File Manager

A lite version of File Manager for Linux

Mini Project Report

By

Sai Suman Chitturi	1602-18-733-097
Praneeth Kapila	1602-18-733-116

Course: Operating Systems lab

Section: B.E. II/IV CSE-B

Semester: IV

Year: II

Department of Computer Science & Engineering

Vasavi College of Engineering

(Autonomous)

(Approved by A.I.C.T.E)

9-5-81, Ibrahimbagh, Hyderabad-31

2019-20

ACKNOWLEDGEMENT

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Mrs. T. Jalaja,** Department of Computer Science and Engineering, whose guidance was unforgettable and innovative ideas as well as her constructive suggestions have made the presentation of my thesis a grand success.

We are thankful to **Dr. T. Adilakshmi,** Head of Department (CSE), **Vasavi College of Engineering** for their help during our course work.

Finally, at last but not least express our heart full thanks to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete my seminar work successively.

Table of Contents

Abstract:

FILE MANAGEMENT SYSTEM

A Mini Project by: Chitturi Sai Suman, Praneeth Kapila

FILE MANAGEMENT SYSTEM:

The File Management System is a Linux application that is designed to manage functions and operations on Files. This application enables users to navigate a filesystem and interact with files and directories.

The application offers an extensive range of operations that can be performed on files. The operations include the ability to create a file/folder, descend a directory hierarchy, delete, modify, sort based on a required attribute, and several others.

The application is architectured to be user friendly and easy to use and yet doesn't compromise on the efficiency.

We hope our project will be beneficial to the users and serve their purpose.

Introduction:

FILE MANAGEMENT SYSTEM

A Mini Project by: Chitturi Sai Suman, Praneeth Kapila

FILE MANAGEMENT SYSTEM:

Our project, titled 'File Management System', is a Linux application that is designed to manage functions and operations on files.

The application is a fully functional file manager that is lightweight and extremely easy to use.

The application offers all the functionalities of a traditional file manager, from renaming files to sorting files, all this while being user friendly and easy to use.

Through this project, we intend to demonstrate the substantial capability of this file manager, which requires no special skill to use.

We hope our project will be beneficial to the users and serve their purpose.

Source Code:

```
#include<stdio.h>
#include<time.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<stdbool.h>
#include<ctype.h>
#include<limits.h>
#include<sys/types.h>
#include<errno.h>
#include<sys/wait.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<dirent.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<sys/shm.h>
#include<sys/sem.h>
#include<pthread.h>
#define and &&
#define or ||
struct file
{
    char* name;
    struct stat attribute;
};
struct file* files;
char conclusion[100];
int compare_1(const void* a, const void* b)//name of file
{
    struct file* f1 = (struct file*)a;
    struct file* f2 = (struct file*)b;
    return strcmp(f1->name, f2->name);
}
int compare_2(const void* a, const void* b)//time of last
access
{
    struct file* f1 = (struct file*)a;
    struct file* f2 = (struct file*)b;
    struct tm t1, t2;
    t1 = *(gmtime(&(f1->attribute.st_atim)));
    t2 = *(gmtime(&(f2->attribute.st_atim)));
    if (t1.tm_year != t2.tm_year)
        return t1.tm_year - t2.tm_year;
    else if (t1.tm_mon != t2.tm_mon)
        return t1.tm_mon - t2.tm_mon;
    else if (t1.tm_mday != t2.tm_mday)
```

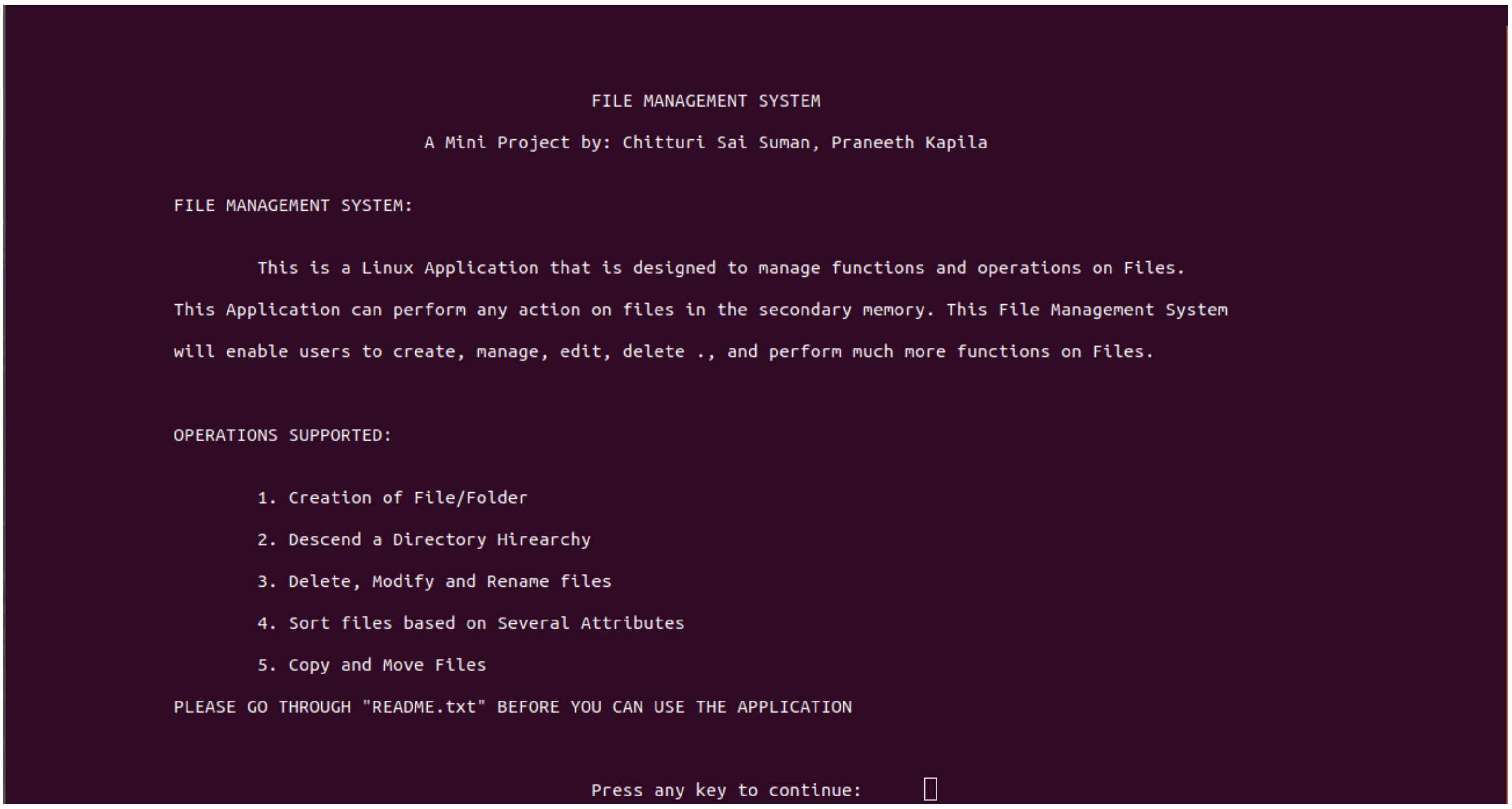
```
        return t1.tm_mday - t2.tm_mday;
    else if (t1.tm_hour != t2.tm_hour)
        return t1.tm_hour - t2.tm_hour;
    else if (t1.tm_min != t2.tm_min)
        return t1.tm_min - t2.tm_min;
    else
        return t1.tm_sec - t2.tm_sec;
}
int compare_3(const void* a, const void* b)//time of last
modification
{
    struct file* f1 = (struct file*)a;
    struct file* f2 = (struct file*)b;
    struct tm t1, t2;
    t1 = *(gmtime(&(f1->attribute.st_mtim)));
    t2 = *(gmtime(&(f2->attribute.st_mtim)));
    if (t1.tm_year != t2.tm_year)
        return t1.tm_year - t2.tm_year;
    else if (t1.tm_mon != t2.tm_mon)
        return t1.tm_mon - t2.tm_mon;
    else if (t1.tm_mday != t2.tm_mday)
        return t1.tm_mday - t2.tm_mday;
    else if (t1.tm_hour != t2.tm_hour)
        return t1.tm_hour - t2.tm_hour;
    else if (t1.tm_min != t2.tm_min)
        return t1.tm_min - t2.tm_min;
    else
        return t1.tm_sec - t2.tm_sec;
}
int compare_4(const void* a, const void* b)//size of file
{
    struct file* f1 = (struct file*)a;
    struct file* f2 = (struct file*)b;
    int size1 = f1->attribute.st_size;
    int size2 = f2->attribute.st_size;
    return size1 - size2;
}
void clrscr()
{
    system("clear");
}
void newline(int n)
{
    while (n--)
        printf("\n");
}
void tab(int n)
{
    while (n--)
        printf("\t");
}
```

```
void space(int n)
{
    while (n > 0)
    {
        printf(" ");
        n -= 1;
    }
}
void display_introduction()
{
    FILE* fptr = fopen("Introduction.txt", "r");
    char ch;
    clrscr();
    while (fscanf(fptr, "%c", &ch) != EOF)
        printf("%c", ch);
    fclose(fptr);
    char trash;
    newline(3);
    tab(7);
    printf("Press any key to continue:\t");
    system("/bin/stty raw");
    trash = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
}
int is_regular_file(const char* path)
{
    struct stat path_stat;
    stat(path, &path_stat);
    return S_ISREG(path_stat.st_mode);
}
void display_files(char* folder_name)
{
    clrscr();
    newline(3);
    tab(3);
    struct dirent* de;
    DIR* dr = opendir(folder_name);
    if (dr == NULL)
    {
        newline(2);
        tab(5);
        printf("Could not open %s directory", getcwd(NULL,
0));
        return;
    }
    printf("Current Directory:\t%s", getcwd(NULL, 0));
    newline(2);
    tab(5);
    printf("Files Present in Current Directory");
    newline(3);
    int number_of_columns = 3;
    int column_length = 40;
    int i = 0;
    tab(1);
    while ((de = readdir(dr)) != NULL)
    {
        if (strcmp(".", de->d_name) == 0 || strcmp("..", de-
>d_name) == 0)
            continue;
        printf("%s", de->d_name);
        space(column_length - (strlen(de->d_name)));
        i = (i + 1) % number_of_columns;
        if (i == 0)
        {
            newline(1);
            tab(1);
        }
    }
    closedir(dr);
    newline(3);
    tab(3);
    printf("Press any key to continue:");
    tab(1);
    char trash;
    system("/bin/stty raw");
    trash = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
}
int display_functions_available()
{
    newline(3);
    tab(2);
    printf("Choose one from the below Options");
    newline(2);
    tab(2);
    printf("0. Open any file to read");
    newline(2);
    tab(2);
    printf("1. Open any file to write");
    newline(2);
    tab(2);
    printf("2. Delete a file");
    newline(2);
    tab(2);
    printf("3. Rename a file");
    newline(2);
    tab(2);
    printf("4. Go to Parent Directory");
    newline(2);
    tab(2);
    printf("5. Go to Sub Directory");
    newline(2);
    tab(2);
    printf("6. Sort Files based on any Attribute");
    newline(2);
    tab(2);
    printf("7. Copy a file");
    newline(2);
    tab(2);
    printf("8. Move a file");
    newline(2);
    tab(2);
    printf("9. Preview the Files in Current Directory");
    newline(2);
    tab(2);
    printf("10. Create a folder with Specified Permissions");
    newline(2);
    tab(4);
    printf("Any other key to Quit");
    newline(2);
    tab(4);
    printf("Integral Choice:");
    tab(1);
    int choice;
    scanf("%d", &choice);
    char ch;
    scanf("%c", &ch);
    clrscr();
    return (choice <= 10 and choice >= 0) ? choice : -1;
}
void load_conclusion()
{
    FILE* fptr = fopen("Conclusion.txt", "r");
    char ch, trash;
    int i = 0;
    while (fscanf(fptr, "%c", &ch) != EOF)
        conclusion[i++] = ch;
    conclusion[i] = '\0';
    fclose(fptr);
    clrscr();
}
void print_conclusion()
{
    clrscr();
    int i = 0;
    while (conclusion[i] != '\0')
        printf("%c", conclusion[i++]);
    newline(5);
    tab(4);
    printf("Press any key to Quit...");
    tab(2);
    system("/bin/stty raw");
    char trash = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
}
void print_segment()
{
    for (int i = 1; i <= 142; i++)
        printf("*");
    printf("\n");
}
void case_0()
{
    clrscr();
    newline(3);
    tab(3);
    printf("Enter the name of the file to be opened for
reading:");
    tab(1);
    char name_of_file[40];
    gets(name_of_file);
    char trash;
    FILE* fptr = fopen(name_of_file, "r");
    clrscr();
    newline(2);
    tab(2);
    printf("Contents of %s", name_of_file);
    char ch;
    newline(1);
    print_segment();
    while (fscanf(fptr, "%c", &ch) != EOF)
        printf("%c", ch);
    fclose(fptr);
    newline(1);
    print_segment();
    newline(1);
    tab(2);
    printf("Press any key to Continue: ");
    system("/bin/stty raw");
    trash = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
}
void reprint_console(char text[], char mode)
{
    clrscr();
    if (mode == 'a')
    {
        int ind;
        newline(2);
        tab(2);
        printf("File Already Exists. File Opened in Append 
Mode.");
        newline(2);
        tab(2);
        printf("Existing Contents");
        newline(1);
        print_segment();
        ind = 0;
        while (text[ind] != '\0')
            printf("%c", text[ind++]);
        newline(1);
        print_segment();
        newline(1);
        tab(1);
        printf("Type to Append");
        newline(1);
        print_segment();
        ind = 0;
        while (text[ind] != '\0')
            printf("%c", text[ind++]);
    }
    else
    {
        int ind = 0;
        newline(2);
        tab(2);
        printf("File Created Successfully!. File Opened in 
Write Mode.");
        newline(2);
        tab(1);
        printf("Type to Write");
        newline(1);
        print_segment();
        while (text[ind] != '\0')
            printf("%c", text[ind++]);
    }
}
void case_1()
{
    clrscr();
    newline(3);
    tab(3);
    printf("Enter the name of the file to be opened for 
Writing:");
    tab(1);
    char name_of_file[40];
    gets(name_of_file);
    char trash;
    clrscr();
    FILE* fptr;
    char text[10000] = { '\0' };
    char ch;
    int ind, i;
    bool flag;
    fptr = fopen(name_of_file, "r");
    bool append_mode = true;
    if (fptr == NULL)
    {
        append_mode = false;
        fptr = fopen(name_of_file, "w");
    }
    else
    {
        ind = 0;
        while (fscanf(fptr, "%c", &ch) != EOF)
            text[ind++] = ch;
        text[ind] = '\0';
        fclose(fptr);
        fptr = fopen(name_of_file, "w");
        newline(2);
        tab(2);
        printf("File Already Exists. File Opened in Append
Mode.");
        newline(2);
        tab(2);
        printf("Existing Contents");
        newline(1);
        print_segment();
        ind = 0;
        while (text[ind] != '\0')
            printf("%c", text[ind++]);
        newline(1);
        print_segment();
        newline(1);
        tab(1);
        printf("Type to Append");
        newline(1);
        print_segment();
        ind = 0;
        while (text[ind] != '\0')
            printf("%c", text[ind++]);
        flag = true;
        i = ind;
        while (flag)
        {
            system("/bin/stty raw");
            ch = (getchar());
            system("/bin/stty cooked");
            if (ch == '\n' or ch == 10)
            {
                text[i++] = ch;
                text[i] = '\0';
            }
            else if (ch == '\b' or ch == 127)
            {
                if (i != 0)
                    i -= 1;
                text[i] = '\0';
            }
            else if (ch == 27)
            {
                flag = false;
                i = 0;
                while (text[i] != '\0')
                    fprintf(fptr, "%c", text[i++]);
            }
            else
            {
                text[i++] = ch;
                text[i] = '\0';
            }
            reprint_console(text, 'a');
        }
        fclose(fptr);
    }
    if (fptr == NULL)
    {
        newline(2);
        tab(2);
        printf("Unable to Open file.!");
        return;
    }
    if (append_mode == false and fptr != NULL)
    {
        ind = 0;
        text[ind] = '\0';
        newline(2);
        tab(2);
        printf("File Created Successfully!. File Opened in
Write Mode.");
        newline(2);
        tab(1);
        printf("Type to Write");
        newline(1);
        print_segment();
        flag = true;
        i = 0;
        while (flag)
        {
            system("/bin/stty raw");
            ch = (getchar());
            system("/bin/stty cooked");
            if (ch == '\n' or ch == 10)
```

```
                {
                    text[i++] = ch;
                    text[i] = '\0';
                }
                else if (ch == '\b' or ch == 127)
                {
                    if (i != 0)
                        i -= 1;
                    text[i] = '\0';
                }
                else if (ch == 27)
                {
                    flag = false;
                    reprint_console(text, 'a');
                    i = 0;
                    while (text[i] != '\0')
                        fprintf(fptr, "%c", text[i++]);
                }
                else
                {
                    text[i++] = ch;
                    text[i] = '\0';
                }
                reprint_console(text, 'w');
            }
            fclose(fptr);
        }
        newline(1);
        print_segment();
        newline(2);
        tab(1);
        printf("File written Successfully. Press any key to
continue...");
        system("/bin/stty raw");
        ch = (getchar());
        system("/bin/stty cooked");
        clrscr();
}
void case_2()
{
        clrscr();
        newline(3);
        tab(3);
        printf("Enter the name of the file to be Deleted:");
        tab(1);
        char name_of_file[40];
        gets(name_of_file);
        char trash;
        newline(3);
        tab(4);
        if (remove(name_of_file) == 0)
            printf("Removed Successfully!");
        else
            printf("Unable to delete the file!");
        newline(3);
        tab(3);
        printf("Press any key to Continue: ");
        system("/bin/stty raw");
        trash = tolower(getchar());
        system("/bin/stty cooked");
        clrscr();
}
void case_3()
{
        clrscr();
        char old_name[100];
        char new_name[100];
        newline(2);
        tab(2);
        printf("Enter the Name of the File to be Renamed: ");
        gets(old_name);
        newline(1);
        tab(2);
        printf("Enter New Name: ");
        gets(new_name);
        bool done = ((rename(old_name, new_name) == 0) ? true :
false);
        newline(2);
        tab(3);
        if (done)
            printf("Rename Successful!");
        else
            printf("Rename failed");
        newline(2);
        tab(3);
        printf("Press any key to Continue...");
        char ch;
        scanf("%c", &ch);
        system("/bin/stty raw");
        char trash = tolower(getchar());
        system("/bin/stty cooked");
        clrscr();
}
void case_4()
{
        clrscr();
        newline(2);
        tab(2);
        printf("Current Directory: %s", getcwd(NULL, 0));
        chdir("..");
        newline(2);
        tab(2);
        printf("Parent Directory: %s", getcwd(NULL, 0));
        newline(2);
        tab(2);
        printf("Reached Parent Directory!");
        newline(2);
        tab(2);
        printf("Current Directory: %s", getcwd(NULL, 0));
        newline(2);
        tab(3);
        printf("Press any key to Continue...");
        system("/bin/stty raw");
        char trash = tolower(getchar());
        system("/bin/stty cooked");
        clrscr();
        display_files(".");
}
void case_5()
{
        clrscr();
        clrscr();
        newline(4);
        tab(4);
        char folder_name[] = ".";
        struct dirent* de;
        DIR* dr = opendir(folder_name);
        if (dr == NULL)
        {
            newline(2);
            tab(5);
            printf("Could not open %s directory", getcwd(NULL,
0));
            return;
        }
        printf("Current Directory:\t%s", getcwd(NULL, 0));
        newline(2);
        tab(5);
        printf("Files Present in Current Directory");
        newline(3);
        int number_of_columns = 3;
        int column_length = 40;
        int i = 0;
        tab(1);
        while ((de = readdir(dr)) != NULL)
        {
            if (strcmp(".", de->d_name) == 0 || strcmp("..", de-
>d_name) == 0)
            continue;
            printf("%s", de->d_name);
            space(column_length - (strlen(de->d_name)));
            i = (i + 1) % number_of_columns;
            if (i == 0)
            {
                newline(1);
                tab(1);
            }
        }
        closedir(dr);
        newline(2);
        tab(2);
        printf("Enter folder name to move into: ");
        char sub_folder[50];
        gets(sub_folder);
        chdir(sub_folder);
        clrscr();
        display_files(".");
    }
    int load_properties()
    {
        struct dirent* de;
        DIR* dr = opendir(".");
        int number_of_files = 0;
        while ((de = readdir(dr)) != NULL)
        {
            if (strcmp(".", de->d_name) == 0 || strcmp("..", de-
    >d_name) == 0)
                continue;
            number_of_files++;
        }
        closedir(dr);
        files = (struct file*)malloc(number_of_files *
    sizeof(struct file));
        int i;
        dr = opendir(".");
        i = 0;
        while ((de = readdir(dr)) != NULL)
        {
            if (strcmp(".", de->d_name) == 0 || strcmp("..", de-
    >d_name) == 0)
                continue;
            files[i].name = (char*)malloc(strlen(de->d_name) *
    sizeof(char));
            strcpy(files[i].name, de->d_name);
            stat(files[i].name, &(files[i].attribute));
            i++;
        }
        return number_of_files;
    }
    void sort(char choice, int number_of_files)
    {
        switch (choice)
        {
```

