efficient file management techniques. Using a combination of Bash scripts and C programming, the project highlights how to automate and simplify file management tasks.

File management systems are essential for organizing and managing data. They provide a way to store, retrieve, and share files in a systematic way.

File management systems can be used to organize data on a personal computer, a network server, or in the cloud.

There are many diverse types of file management systems available, and the best system for a particular user or organization will depend on their specific Needs.

This project aims to develop an efficient file management system that improves data organization and access within an operating system.

The project focuses on optimizing performance, ensuring reliability, and enhancing overall productivity by providing users with a seamless file management experience.

➢ *Aim:*

To design and implement an ***efficient file management system*** *that enhances data organization, minimizes manual intervention, and supports comprehensive file and directory operations on Ubuntu Server.*

➢ *Related Work:*

Traditional File Managers*: Tools like Nautilus and Dolphin offer GUI-based file management but are resource-intensive.*

Command-Line Utilities*: Utilities such as ls, rm, and mv are powerful but lack an integrated user-friendly interface.*

Custom File Management Scripts: *Previous research has explored integrating basic commands into scripts to automate specific tasks. This project builds on these ideas to offer a comprehensive solution.*

➢ *The Problem Analysis (Motivation):*

Problem Statement:

Manual file management is error-prone and time-consuming, especially on servers without GUI access. There is a need for a unified tool to manage files efficiently.

Objectives:

Reduce the complexity of managing files and directories.

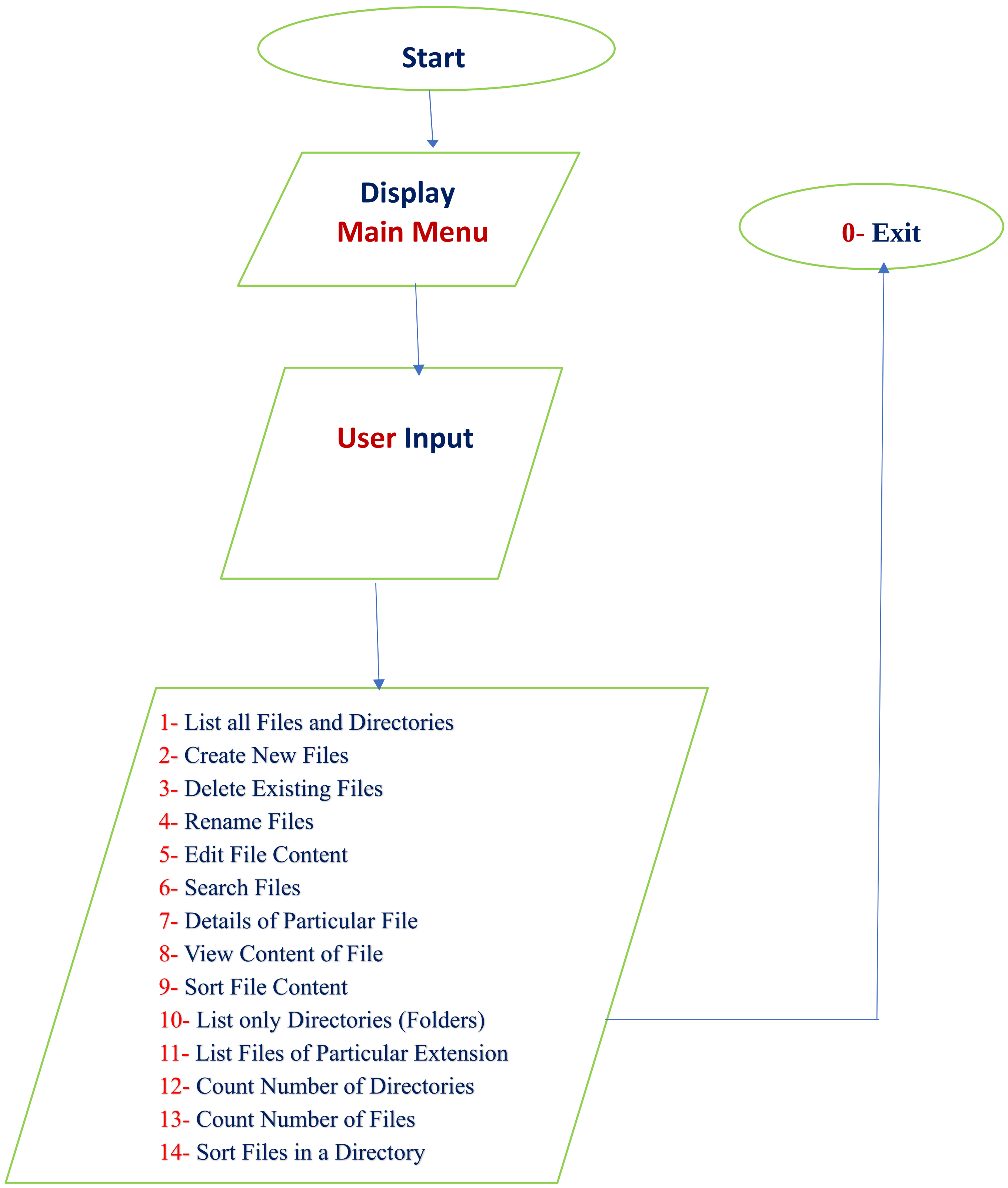
Provide quick access to file-related operations.

Automate repetitive tasks, ensuring accuracy and time savings.