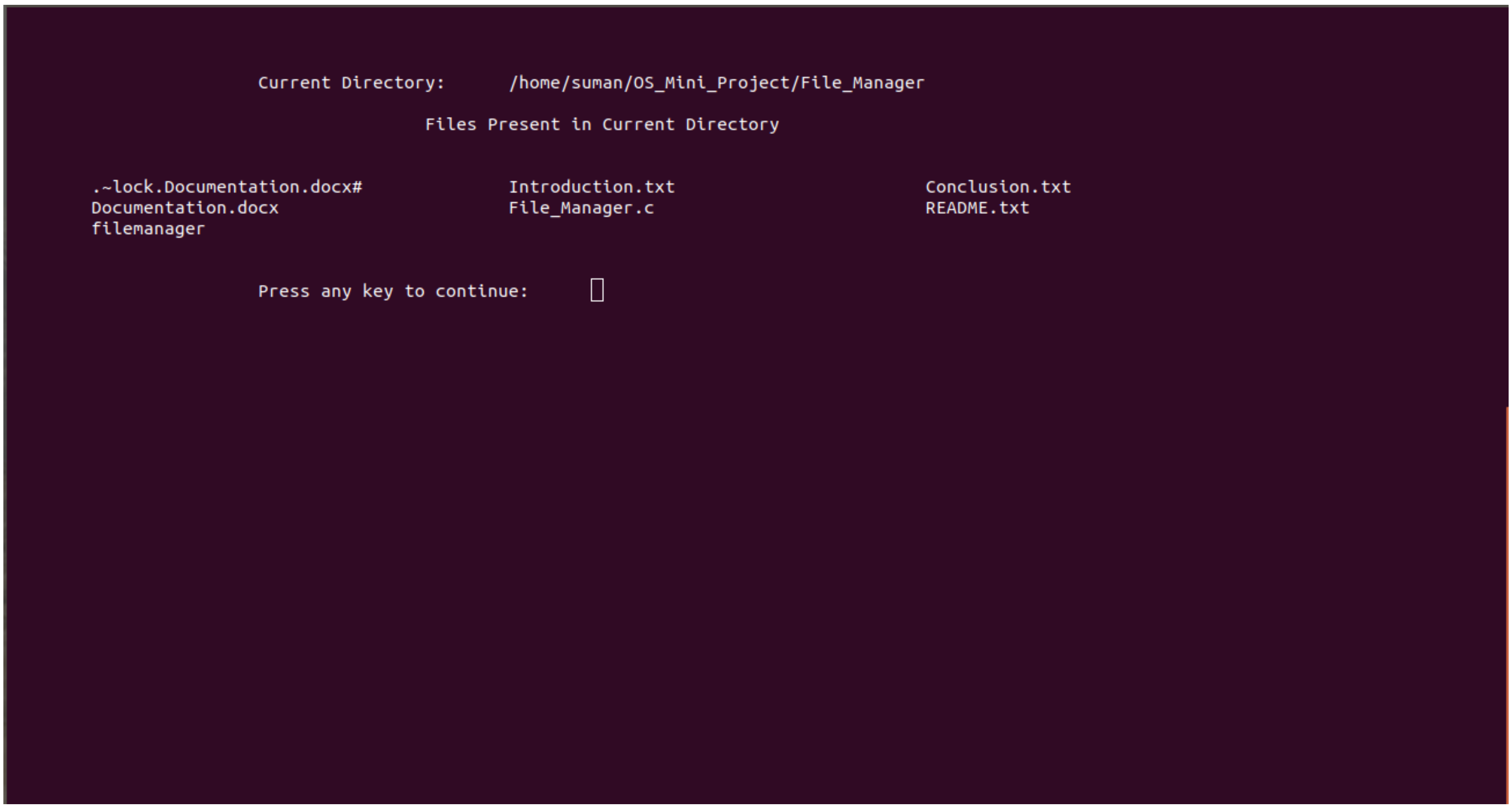
```
    case '1': qsort(files, number_of_files, sizeof(struct
file), compare_1); break;
    case '2': qsort(files, number_of_files, sizeof(struct
file), compare_2); break;
    case '3': qsort(files, number_of_files, sizeof(struct
file), compare_3); break;
    case '4': qsort(files, number_of_files, sizeof(struct
file), compare_4); break;
    }
}
void case_6()
{
    int number_of_files = load_properties();
    clrscr();
    newline(2);
    tab(2);
    printf("Select one of the Attributes to sort the files");
    newline(2);
    tab(3);
    printf("1. Name");
    newline(2);
    tab(3);
    printf("2. Date of last access");
    newline(2);
    tab(3);
    printf("3. Date of last modification");
    newline(2);
    tab(3);
    printf("4. Size of File");
    newline(2);
    tab(2);
    printf("Integral Choice:    ");
    char choice;
    system("/bin/stty raw");
    choice = (getchar());
    system("/bin/stty cooked");
    sort(choice, number_of_files);
    clrscr();
    newline(2);
    tab(2);
    printf("The files have been sorted According to ");
    switch (choice)
    {
    case '1': printf("Name of files"); break;
    case '2': printf("Date of Last Access"); break;
    case '3': printf("Date of Modification"); break;
    case '4': printf("Size of File"); break;
    }
    newline(2);
    tab(3);
    printf("Files after sorting");
    newline(2);
    tab(2);
    for (int i = 0; i < number_of_files; i++)
    {

        printf("%s", files[i].name);
        if (i % 2 == 0 and i != 0)
        {
            newline(1);
            tab(2);
        }
        else
            space(30 - strlen(files[i].name));
    }
    newline(2);
    tab(3);
    free(files);
    printf("Press any key to Continue...");
    system("/bin/stty raw");
    char trash = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
}
void case_7()
{
    char command[200] = "cp ";
    char source[80];
    char dest[80];
    clrscr();
    newline(2);
    tab(2);
    printf("Enter the File Name to be copied: ");
    gets(source);
    newline(2);
    tab(2);
    printf("Enter the Destination: ");
    gets(dest);
    int i;
    for (i = 0; source[i] != '\0'; i++)
        command[i + 3] = source[i];
    command[i + 3] = ' ';
    int j = i + 4;
    for (i = 0; dest[i] != '\0'; i++)
        command[j++] = dest[i];
    command[j] = '\0';
    system(command);
    newline(2);
    tab(3);
    printf("Copy Successful!");
    newline(2);
    tab(3);
    printf("Press any key to Continue...");
    system("/bin/stty raw");
    char trash = (getchar());
    system("/bin/stty cooked");
    clrscr();
}
void case_8()
{
    char command[200] = "mv ";
    char source[80];
    char dest[80];
    clrscr();
    newline(2);
    tab(2);
    printf("Enter the File Name to be Moved: ");
    gets(source);
    newline(2);
    tab(2);
    printf("Enter the Destination: ");
    gets(dest);
    int i;
    for (i = 0; source[i] != '\0'; i++)
        command[i + 3] = source[i];
    command[i + 3] = ' ';
    int j = i + 4;
    for (i = 0; dest[i] != '\0'; i++)
        command[j++] = dest[i];
    command[j] = '\0';
    system(command);
    newline(2);
    tab(3);
    printf("Move Operation Successful!");
    newline(2);
    tab(3);
    printf("Press any key to Continue...");
    system("/bin/stty raw");
    char trash = (getchar());
    system("/bin/stty cooked");
    clrscr();
}
void case_9()
{
    clrscr();
    display_files(".");
    clrscr();
}
void case_10()
{
    char folder_name[32];
    int permissions = 0;
    char bits[11];
    clrscr();
    display_files(".");
    clrscr();
    newline(3);
    tab(3);
    printf("Enter the Name of the Folder:\t");
    scanf("%s", folder_name);
    newline(2);
    tab(3);
    printf("Enter 10 bits corresponding to Permissions: ");
    scanf("%s", bits);
    for (int i = strlen(bits) - 1; i >= 0; i--)
        permissions += (bits[i] - '0') * pow(2, strlen(bits) -
(i + 1));
    newline(2);
    tab(3);
    printf("Permissions: %d\n", permissions);
    newline(2);
    tab(2);
    if (!mkdir(folder_name, permissions))
    {
        printf("Folder Created Successfully!");
    }
    else
    {
        printf("Folder Creation Failed!");
    }
    newline(2);
    tab(3);
    char trash;
    scanf("%c", &trash);
    printf("Press any key to Continue...");
    system("/bin/stty raw");
    char ch = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
}
bool case_quit()
{
    newline(1);
    tab(4);
    printf("Are You Sure to Quit the Application(Y/N): ");
    system("/bin/stty raw");
    char ch = tolower(getchar());
    system("/bin/stty cooked");
    clrscr();
    return (ch == 'y' or ch == 'Y');
}
int main()
{
    display_introduction();
    display_files(".");
    load_conclusion();
    bool flag = true;
    while (flag)
    {
        int option = display_functions_available();
        switch (option)
        {
        case 0: case_0();
            break;
        case 1: case_1();
            break;
        case 2: case_2();
            break;
        case 3: case_3();
            break;
        case 4: case_4();
            break;
        case 5: case_5();
            break;
        case 6: case_6();
            break;
        case 7: case_7();
            break;
        case 8: case_8();
            break;
        case 9: case_9();
            break;
        case 10: case_10();
            break;
        case -1: if (case_quit())
            flag = false;
            break;
        }
    }
    print_conclusion();
    return 0;
}
```

Screenshots:

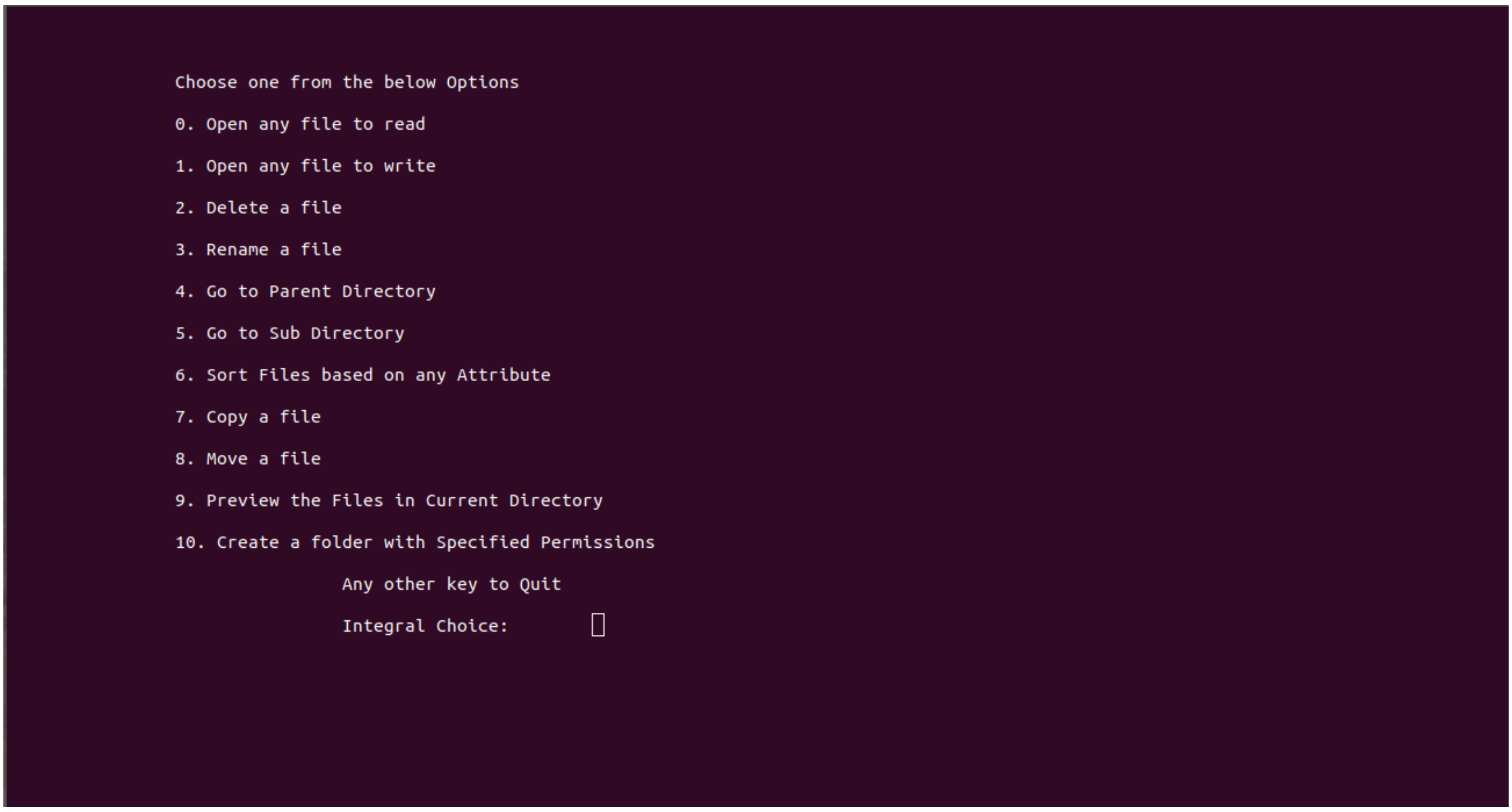


```
                                        FILE MANAGEMENT SYSTEM

                          A Mini Project by: Chitturi Sai Suman, Praneeth Kapila

FILE MANAGEMENT SYSTEM:

        This is a Linux Application that is designed to manage functions and operations on Files.

This Application can perform any action on files in the secondary memory. This File Management System

will enable users to create, manage, edit, delete ., and perform much more functions on Files.


OPERATIONS SUPPORTED:

        1. Creation of File/Folder

        2. Descend a Directory Hirearchy

        3. Delete, Modify and Rename files

        4. Sort files based on Several Attributes

        5. Copy and Move Files

PLEASE GO THROUGH "README.txt" BEFORE YOU CAN USE THE APPLICATION


                                        Press any key to continue:
```

Screenshot: Initial Interface

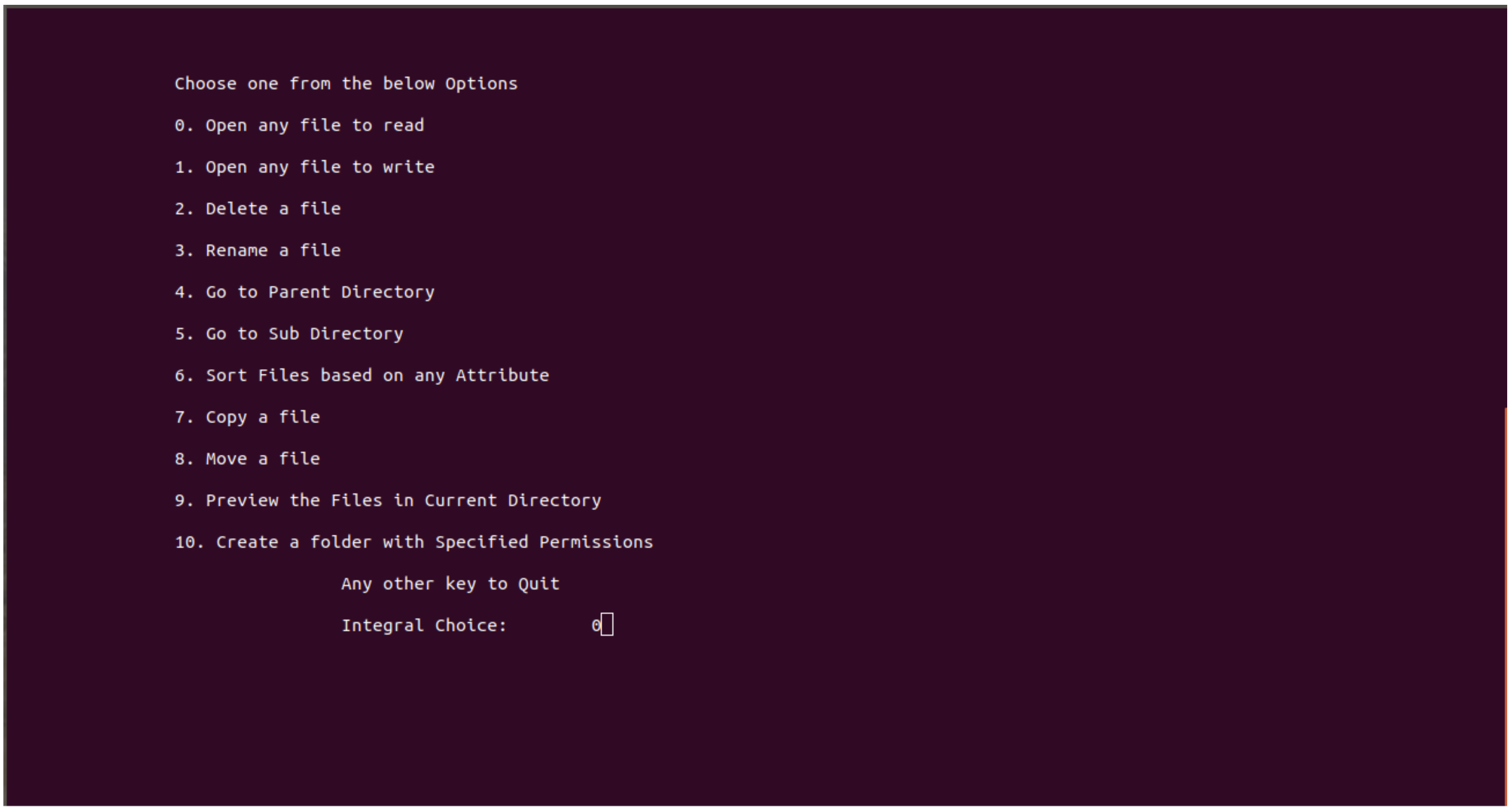


```
        Current Directory:      /home/suman/OS_Mini_Project/File_Manager

                        Files Present in Current Directory

.~lock.Documentation.docx#              Introduction.txt                        Conclusion.txt
Documentation.docx                      File_Manager.c                          README.txt
filemanager

        Press any key to continue:
```

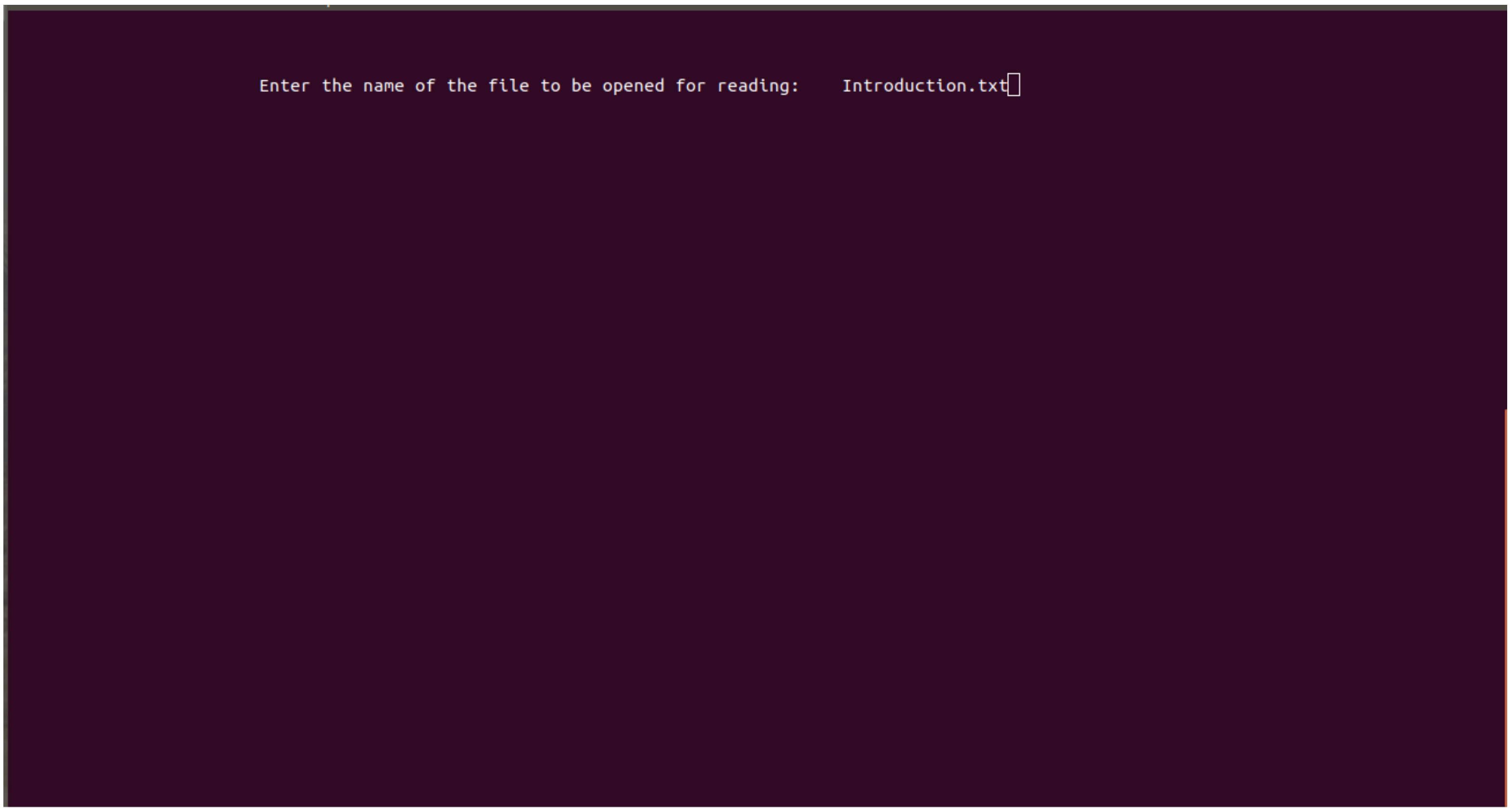


```
Choose one from the below Options
0. Open any file to read
1. Open any file to write
2. Delete a file
3. Rename a file
4. Go to Parent Directory
5. Go to Sub Directory
6. Sort Files based on any Attribute
7. Copy a file
8. Move a file
9. Preview the Files in Current Directory
10. Create a folder with Specified Permissions
                Any other key to Quit
                Integral Choice:
```

Screenshot: Preview of Functions

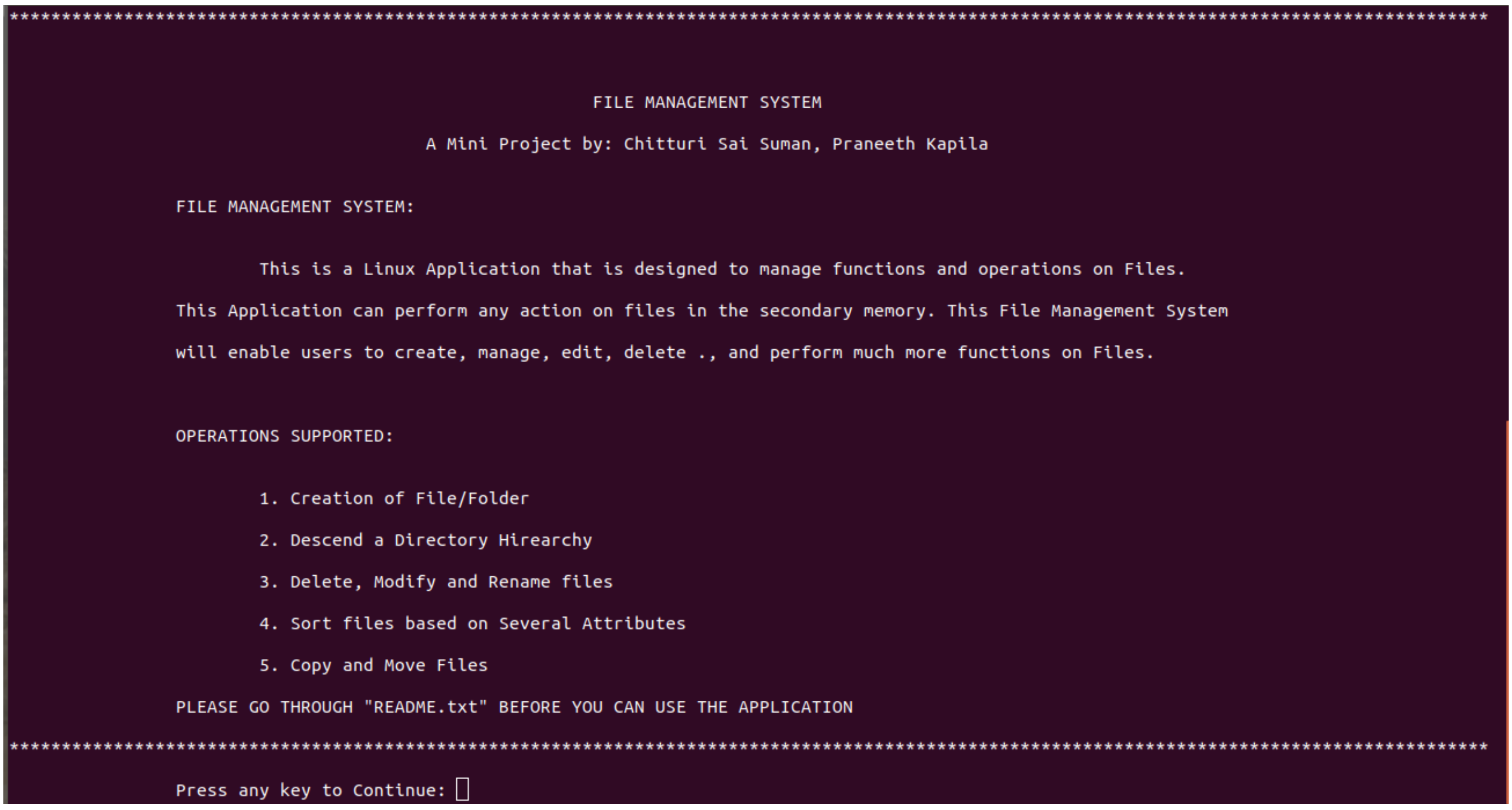


```
Choose one from the below Options
0. Open any file to read
1. Open any file to write
2. Delete a file
3. Rename a file
4. Go to Parent Directory
5. Go to Sub Directory
6. Sort Files based on any Attribute
7. Copy a file
8. Move a file
9. Preview the Files in Current Directory
10. Create a folder with Specified Permissions
                Any other key to Quit
                Integral Choice:        0
```