Background of the Used Concepts:

- *Operating System Fundamentals*
- ***File Systems:*** *Understanding file and directory structures, permissions, and metadata.*
- ***System Calls:*** *Using calls like stat, open, and rename for file manipulation.*
- *Shell Scripting*
- *Automation of command-line tasks using Bash.*
- *Integration with system utilities for advanced operations.*
- *C Programming*
- *Efficient implementation of menu-driven interfaces.*
- *Enhanced performance for computational tasks.*

➢ *Project Flow Chart for Implementation Operations (work flow):*



Start
Display
Main Menu
0- Exit
User Input
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- Details of Particular File
8- View Content of File
9- Sort File Content
10- List only Directories (Folders)
11- List Files of Particular Extension
12- Count Number of Directories
13- Count Number of Files
14- Sort Files in a Directory

➢ *Project Code:*

Menu Code:

```
#include
int main(void) {
printf("==============================================================
 ============ \n");
                 printf("-----------------------File Management Project------------------- --------\n");
printf("==============================================================
 ============ \n");
                printf("Welcome, The Main Menu is given below:\n"); printf("1- List
all Files and Directories\n");
                printf("2- Create New Files\n");
                printf("3- Delete Existing Files\n");
                printf("4- Rename Files\n");
                printf("5- Edit File Content\n");
                printf("6- Search Files\n");
                printf("7- Details of Particular File\n");
                printf("8- View Content of File\n");
                printf("9- Sort File Content\n");
                printf("10- List only Directories(Folders)\n");
                printf("11- List Files of Particular Extension\n");
                printf("12- Count Number of Directories\n");
                printf("13- Count Number of Files\n");
                printf("14- Sort Files in a Directory\n");
                printf("0- Exit\n");
                printf("\nWhat action you want to Perform?\nEnter 1-14\n");;
return 0;
}
```

Main Code:

```
#!/bin/bash
    i="0"
    while [ $i -lt 100 ]
    do
    gcc project.c -o proj
    ./proj
    read opt1
    if [ $opt1 == 1 ]
    then
          echo "List all files and Directories here.."
          echo "Showing all files and directories...."
          sleep 3
          echo "Loading.."
sleep 3
          echo "------------------------------OutPut------------------------
    -----------"
ls
echo " "
    elif [ $opt1 == 2 ]
    then
echo "Create New Files here.."
       echo "Which type of file you want to create !"
echo "1- .c"
      echo "2- .sh"
echo "3- .txt"
echo "Enter your choice from 1-3"
read filechoice
      if [ $filechoice == 1 ]
then
      echo "Enter File Name without .c Extension"
      read filename
      touch $filename.c
      echo "------------------------------OutPut------------------------ -----------"
      echo "File Created Successfully"
      echo " "
      elif [ $filechoice == 2 ]
then
      echo "Enter File Name without .sh Extension"
      read filename2
      touch $filename2.sh
      echo "------------------------------OutPut------------------------
-----------"
      echo "File Created Successfully"
```

```
        echo " "
        elif [ $filechoice == 3 ]
        then

         echo " "
echo "Enter File Name without .txt Extension"
        read filename3
        touch $filename3.txt
        echo "-------------------------------OutPut------------------------- -----------"
        echo "File Created Successfully"
        echo " "
        else
        echo "Inavlid Input..Try Again."
        echo " "
fi
elif [ $opt1 == 3 ]
then
        echo "Delete existing files here.. "
        echo "Enter name of File you want to Delete!"
        echo "Note: Please Enter full Name with Extension."
read delfile
        echo "-------------------------------OutPut------------------------- -----------"
        if [ -f "$delfile" ];
        then
        if [ -f "$delfile" ];
        then
                rm $delfile
                echo "Successfully Deleted."

                 else
echo "File Does not Exist..Try again"
echo " "
fi
elif [ $opt1 == 4 ]
then
        echo "-------------------------------OutPut------------------------- -----------"
        echo "Rename files here.."
        echo "Enter Old Name of File with Extension.."
        read old
        echo "Checking for file..."
        sleep 3
        if [ -f "$old" ];
        then
else
fi
echo " "
echo "Ok File Exist."
echo "Now Enter New Name for file with Extension"
read new
mv $old $new
echo "Successfully Rename."
echo "Now Your File Exist with $new Name"
echo "$old does not exist..Try again with correct filename."

         elif [ $opt1 == 5 ]
then
        echo "Edit file content here.."
        echo "Enter File Name with Extension : "
read edit
        echo "-------------------------------OutPut------------------------ -----------"
        echo "Checking for file.."
        sleep 3
        if [ -f "$edit" ];
        then
else
fi
echo "Opening file.."
sleep 3
nano $edit
echo " "
echo "$edit File does not exist..Try again."
elif [ $opt1 == 6 ]
then
echo "Search files here.."
echo "Enter File Name with Extension to search"
read f
```

```
        echo "-------------------------------OutPut------------------------ -----------"
        if [ -f "$f" ];
        then

         else
fi
echo "Searching for $f File"
echo "File Found."
find /home -name $f
echo " "
echo "File Does not Exist..Try again."
echo " "
elif [ $opt1 == 7 ]
then
then
        echo "Detail of file here.."
        echo "Enter File Name with Extension to see Detail : "
read detail
        echo "-------------------------------OutPut------------------------ -----------"
        echo "Checking for file.."
        sleep 4
        if [ -f "$detail" ];
        then
else
echo "Loading Properties.."
stat $detail
echo "$detail File does not exist..Try again"

         fi echo " "
elif [ $opt1 == 8 ]
then
        echo "View content of file here.."
        echo "Enter File Name : "
        read readfile
        echo "-------------------------------OutPut------------------------ -----------"
        if [ -f "$readfile" ];
        then
echo "Showing file content.."
```

```
else
fi
echo " "
sleep 3
cat $readfile
echo "$readfile does not exist"
elif [ $opt1 == 9 ]
then
        echo "Sort files content here.."
        echo "Enter File Name with Extension to sort :"
        read sortfile
        echo "-------------------------------OutPut------------------------ -----------"
        if [ -f "$sortfile" ];
        then

         else
fi echo " "
echo "Sorting File Content.."
sleep 3
sort $sortfile
echo "$sortfile File does not exist..Try again."
elif [ $opt1 == 10 ]
then
        echo "-------------------------------OutPut------------------------ -----------"
        echo "List of all Directories here.."
        echo "showing all Directories..."
        echo "Loading.."
        sleep 3
        ls -d */
        echo " "
elif [ $opt1 == 11 ]
then
        echo "List of Files with Particular extensions here.."
        echo "Which type of file list you want to see?"
        echo "1- .c"
        echo "2- .sh"

                 echo "3- .txt"
        echo "Enter your choice from 1-3"
        read extopt
        echo "-------------------------------OutPut------------------------- -----------"
if [ $extopt == 1 ]
        then
        echo "List of .c Files shown below."
echo "Loading.."
sleep 3 ls *.c
elif [ $extopt == 2 ]
        then
        echo "List of .sh Files shown below."
        echo "Loading.."
        sleep 3
        ls *.sh
        elif [ $extopt == 3 ]
        then
        echo "List of .txt Files shown below."
        echo "Loading.."
        sleep 3
        ls *.txt
        else
        echo "Invalid Input..Try again.."
fi
echo " "

         elif [ $opt1 == 12 ]
then
        echo "-------------------------------OutPut------------------------- -----------"
        echo "Total number of Directories here.."
        echo "Loading all directories.."
        sleep 3
        echo "Counting.."
        sleep 3
        echo "Number of Directories are : "
        echo */ | wc -w
echo " "
elif [ $opt1 == 13 ]
then
```

```
          echo "-------------------------------OutPut------------------------- -----------"
          echo "Total Numbers of Files in Current Directory here.."
          echo "Loading all files.."
          sleep 3
          echo "Number of Files are : "
          ls -l | grep -v 'total' | grep -v '^d' | wc -l
echo " "
elif [ $opt1 == 14 ]
then
echo "-------------------------------OutPut--------------------------------- ---"

echo "Sort Files here.."
echo "Your Request of Sorting file is Generated."
echo "Sorting.."
sleep 3
          ls | sort
echo " "
elif [ $opt1 == 0 ]
then
          echo "Good Bye.."
          echo "Successfully Exit"
          break
else
echo "Invalid Input..Try again...."
fi
i=$[$i+1]
done
```