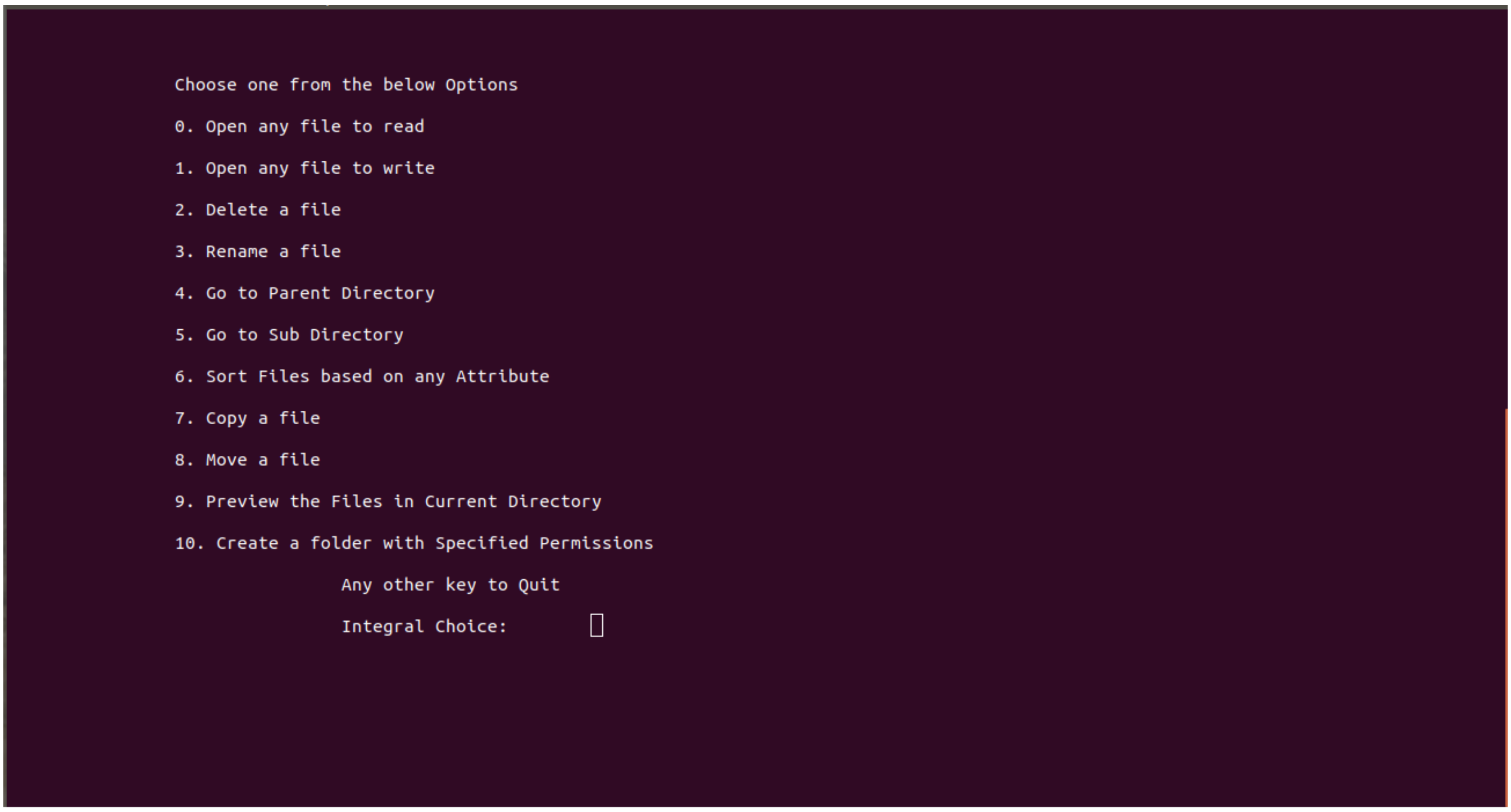


```
        Enter the name of the file to be opened for reading:    Introduction.txt
```

Screenshot: 1. Prompt for opening a file for reading. 2. Contents of File displayed

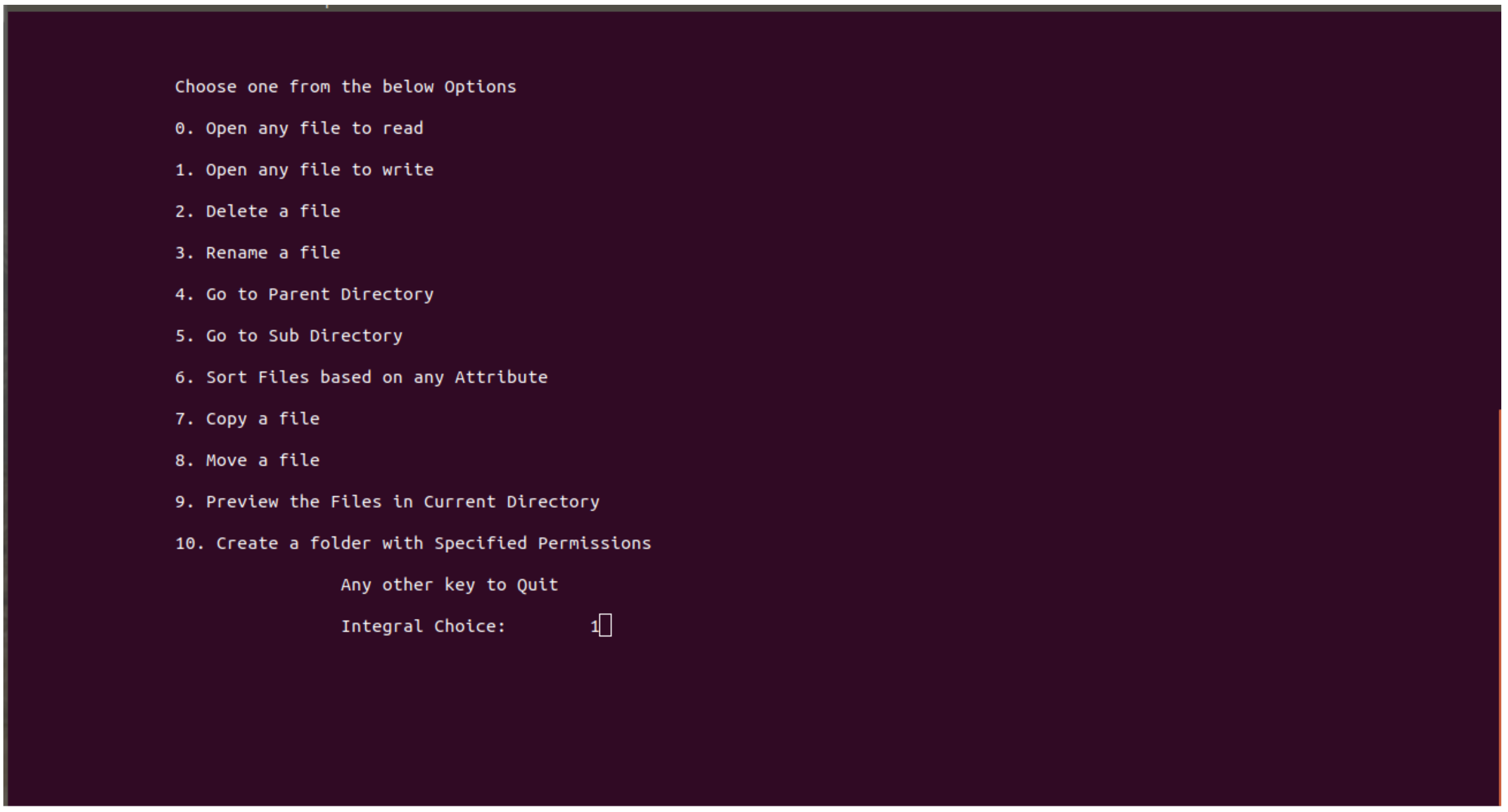


```
************************************************************************************************************************************************

                                                        FILE MANAGEMENT SYSTEM

                                        A Mini Project by: Chitturi Sai Suman, Praneeth Kapila

                FILE MANAGEMENT SYSTEM:

                        This is a Linux Application that is designed to manage functions and operations on Files.

                This Application can perform any action on files in the secondary memory. This File Management System

                will enable users to create, manage, edit, delete ., and perform much more functions on Files.


                OPERATIONS SUPPORTED:

                        1. Creation of File/Folder

                        2. Descend a Directory Hirearchy

                        3. Delete, Modify and Rename files

                        4. Sort files based on Several Attributes

                        5. Copy and Move Files

                PLEASE GO THROUGH "README.txt" BEFORE YOU CAN USE THE APPLICATION

************************************************************************************************************************************************

                Press any key to Continue:
```

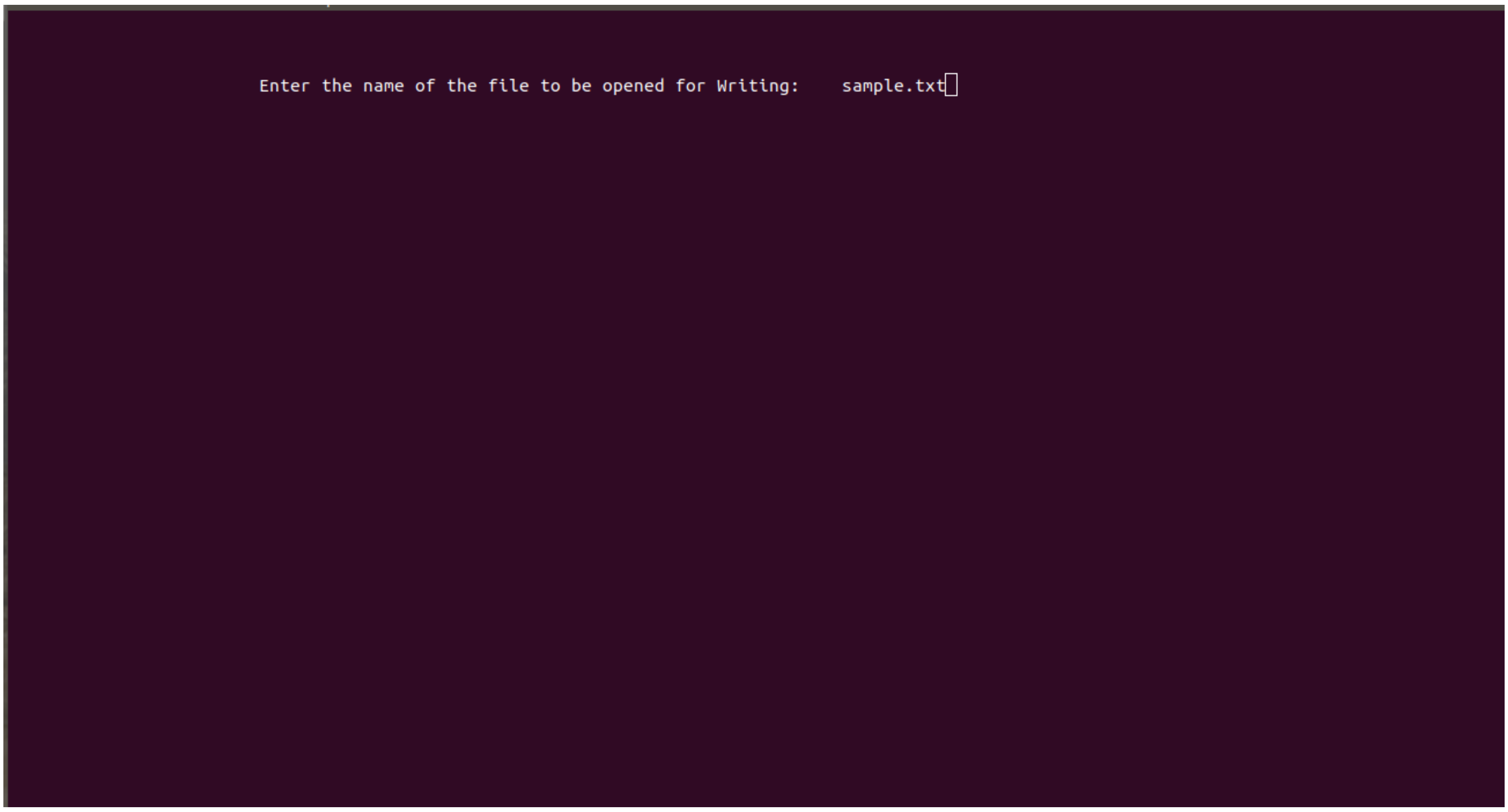


```
Choose one from the below Options

0. Open any file to read

1. Open any file to write

2. Delete a file

3. Rename a file

4. Go to Parent Directory

5. Go to Sub Directory

6. Sort Files based on any Attribute

7. Copy a file

8. Move a file

9. Preview the Files in Current Directory

10. Create a folder with Specified Permissions

                Any other key to Quit

                Integral Choice:
```

<u>Screenshot: 1. Functions displayed. 2. User Selecting Option 2</u>

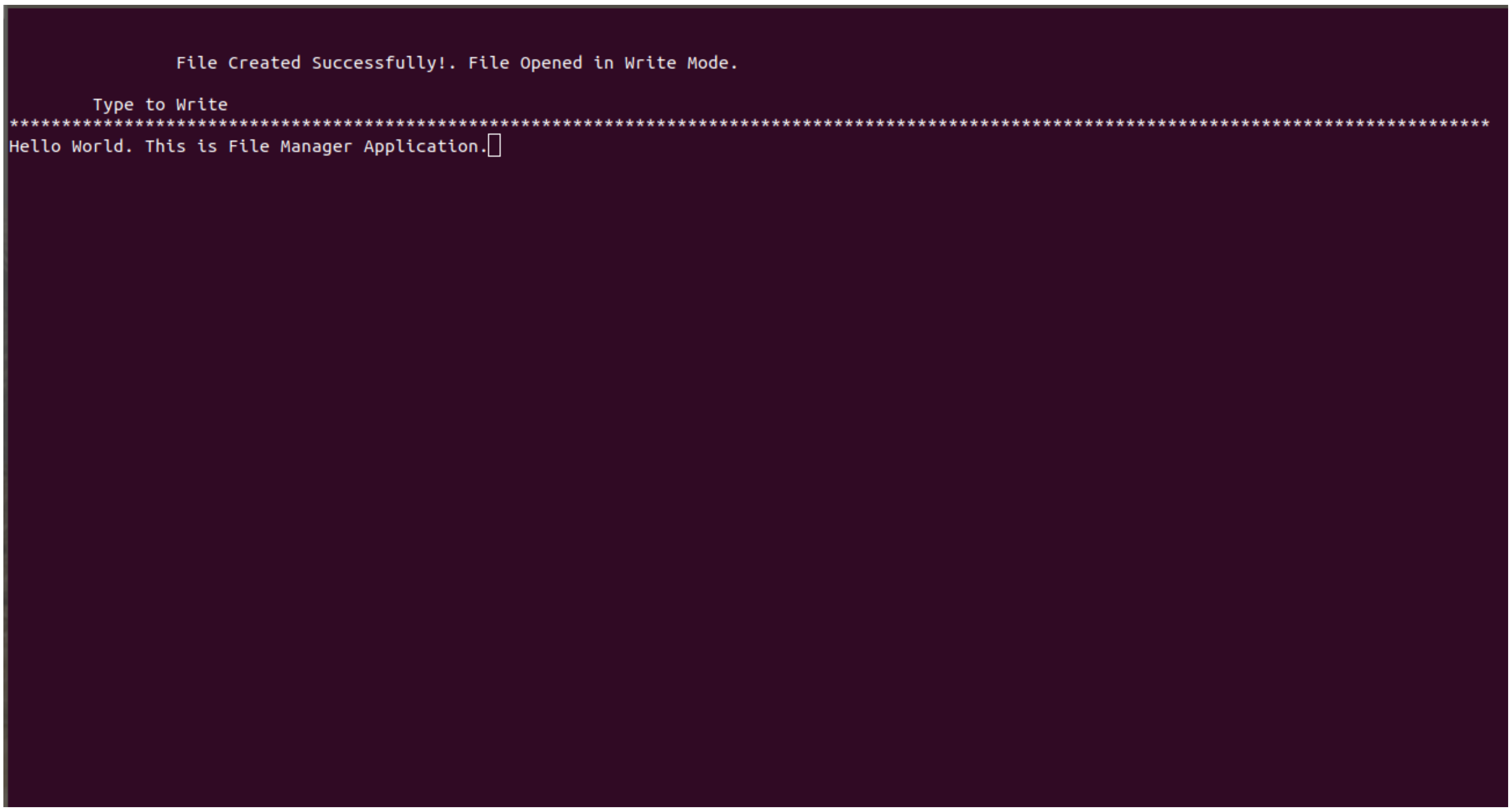


```
Choose one from the below Options

0. Open any file to read

1. Open any file to write

2. Delete a file

3. Rename a file

4. Go to Parent Directory

5. Go to Sub Directory

6. Sort Files based on any Attribute

7. Copy a file

8. Move a file

9. Preview the Files in Current Directory

10. Create a folder with Specified Permissions

                Any other key to Quit

                Integral Choice:        1
```

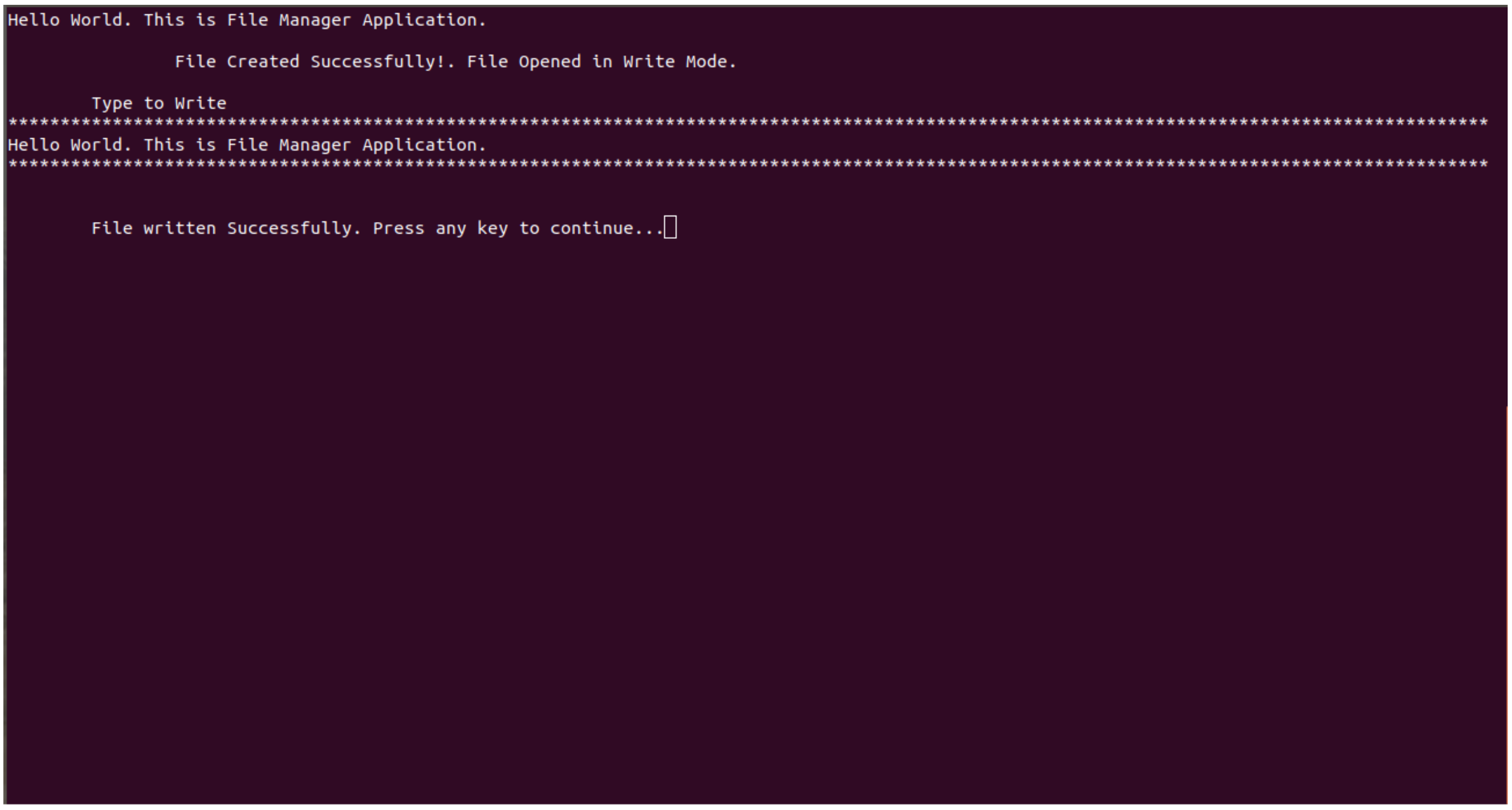


```
                        Enter the name of the file to be opened for Writing:    sample.txt
```

Screenshot: File opened for Writing

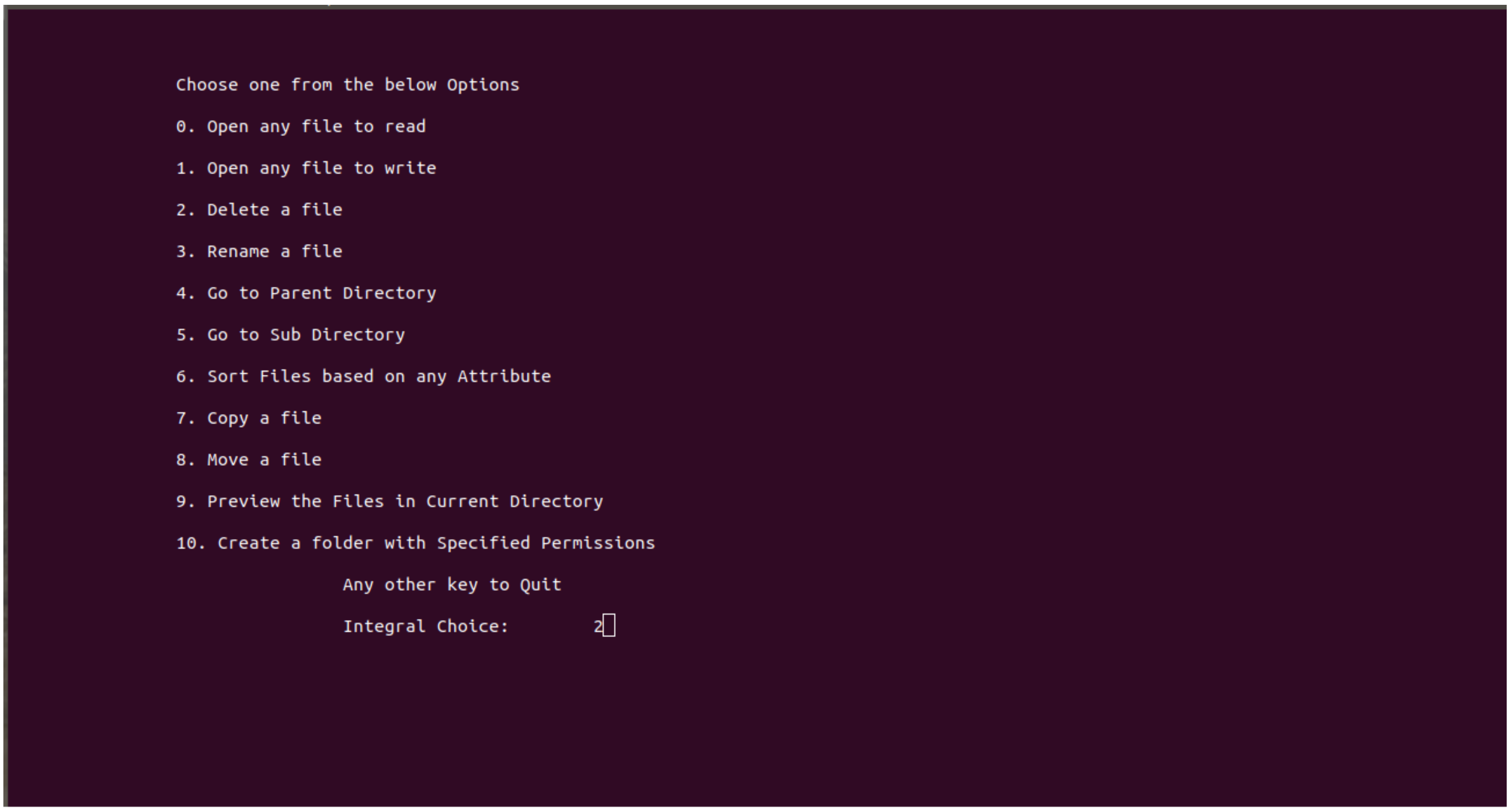


```
                File Created Successfully!. File Opened in Write Mode.

        Type to Write
**********************************************************************************************************************************************
Hello World. This is File Manager Application.
```

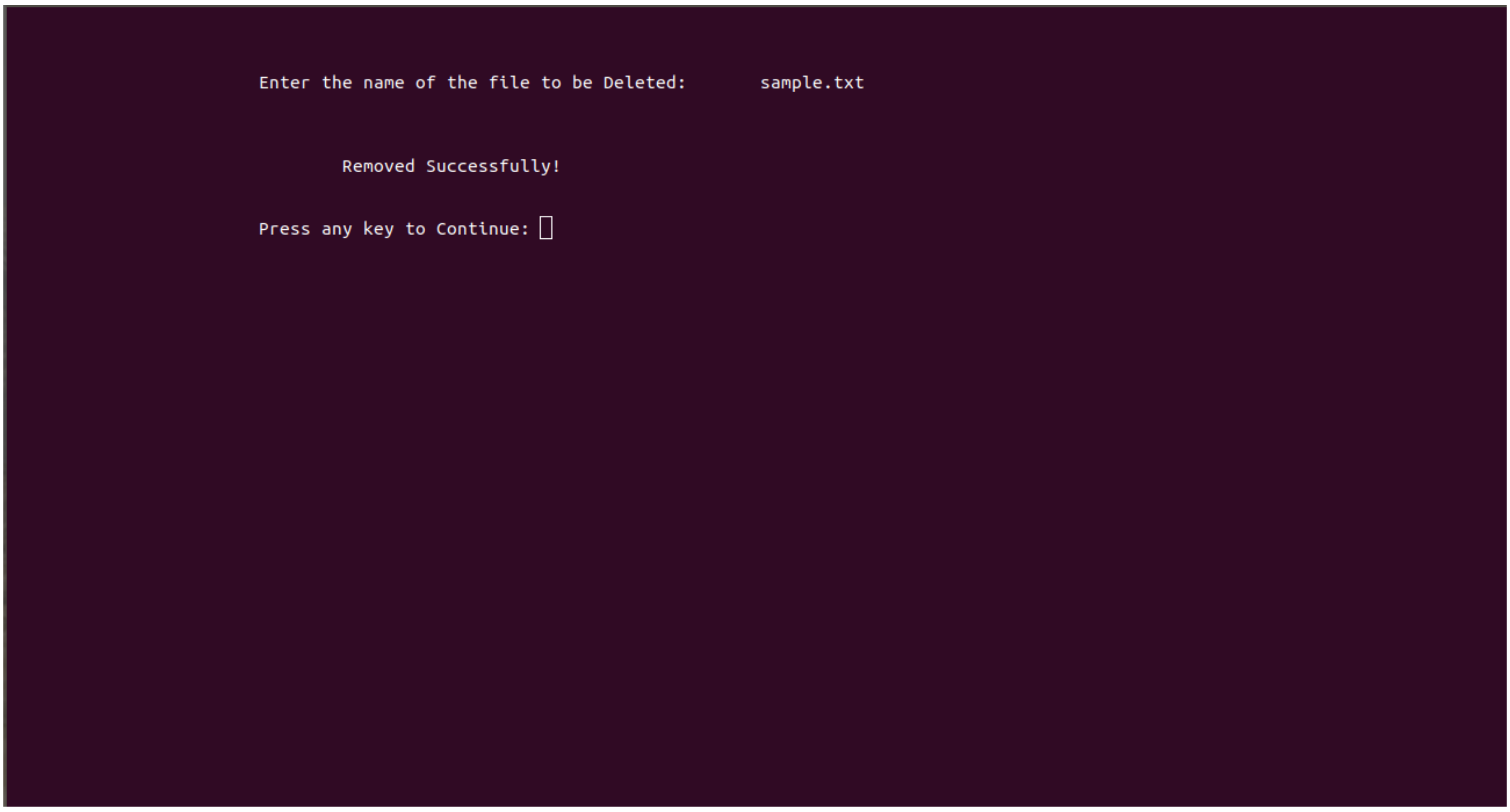


```
Hello World. This is File Manager Application.

                File Created Successfully!. File Opened in Write Mode.

        Type to Write
************************************************************************************************************************************************************
Hello World. This is File Manager Application.
************************************************************************************************************************************************************


        File written Successfully. Press any key to continue...
```

<u>Screenshot: 1. File written successfully. 2. Functions displayed</u>

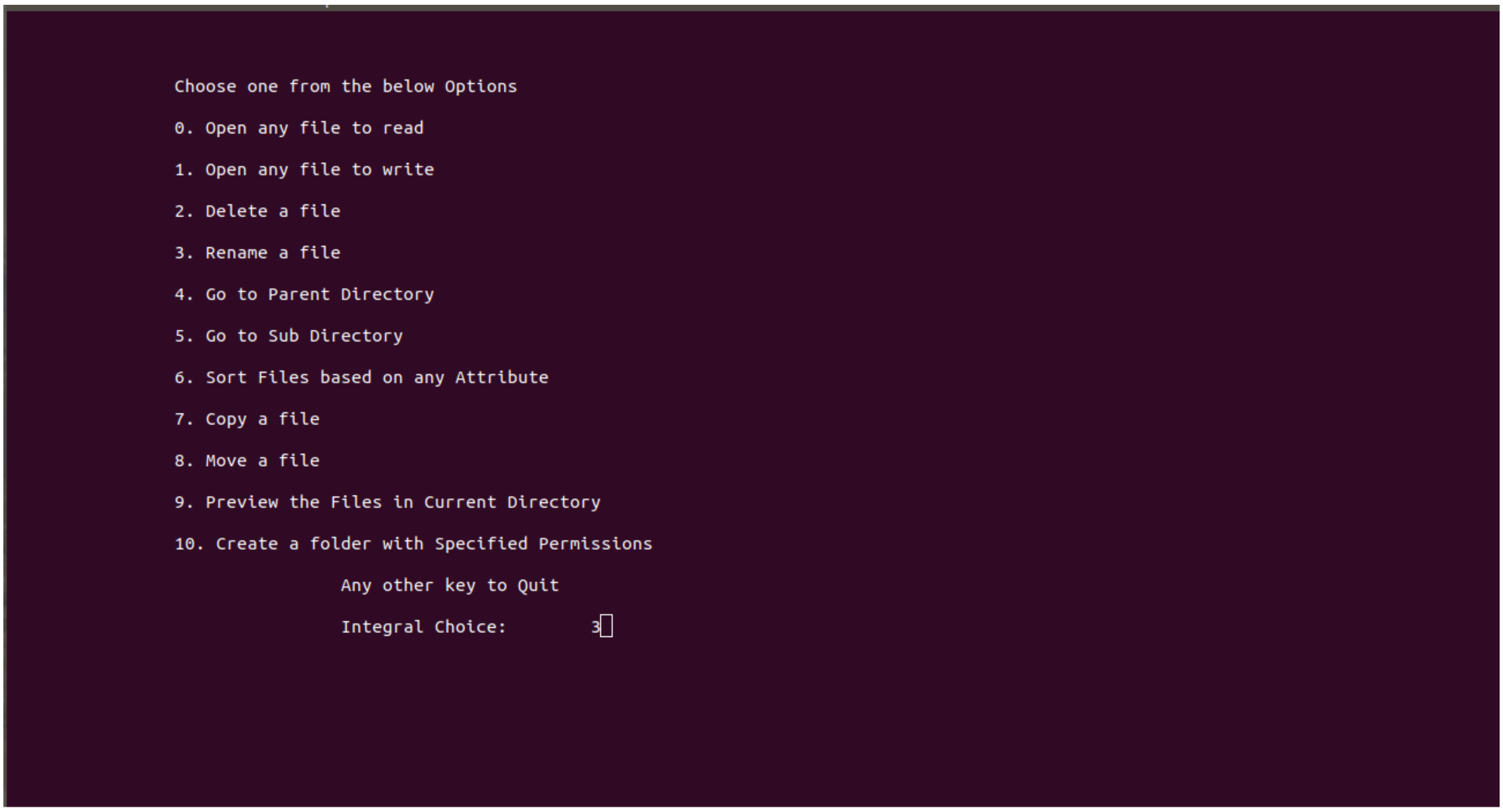


```

                Choose one from the below Options

                0. Open any file to read

                1. Open any file to write

                2. Delete a file

                3. Rename a file

                4. Go to Parent Directory

                5. Go to Sub Directory

                6. Sort Files based on any Attribute

                7. Copy a file

                8. Move a file

                9. Preview the Files in Current Directory

                10. Create a folder with Specified Permissions

                                Any other key to Quit

                                Integral Choice:        2
```