➢ *Running the Project:*

Prerequisites:

Ubuntu Server (or any Linux distribution).

Installed GCC compiler.

Basic knowledge of command-line operations.

Steps to Execute:

Compile the C Program:

gcc project.c -o proj

Run the Main Script:

bash main.sh

Interact with the Menu:

Follow the on-screen instructions to perform desired operations.

Troubleshooting

If the script fails to execute, ensure it has executable permissions:

chmod +x main.sh

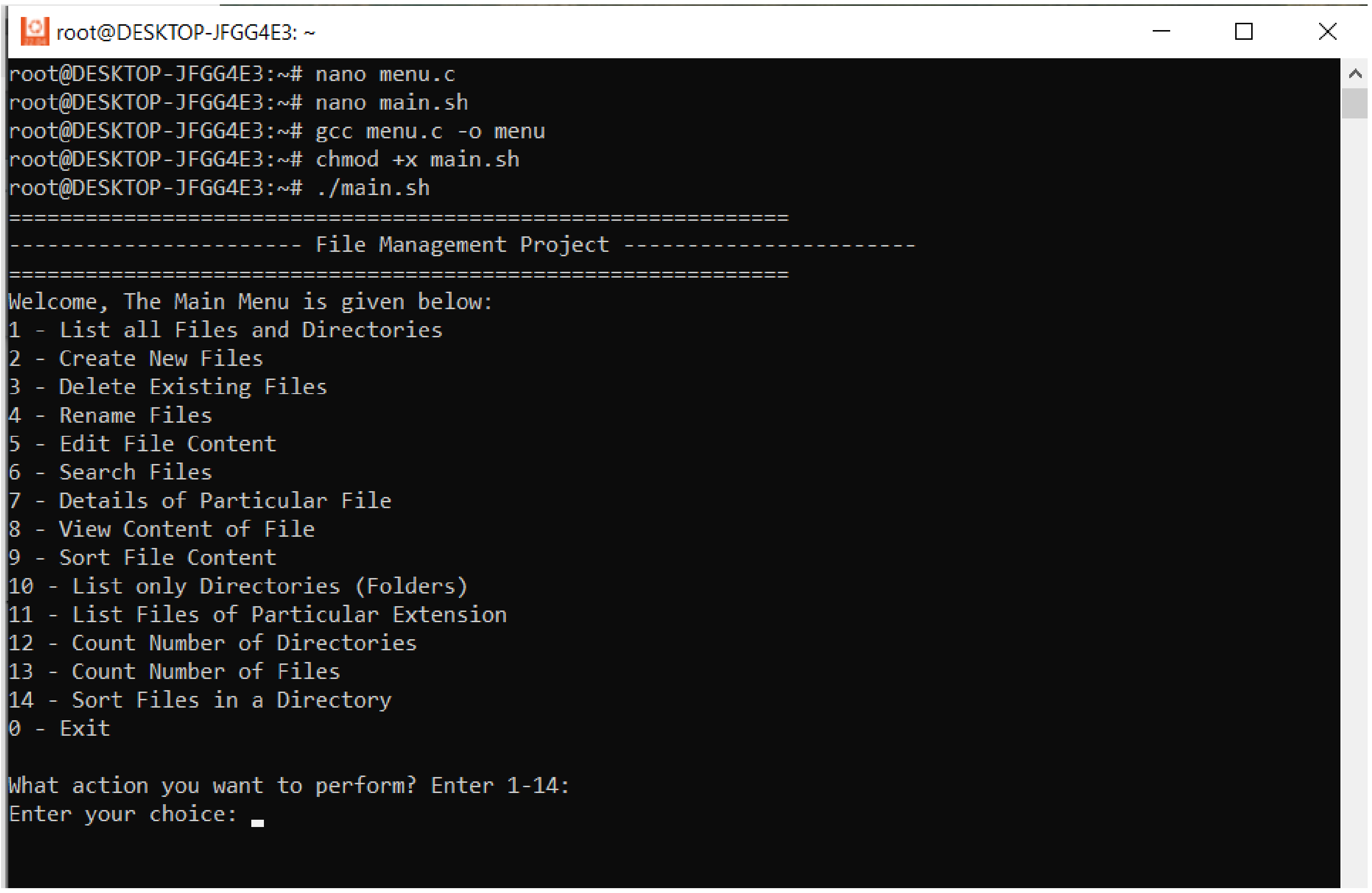
Verify that required dependencies (e.g., nano, find) are installed.

➢ *Algorithm:*

```
1. Start
2. Initialize i to 0
3. While i is less than 100:
   - Compile project.c into proj
   - Execute proj
   - Prompt user for choice (opt1)
- Decision based on opt1:
  - Option 1 (List files and directories):
    - List files and directories
  - Option 2 (Create new file):
    - Prompt for file type
    - Create file based on chosen type
  - Option 3 (Delete file):
    - Prompt for file to delete
    - Delete file if it exists
  - Option 4 (Rename file):
    - Prompt for old and new filenames
    - Rename file if it exists
  - Option 5 (Edit file):
    - Prompt for file to edit
    - Open file in nano editor if it exists
  - Option 6 (Search file):
    - Prompt for file to search
    - Search for file using "find" command
  - Option 7 (File details):
    - Prompt for file to view details
    - Display file properties using "stat" command
  - Option 8 (View file content):
    - Prompt for file to view
    - Display file content using "cat" command
  - Option 9 (Sort file content):
    - Prompt for file to sort
    - Sort file content using "sort" command
  - Option 10 (List directories):
    - List directories
  - Option 11 (List files with specific extension):
    - Prompt for file extension
    - List files with chosen extension
    - Option 12 (Count directories):
      - Count and display the number of directories
    - Option 13 (Count files):
      - Count and display the number of files
    - Option 14 (Sort files):
      - Sort files using "ls | sort"
    - Option 0 (Exit):
      - Display "Good Bye" message
      - Exit loop
  - Increment i
4. End
```

➢ Results and Discussion (Working and Outputs of code):

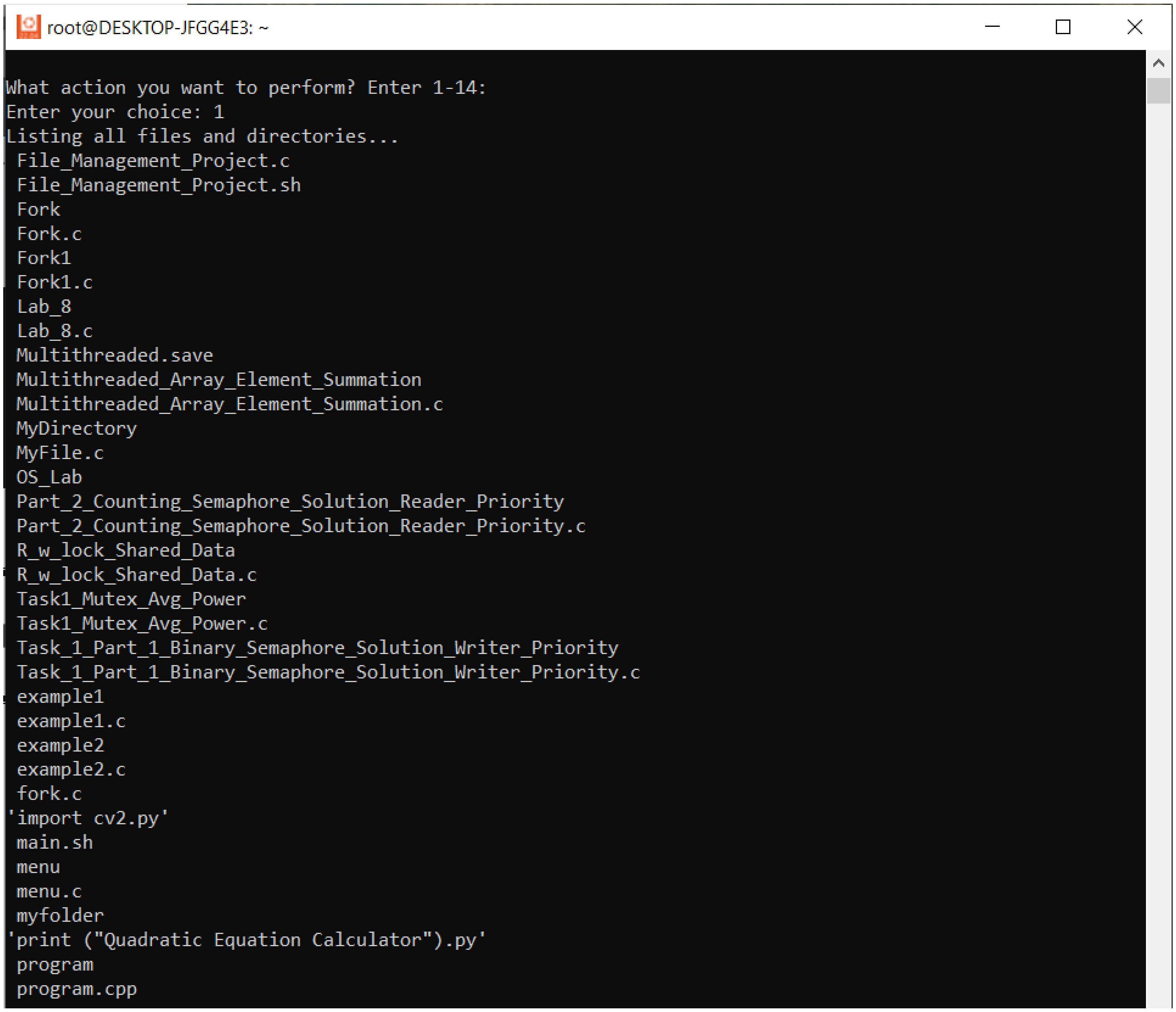
As soon as we run the code, we get the details of this project and further we can continue into working by entering any key :



root@DESKTOP-JFGG4E3: ~

```
root@DESKTOP-JFGG4E3:~# nano menu.c
root@DESKTOP-JFGG4E3:~# nano main.sh
root@DESKTOP-JFGG4E3:~# gcc menu.c -o menu
root@DESKTOP-JFGG4E3:~# chmod +x main.sh
root@DESKTOP-JFGG4E3:~# ./main.sh
=============================================================
----------------------- File Management Project -----------------------
=============================================================
Welcome, The Main Menu is given below:
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 - Details of Particular File
8 - View Content of File
9 - Sort File Content
10 - List only Directories (Folders)
11 - List Files of Particular Extension
12 - Count Number of Directories
13 - Count Number of Files
14 - Sort Files in a Directory
0 - Exit

What action you want to perform? Enter 1-14:
Enter your choice: _
```