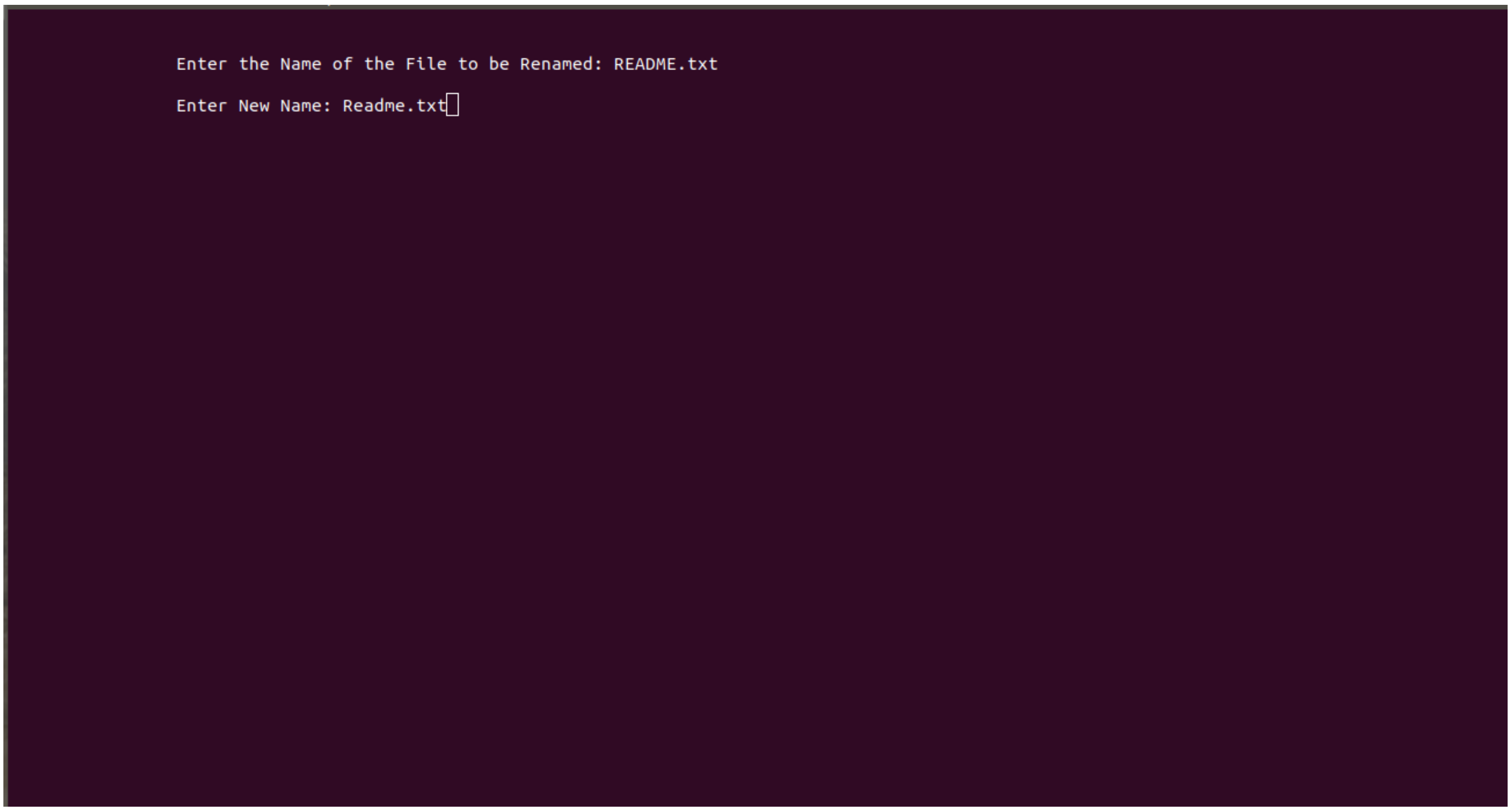


```
Enter the name of the file to be Deleted:       sample.txt


        Removed Successfully!

Press any key to Continue: 
```

Screenshot: File deleted successfully

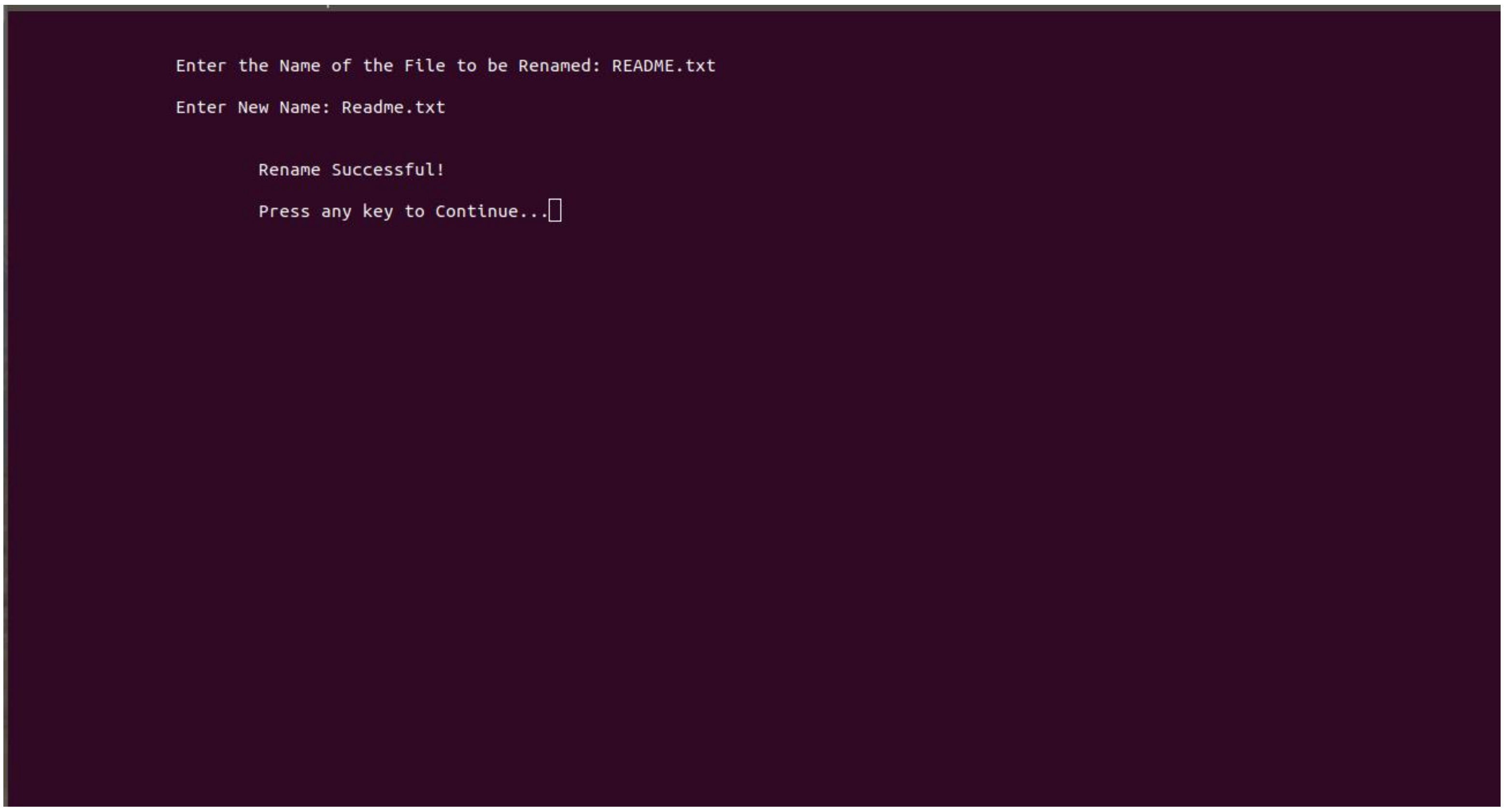


```
Choose one from the below Options

0. Open any file to read

1. Open any file to write

2. Delete a file

3. Rename a file

4. Go to Parent Directory

5. Go to Sub Directory

6. Sort Files based on any Attribute

7. Copy a file

8. Move a file

9. Preview the Files in Current Directory

10. Create a folder with Specified Permissions

                Any other key to Quit

                Integral Choice:        3
```

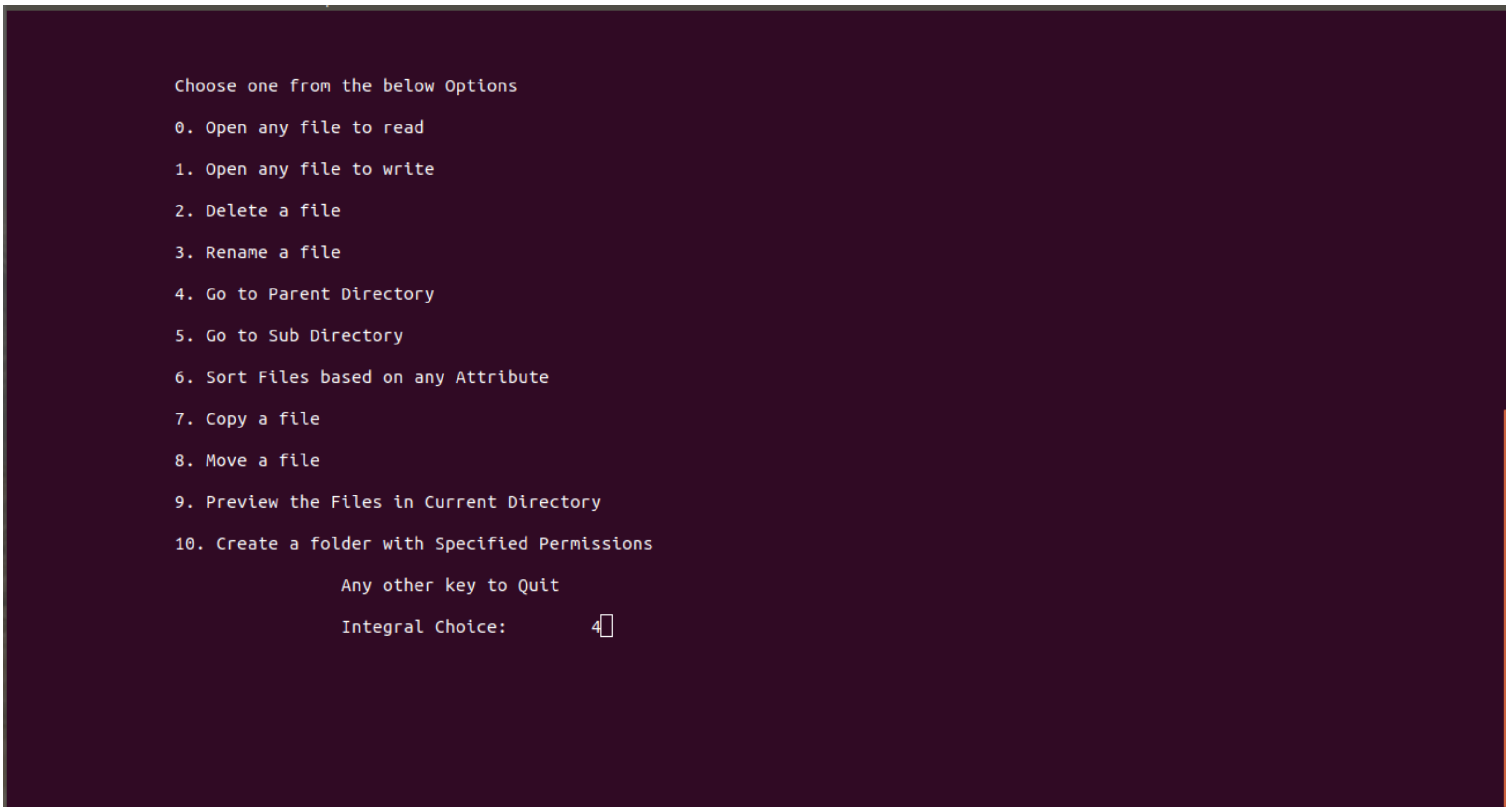


```
Enter the Name of the File to be Renamed: README.txt

Enter New Name: Readme.txt
```

Screenshot: File Renamed Successfully

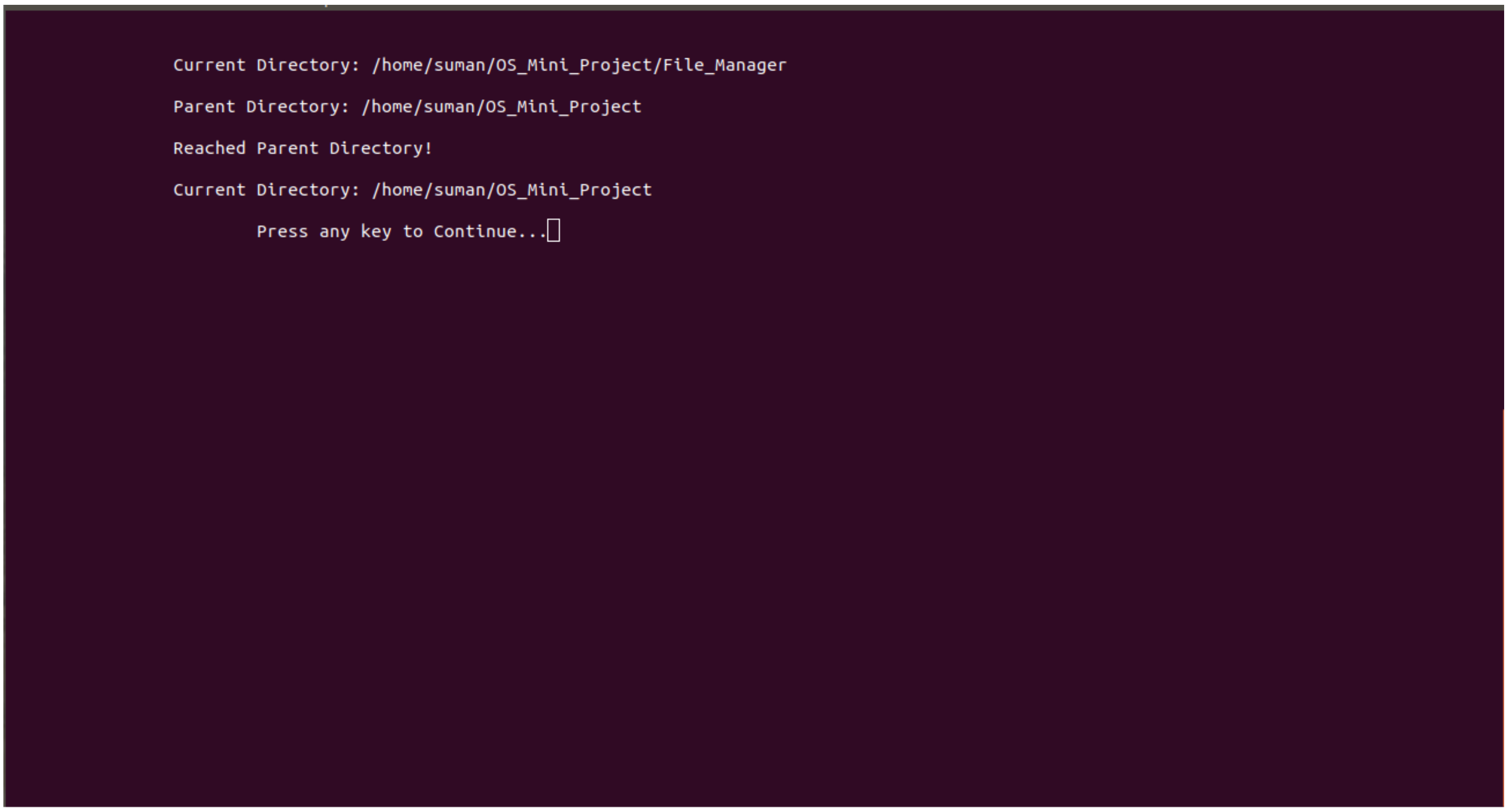


```
Enter the Name of the File to be Renamed: README.txt

Enter New Name: Readme.txt

        Rename Successful!

        Press any key to Continue...
```