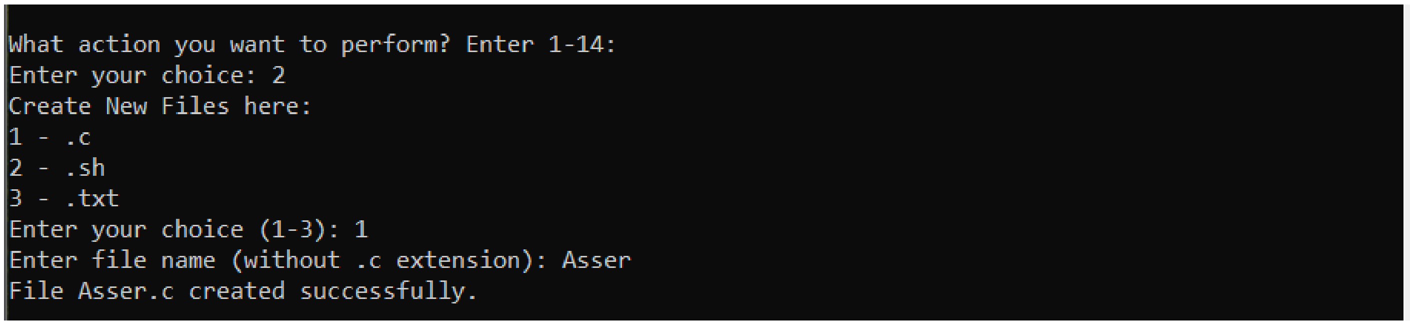
1 - List all Files and Directories:



root@DESKTOP-JFGG4E3: ~

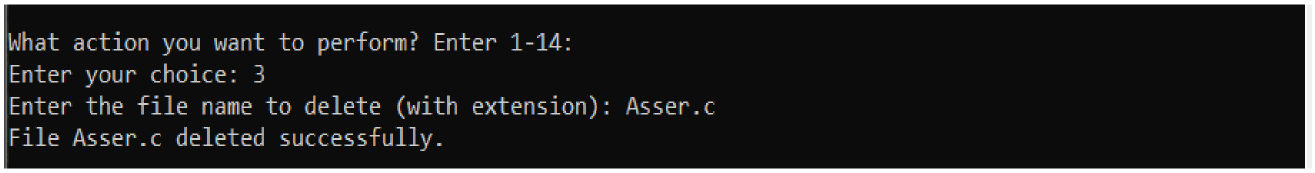
```
What action you want to perform? Enter 1-14:
Enter your choice: 1
Listing all files and directories...
 File_Management_Project.c
 File_Management_Project.sh
 Fork
 Fork.c
 Fork1
 Fork1.c
 Lab_8
 Lab_8.c
 Multithreaded.save
 Multithreaded_Array_Element_Summation
 Multithreaded_Array_Element_Summation.c
 MyDirectory
 MyFile.c
 OS_Lab
 Part_2_Counting_Semaphore_Solution_Reader_Priority
 Part_2_Counting_Semaphore_Solution_Reader_Priority.c
 R_w_lock_Shared_Data
 R_w_lock_Shared_Data.c
 Task1_Mutex_Avg_Power
 Task1_Mutex_Avg_Power.c
 Task_1_Part_1_Binary_Semaphore_Solution_Writer_Priority
 Task_1_Part_1_Binary_Semaphore_Solution_Writer_Priority.c
 example1
 example1.c
 example2
 example2.c
 fork.c
'import cv2.py'
 main.sh
 menu
 menu.c
 myfolder
'print ("Quadratic Equation Calculator").py'
 program
 program.cpp
```

2- Create New Files:



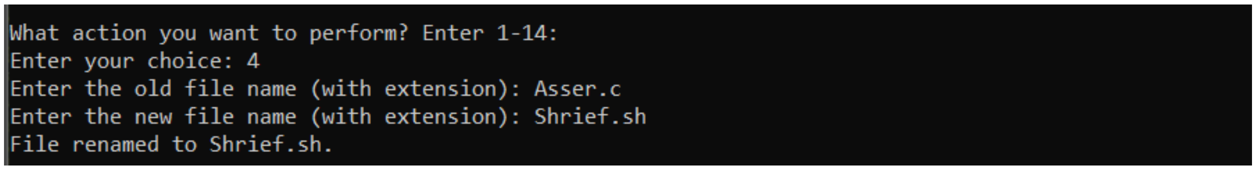
```
What action you want to perform? Enter 1-14:
Enter your choice: 2
Create New Files here:
1 - .c
2 - .sh
3 - .txt
Enter your choice (1-3): 1
Enter file name (without .c extension): Asser
File Asser.c created successfully.
```

3- Delete Existing Files:



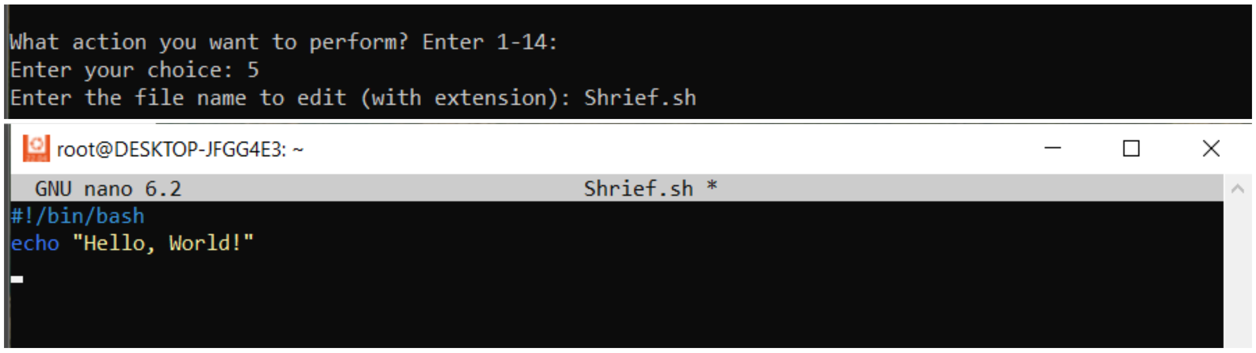
```
What action you want to perform? Enter 1-14:
Enter your choice: 3
Enter the file name to delete (with extension): Asser.c
File Asser.c deleted successfully.
```

4- Rename Files:



```
What action you want to perform? Enter 1-14:
Enter your choice: 4
Enter the old file name (with extension): Asser.c
Enter the new file name (with extension): Shrief.sh
File renamed to Shrief.sh.
```

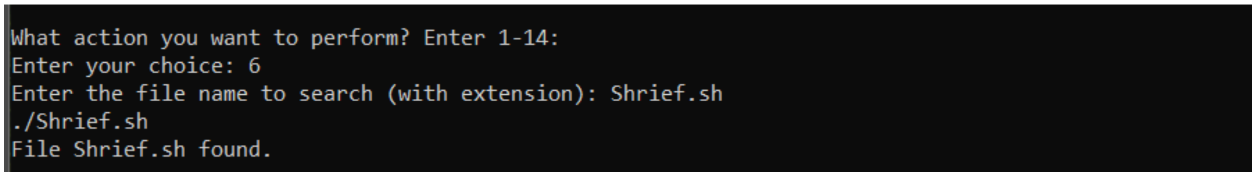
5- Edit File Content:



```
What action you want to perform? Enter 1-14:
Enter your choice: 5
Enter the file name to edit (with extension): Shrief.sh
```

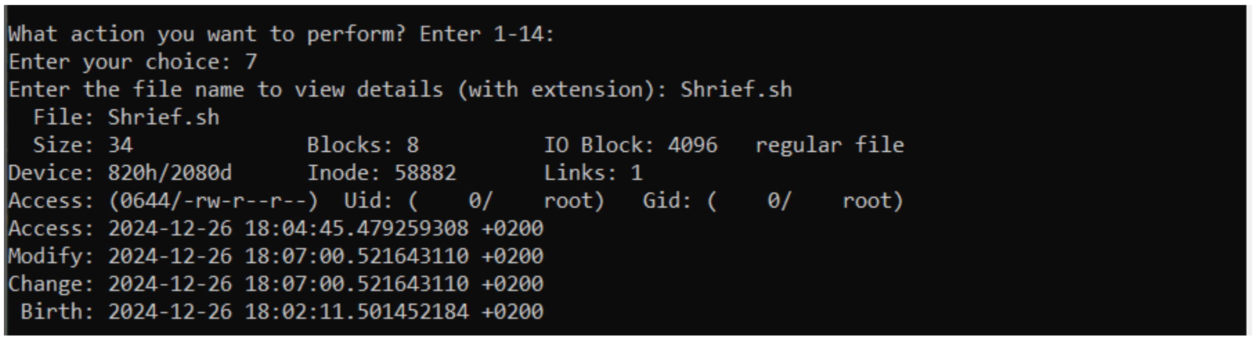
```
  GNU nano 6.2                          Shrief.sh *
#!/bin/bash
echo "Hello, World!"
```

6- Search Files:



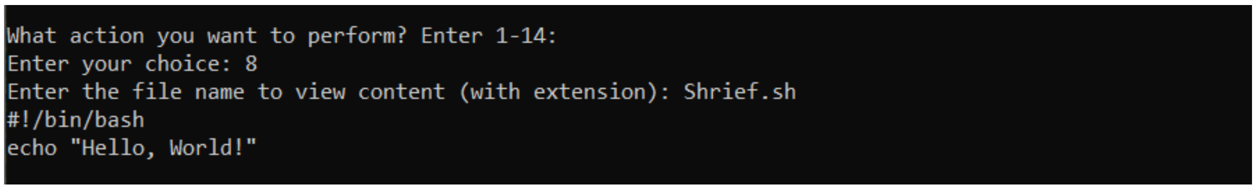
```
What action you want to perform? Enter 1-14:
Enter your choice: 6
Enter the file name to search (with extension): Shrief.sh
./Shrief.sh
File Shrief.sh found.
```

7- Details of Particular File:



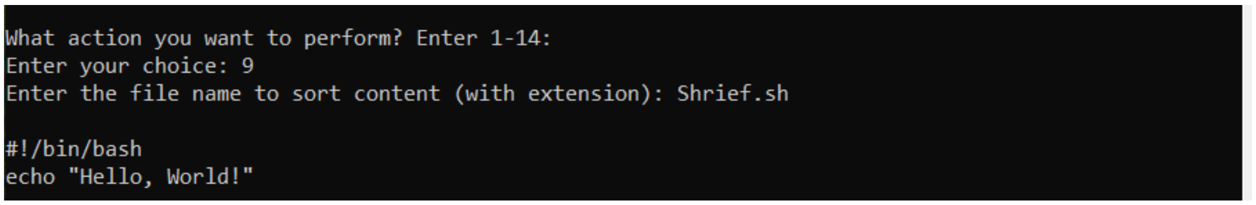
```
What action you want to perform? Enter 1-14:
Enter your choice: 7
Enter the file name to view details (with extension): Shrief.sh
  File: Shrief.sh
  Size: 34              Blocks: 8          IO Block: 4096   regular file
Device: 820h/2080d      Inode: 58882       Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-26 18:04:45.479259308 +0200
Modify: 2024-12-26 18:07:00.521643110 +0200
Change: 2024-12-26 18:07:00.521643110 +0200
 Birth: 2024-12-26 18:02:11.501452184 +0200
```

8- View Content of File:



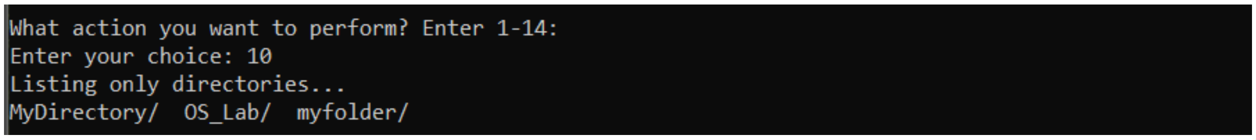
```
What action you want to perform? Enter 1-14:
Enter your choice: 8
Enter the file name to view content (with extension): Shrief.sh
#!/bin/bash
echo "Hello, World!"
```

9- Sort File Content:



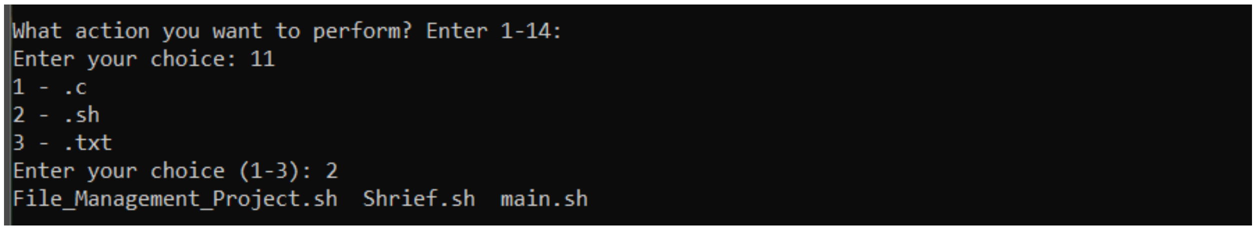
```
What action you want to perform? Enter 1-14:
Enter your choice: 9
Enter the file name to sort content (with extension): Shrief.sh

#!/bin/bash
echo "Hello, World!"
```