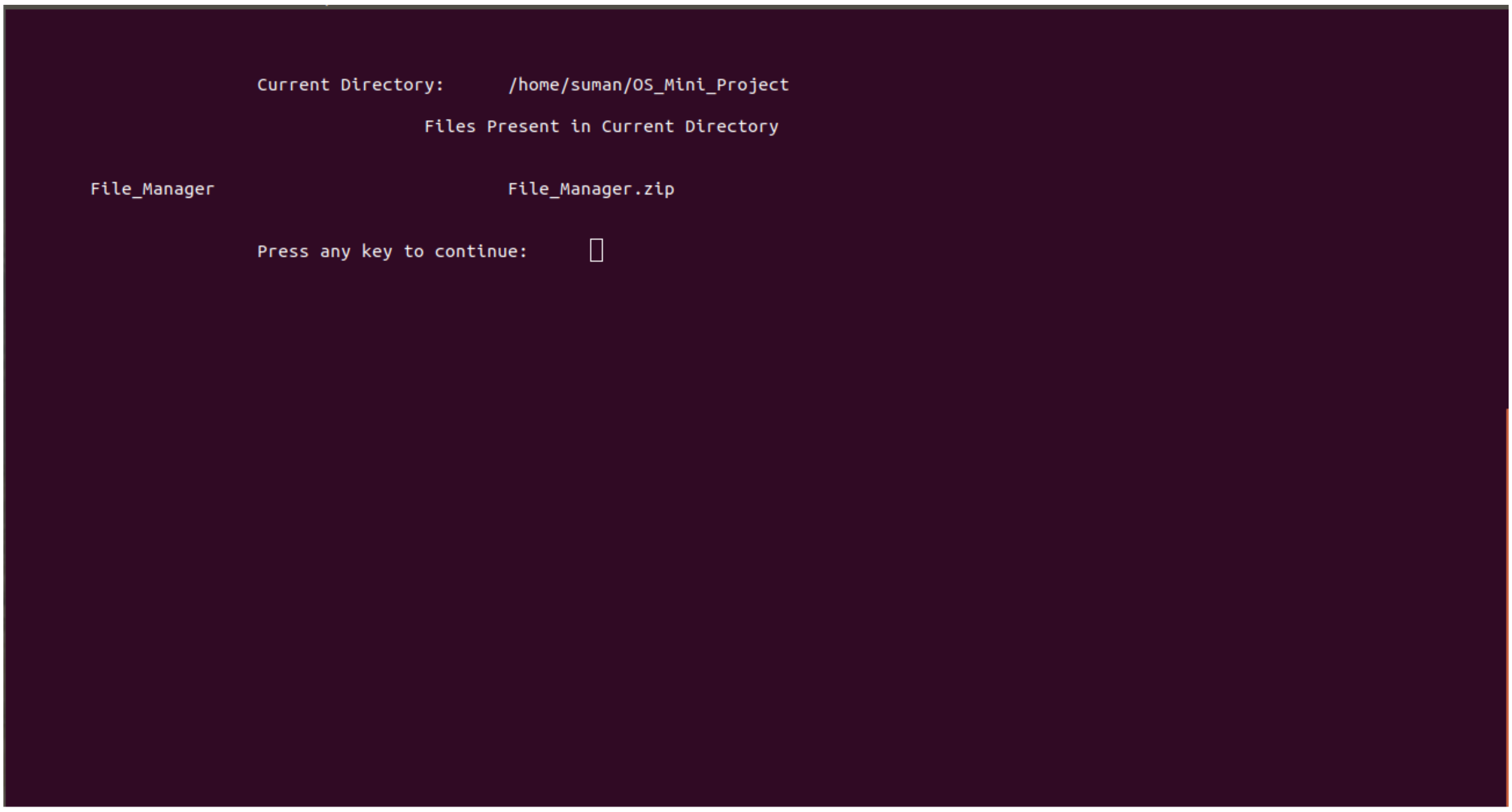


```
Choose one from the below Options

0. Open any file to read

1. Open any file to write

2. Delete a file

3. Rename a file

4. Go to Parent Directory

5. Go to Sub Directory

6. Sort Files based on any Attribute

7. Copy a file

8. Move a file

9. Preview the Files in Current Directory

10. Create a folder with Specified Permissions

                Any other key to Quit

                Integral Choice:        4
```

Screenshot: Navigating to Parent Directory

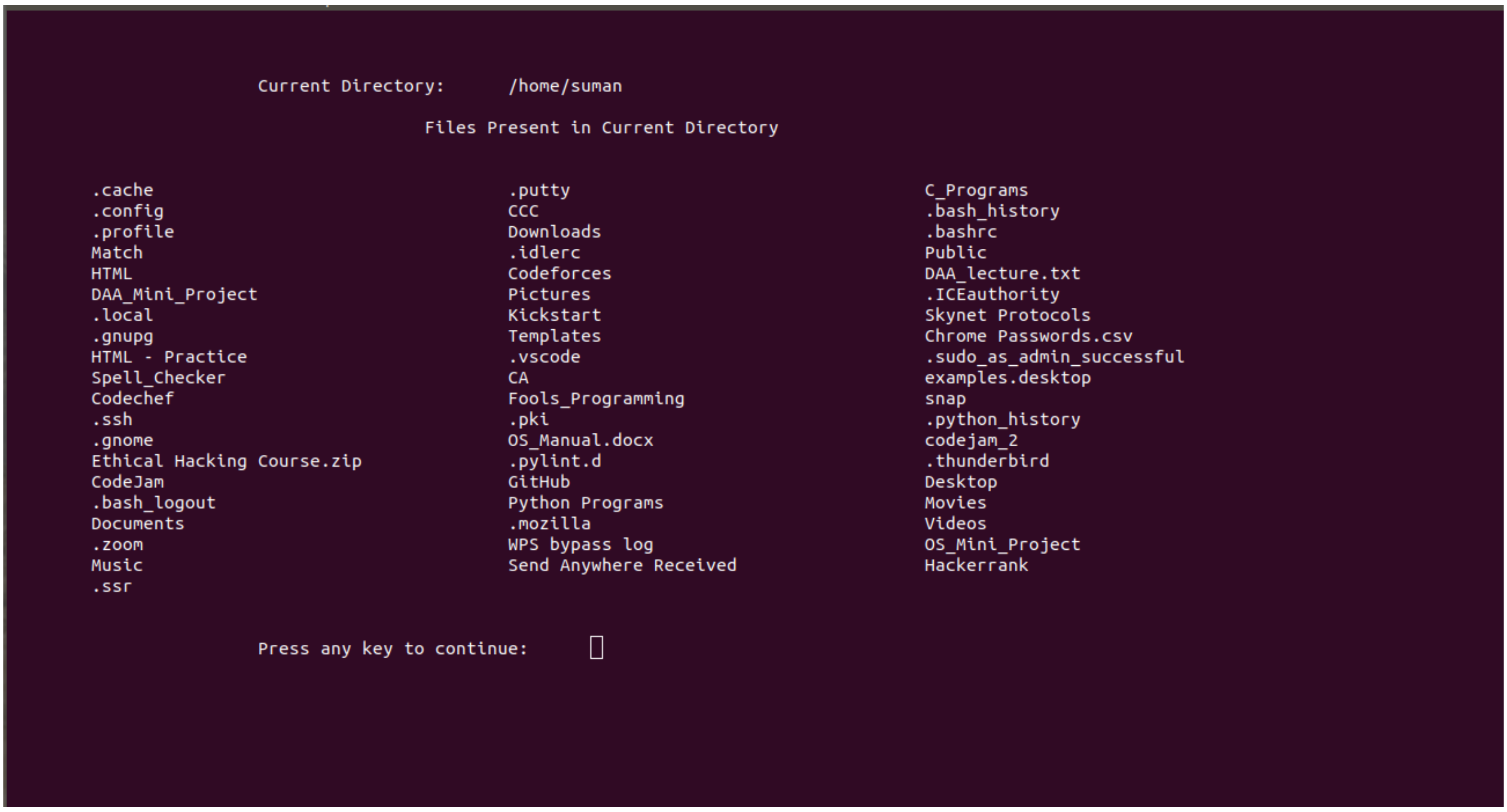


```
Current Directory: /home/suman/OS_Mini_Project/File_Manager

Parent Directory: /home/suman/OS_Mini_Project

Reached Parent Directory!

Current Directory: /home/suman/OS_Mini_Project

        Press any key to Continue...
```

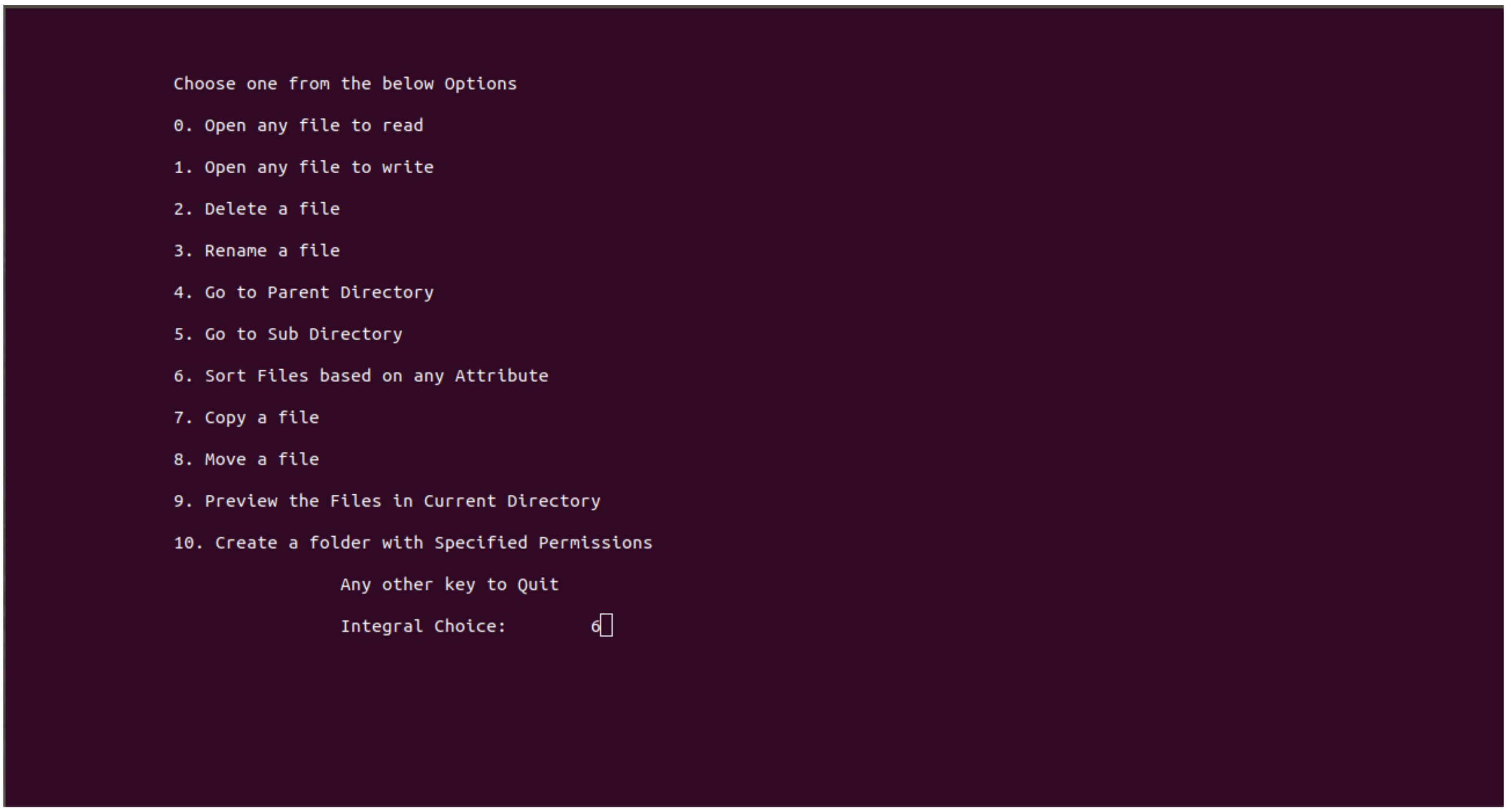


```
                Current Directory:      /home/suman/OS_Mini_Project

                                Files Present in Current Directory


    File_Manager                                File_Manager.zip


                Press any key to continue:
```