10- List only Directories (Folders):



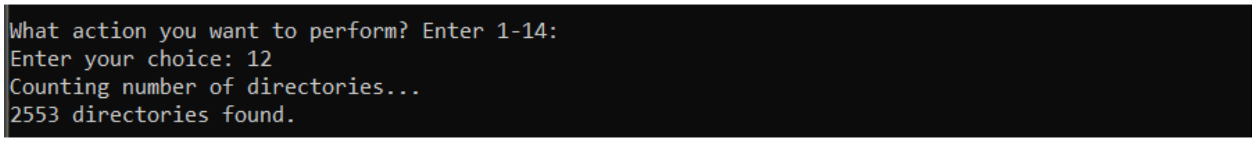
```
What action you want to perform? Enter 1-14:
Enter your choice: 10
Listing only directories...
MyDirectory/  OS_Lab/  myfolder/
```

11- List Files of Particular Extension:



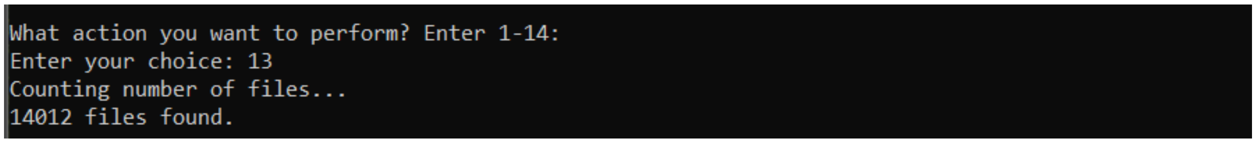
```
What action you want to perform? Enter 1-14:
Enter your choice: 11
1 - .c
2 - .sh
3 - .txt
Enter your choice (1-3): 2
File_Management_Project.sh  Shrief.sh  main.sh
```

12- Count Number of Directories:



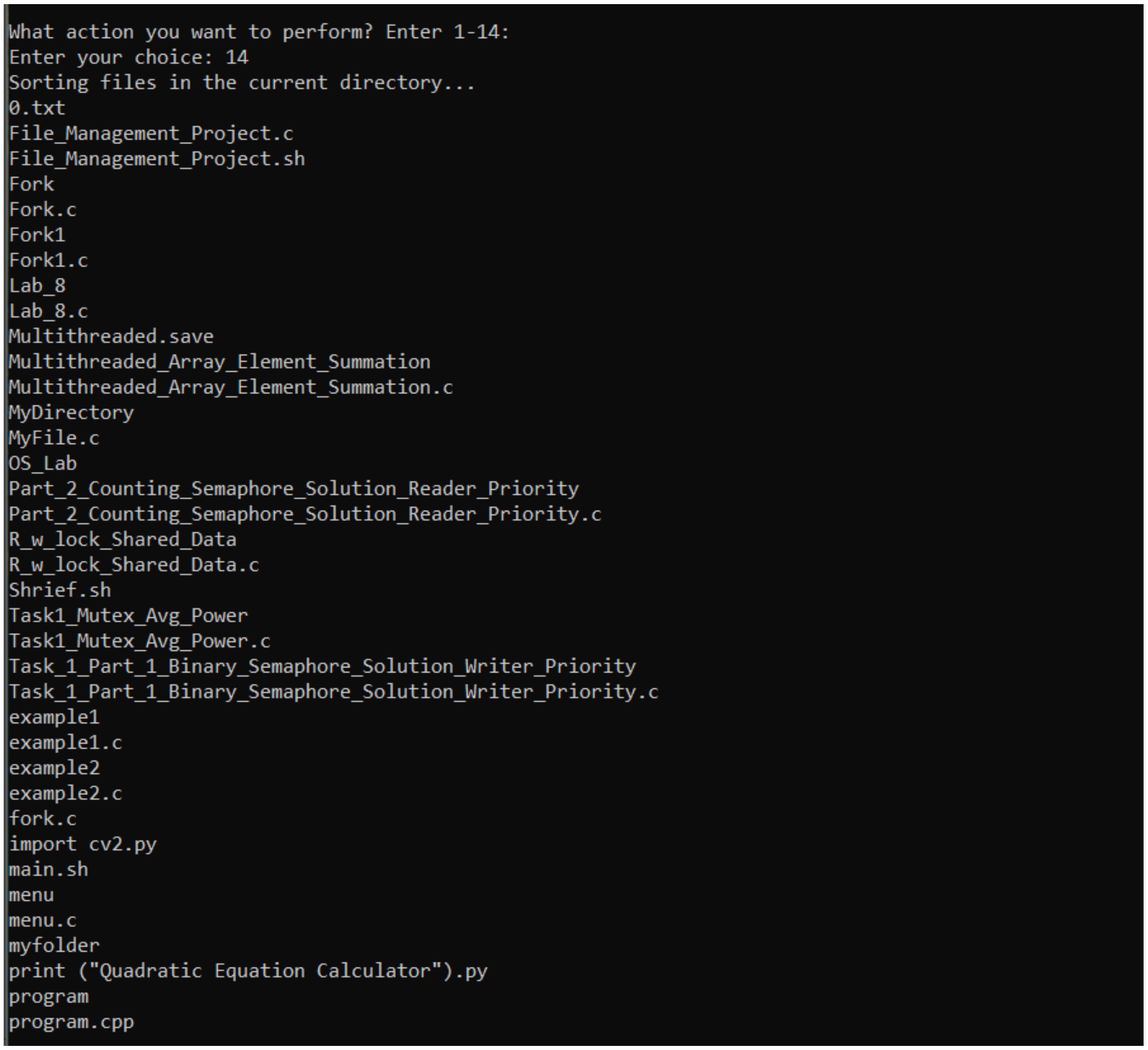
```
What action you want to perform? Enter 1-14:
Enter your choice: 12
Counting number of directories...
2553 directories found.
```

13- Count Number of Files:



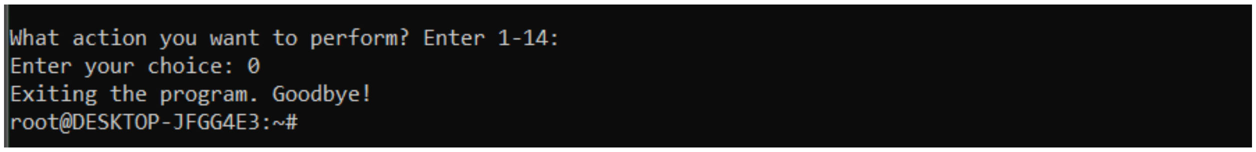
```
What action you want to perform? Enter 1-14:
Enter your choice: 13
Counting number of files...
14012 files found.
```

14- Sort Files in a Directory:



```
What action you want to perform? Enter 1-14:
Enter your choice: 14
Sorting files in the current directory...
0.txt
File_Management_Project.c
File_Management_Project.sh
Fork
Fork.c
Fork1
Fork1.c
Lab_8
Lab_8.c
Multithreaded.save
Multithreaded_Array_Element_Summation
Multithreaded_Array_Element_Summation.c
MyDirectory
MyFile.c
OS_Lab
Part_2_Counting_Semaphore_Solution_Reader_Priority
Part_2_Counting_Semaphore_Solution_Reader_Priority.c
R_w_lock_Shared_Data
R_w_lock_Shared_Data.c
Shrief.sh
Task1_Mutex_Avg_Power
Task1_Mutex_Avg_Power.c
Task_1_Part_1_Binary_Semaphore_Solution_Writer_Priority
Task_1_Part_1_Binary_Semaphore_Solution_Writer_Priority.c
example1
example1.c
example2
example2.c
fork.c
import cv2.py
main.sh
menu
menu.c
myfolder
print ("Quadratic Equation Calculator").py
program
program.cpp
```

0- Exit:



```
What action you want to perform? Enter 1-14:
Enter your choice: 0
Exiting the program. Goodbye!
root@DESKTOP-JFGG4E3:~#
```

➢ *Results and Discussion:*

Results:

- *Successfully implemented file operations including listing, creation, deletion, renaming, and editing.*
- *Achieved automation for directory-specific tasks such as counting and sorting files.*
- *Delivered a user-friendly menu-driven interface for seamless interaction.*

Discussion:

The project demonstrates the capability of combining Bash and C programming to handle file management efficiently. The system is scalable and can be extended with additional features such as file compression and encryption.

➢ *Conclusion and Future Work:*

Conclusion:

The project achieved its objective of providing a comprehensive file management system. The integration of menu-driven operations and automation enhances productivity and minimizes errors.

Functionality:

- The code creates a command-line interface (CLI) for basic file management tasks within the current directory.

- It offers options to: - List files and directories - Create new files (.c, .sh, or .txt) - Delete, rename, edit, search, view details, and sort files - Count files and directories

Key Features:

- User-friendly menu-driven interface

- Basic error handling for invalid input and file operations

- Iterative execution until user exits

- The code repeatedly compiles "project.c" within the loop, which may not be necessary for each iteration.

- The sleep commands are likely for visual effects and can be removed for efficiency.

The code could be enhanced with features like:

- Advanced file/directory operations (copy, move, permissions)

- Navigating different directories

- Customizable output formatting

-Improved error handling and validation

To sum it up:

The code provides a functional foundation for file management tasks in the terminal, but it has room for refinement and expansion.

- ***Future Work:***

- *Extend functionality to support file encryption and compression.*
- *Develop a web-based interface for remote file management.*
- *Optimize performance for handling large datasets.*

Functionality:

Expand file operations:

- Support additional file types (e.g., .pdf, .docx, .jpg).
- Implement file copying, moving, and linking.
- Allow for recursive directory operations (e.g., listing, searching).

Enhance file editing:

- Offer a choice of text editors (e.g., vim, Emacs).
- Enable basic editing features within the script (e.g., search and replace).

Incorporate advanced file management:

- Implement permissions and ownership management file.
- Provide file compression and archiving capabilities.

Add searching features:

- Allow searching within file contents.
- Support regular expressions for advanced pattern matching.

Implement user-friendly interface:
- Offer a menu-driven interface for easy navigation.
- Provide clear instructions and prompts.
- Validate user input to prevent errors.
Structure and Efficiency:
Refactor code:
- Break down large if statements into functions for better readability and maintainability.
- Use loops for repetitive tasks instead of manual repetition.
Optimize performance:
- Reduce unnecessary delays (e.g., sleep commands).
- Explore alternative approaches for file operations (e.g., using find or grep for searching).
Handle errors gracefully:
- Implement error trapping and meaningful error messages.
- Provide options for recovery or retry.
User Experience:
Personalize settings:
- Allow users to customize default file extensions, editor preferences, and other settings.
Provide context-sensitive help:
- Offer help options for specific functions and features.
Implement undo/redo functionality:
- Allow users to revert changes or repeat actions.
Security:
- Validate user input to prevent malicious code injection.
- Consider authentication and authorization for sensitive operations.
Cross-platform compatibility:
- Test the script on different operating systems to ensure compatibility.
Version control:
- Use a version control system (e.g., Git) to track changes and collaborate.

➢ ***References:***

1. *What is a File Management System? - Definition from Techopedia*

2. *Not found | Bynder*

3. *Bash Scripting Tutorial – Linux Shell Script and Command Line for ... https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script and-command-line-for-beginners/*

4. *File Management: A Comprehensive Guide | Canto*

5. *Best File Management Systems and Software in 2024! - Bit Blog*

6. *How to Write a Bash Script: A Simple Bash Scripting Tutorial | DataCamp https://www.datacamp.com/tutorial/how-to-write-bash-script-tutorial*

7. *Bash Scripting Tutorial - LinuxConfig.org https://linuxconfig.org/bash-scripting tutorial*

8. *Bash Scripting - Introduction to Bash and Bash Scripting - GeeksforGeeks https://www.geeksforgeeks.org/bash-scripting-introduction-to-bash-and-bash scripting/*

9. *Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems.*

10. *Nemeth, E., Snyder, G., Hein, T. R., & Whaley, B. (2017). UNIX and Linux System Administration Handbook.*

11. *Linux Manual Pages (man bash, man gcc, man ls).*