Screenshot: 1. Navigation successful. 2. files in Current Working Directory

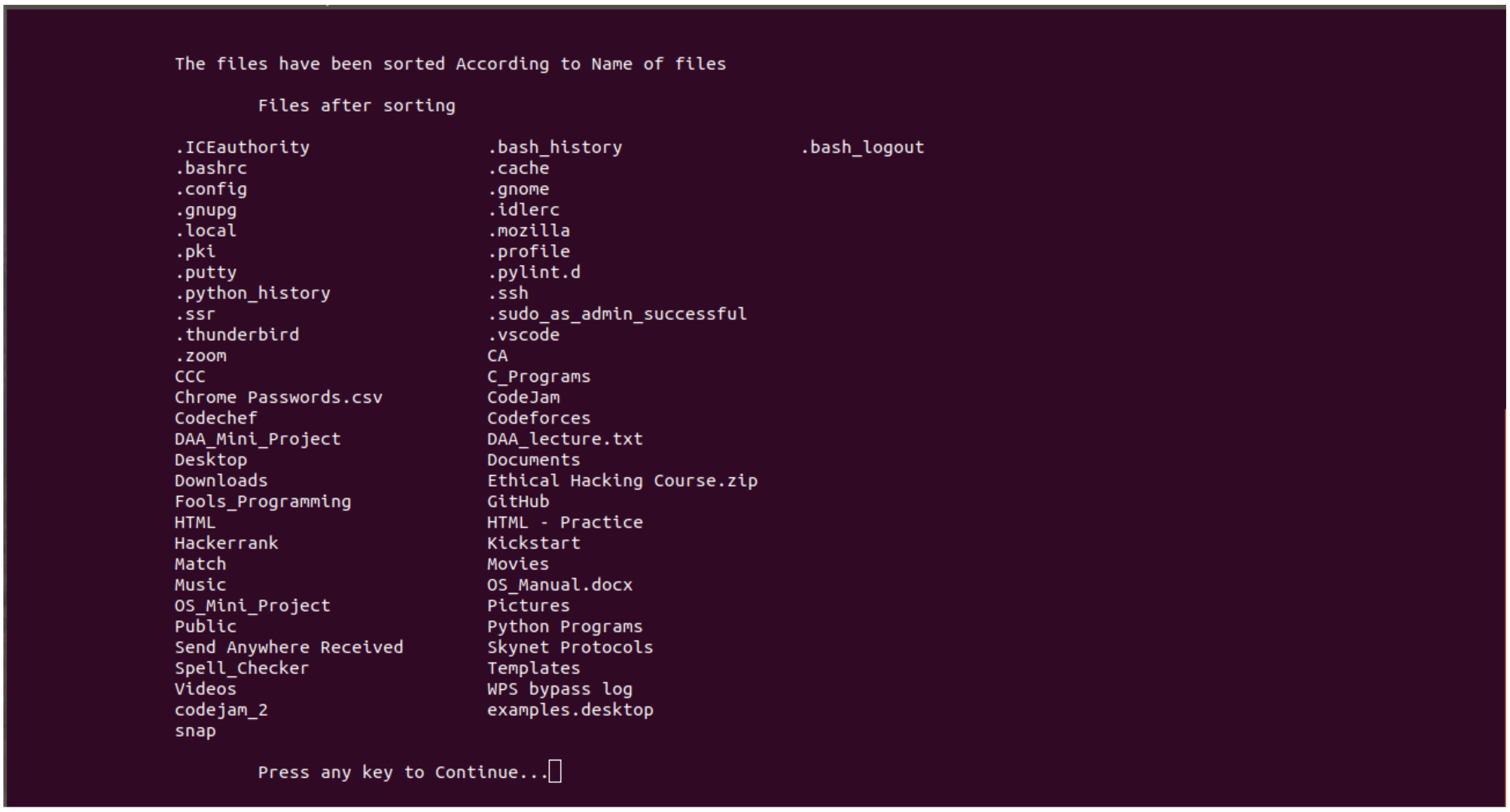


```
                Current Directory:      /home/suman

                                Files Present in Current Directory


    .cache                                      .putty                                      C_Programs
    .config                                     CCC                                         .bash_history
    .profile                                    Downloads                                   .bashrc
    Match                                       .idlerc                                     Public
    HTML                                        Codeforces                                  DAA_lecture.txt
    DAA_Mini_Project                            Pictures                                    .ICEauthority
    .local                                      Kickstart                                   Skynet Protocols
    .gnupg                                      Templates                                   Chrome Passwords.csv
    HTML - Practice                             .vscode                                     .sudo_as_admin_successful
    Spell_Checker                               CA                                          examples.desktop
    Codechef                                    Fools_Programming                           snap
    .ssh                                        .pki                                        .python_history
    .gnome                                      OS_Manual.docx                              codejam_2
    Ethical Hacking Course.zip                  .pylint.d                                   .thunderbird
    CodeJam                                     GitHub                                      Desktop
    .bash_logout                                Python Programs                             Movies
    Documents                                   .mozilla                                    Videos
    .zoom                                       WPS bypass log                              OS_Mini_Project
    Music                                       Send Anywhere Received                      Hackerrank
    .ssr


                Press any key to continue:
```

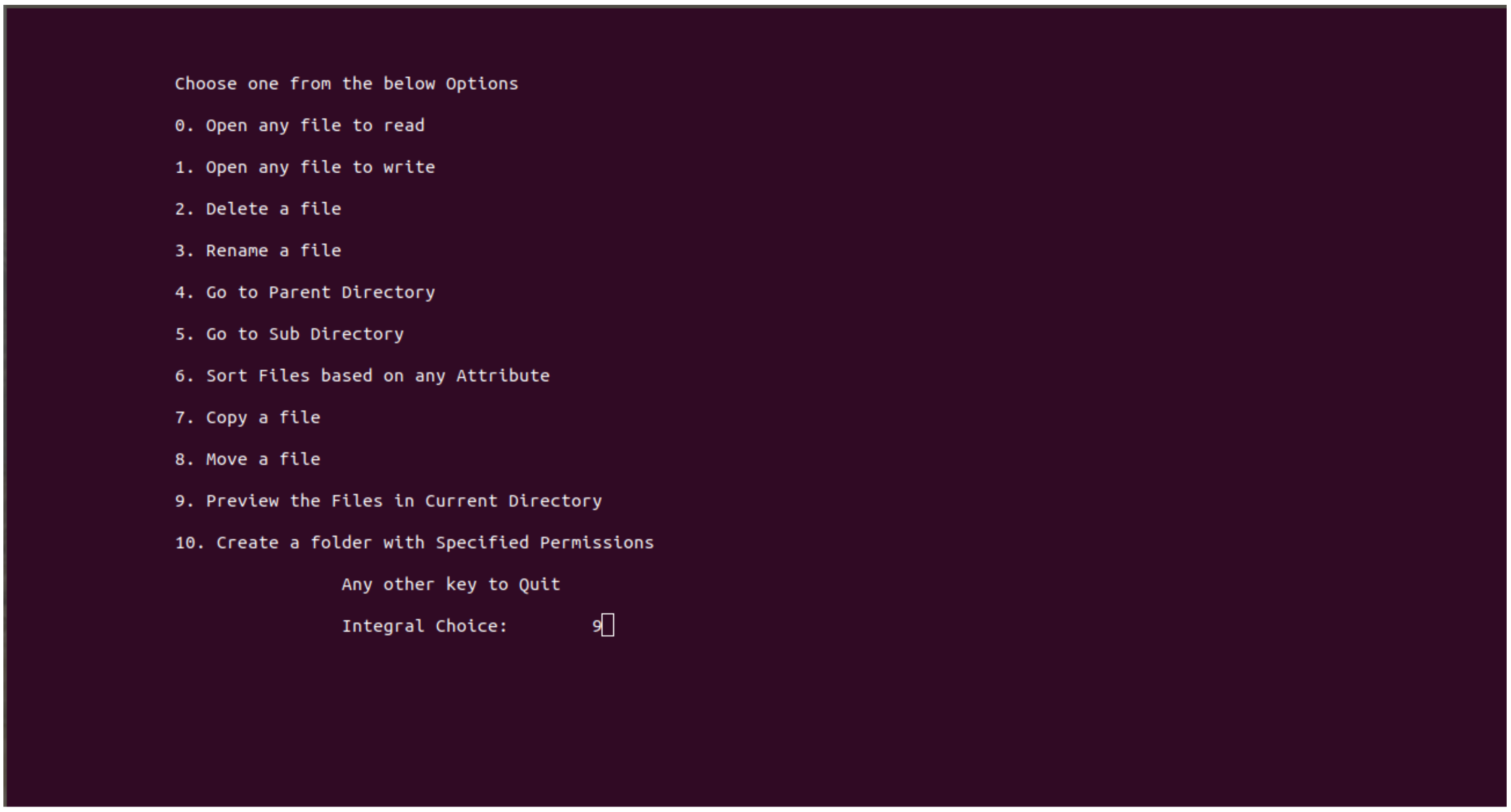


```
Choose one from the below Options

0. Open any file to read

1. Open any file to write

2. Delete a file

3. Rename a file

4. Go to Parent Directory

5. Go to Sub Directory

6. Sort Files based on any Attribute

7. Copy a file

8. Move a file

9. Preview the Files in Current Directory

10. Create a folder with Specified Permissions

                Any other key to Quit

                Integral Choice:        6
```

Screenshot: Files sorted based on Names in Lexicographic order

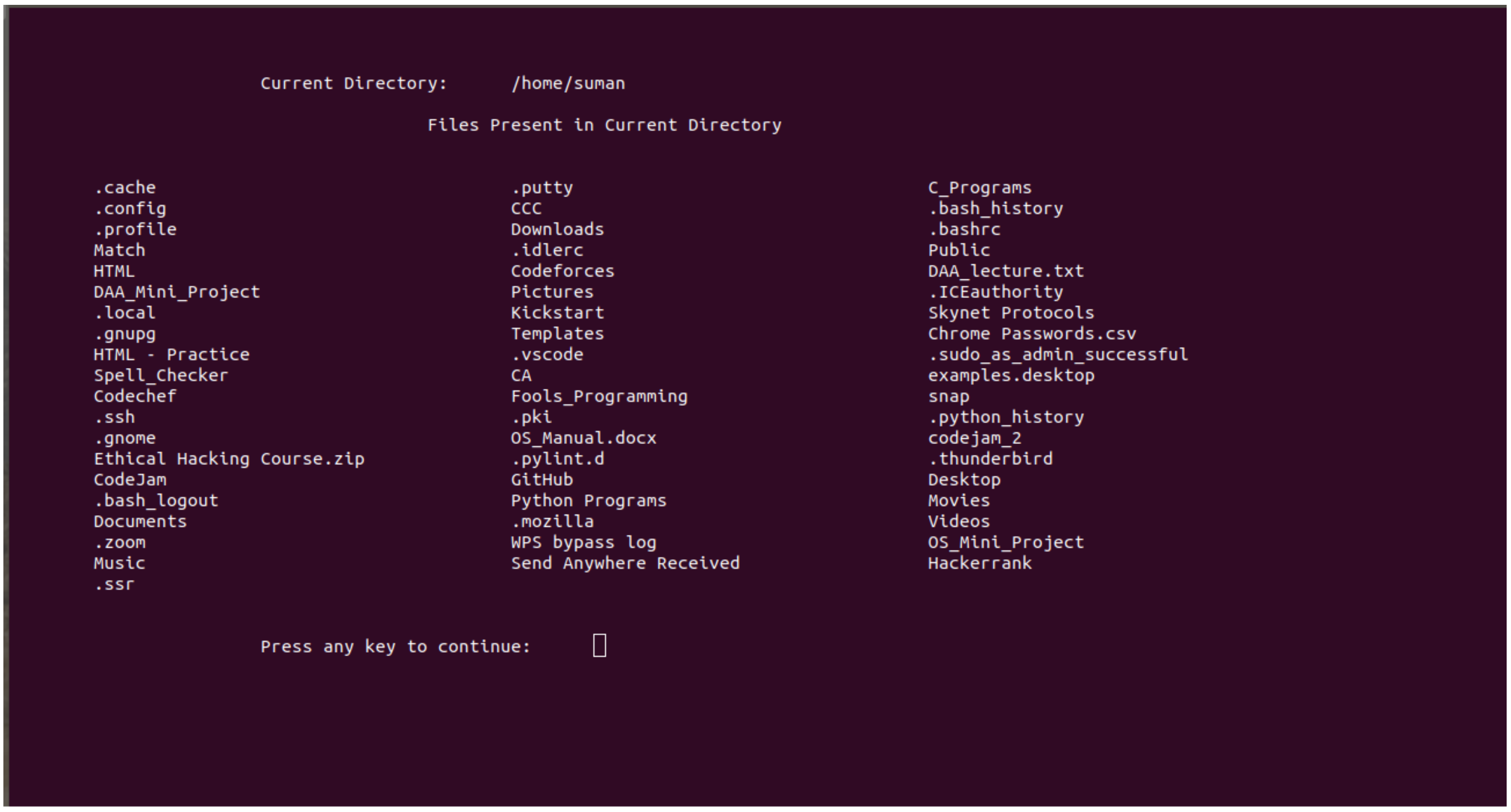


```
The files have been sorted According to Name of files

        Files after sorting

.ICEauthority                 .bash_history                 .bash_logout
.bashrc                       .cache
.config                       .gnome
.gnupg                        .idlerc
.local                        .mozilla
.pki                          .profile
.putty                        .pylint.d
.python_history               .ssh
.ssr                          .sudo_as_admin_successful
.thunderbird                  .vscode
.zoom                         CA
CCC                           C_Programs
Chrome Passwords.csv          CodeJam
Codechef                      Codeforces
DAA_Mini_Project              DAA_lecture.txt
Desktop                       Documents
Downloads                     Ethical Hacking Course.zip
Fools_Programming             GitHub
HTML                          HTML - Practice
Hackerrank                    Kickstart
Match                         Movies
Music                         OS_Manual.docx
OS_Mini_Project               Pictures
Public                        Python Programs
Send Anywhere Received        Skynet Protocols
Spell_Checker                 Templates
Videos                        WPS bypass log
codejam_2                     examples.desktop
snap

        Press any key to Continue...
```

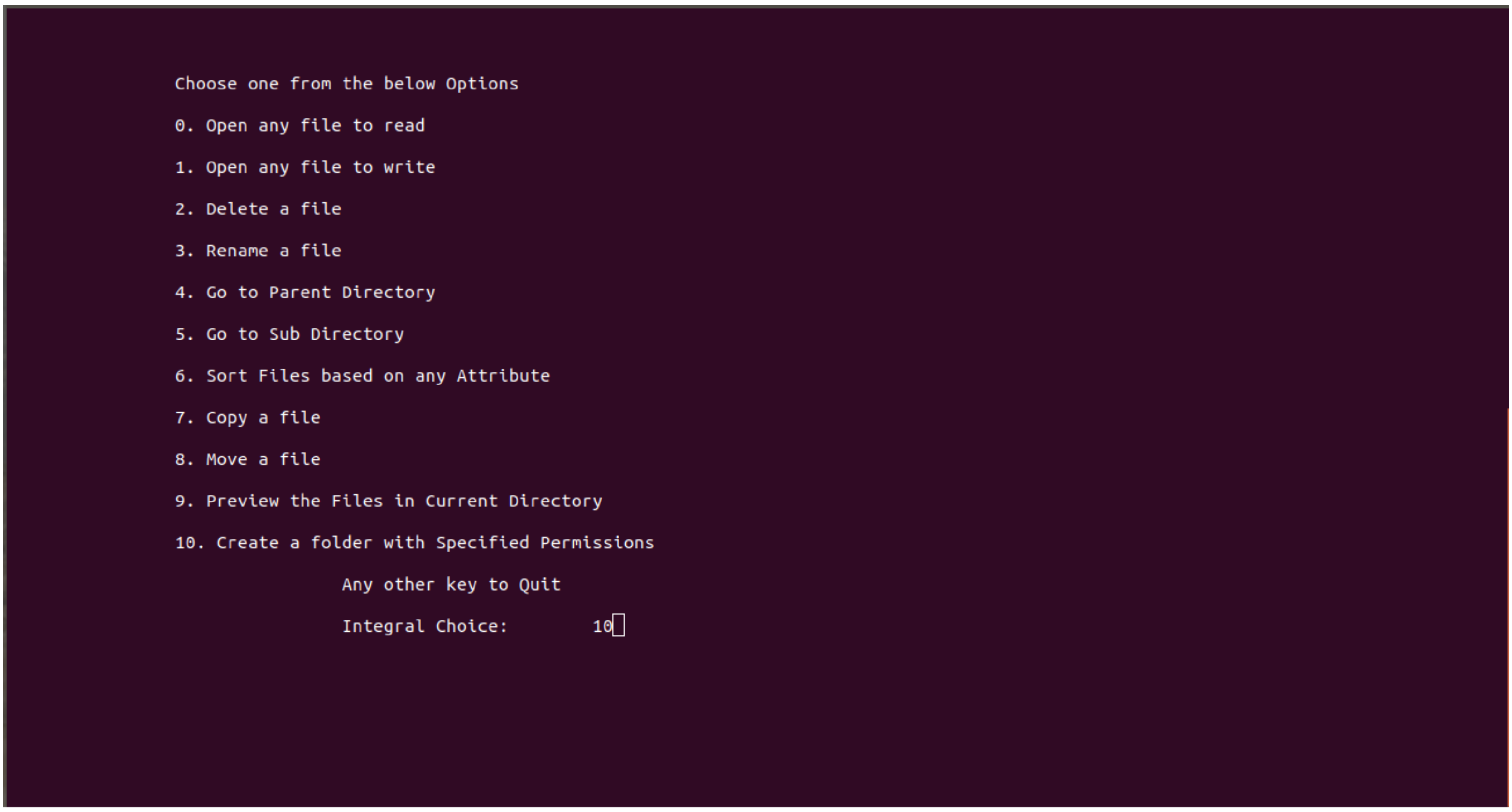


```
        Choose one from the below Options

        0. Open any file to read

        1. Open any file to write

        2. Delete a file

        3. Rename a file

        4. Go to Parent Directory

        5. Go to Sub Directory

        6. Sort Files based on any Attribute

        7. Copy a file

        8. Move a file

        9. Preview the Files in Current Directory

        10. Create a folder with Specified Permissions

                        Any other key to Quit

                        Integral Choice:        9
```

Screenshot: Preview of files in Current Working Directory

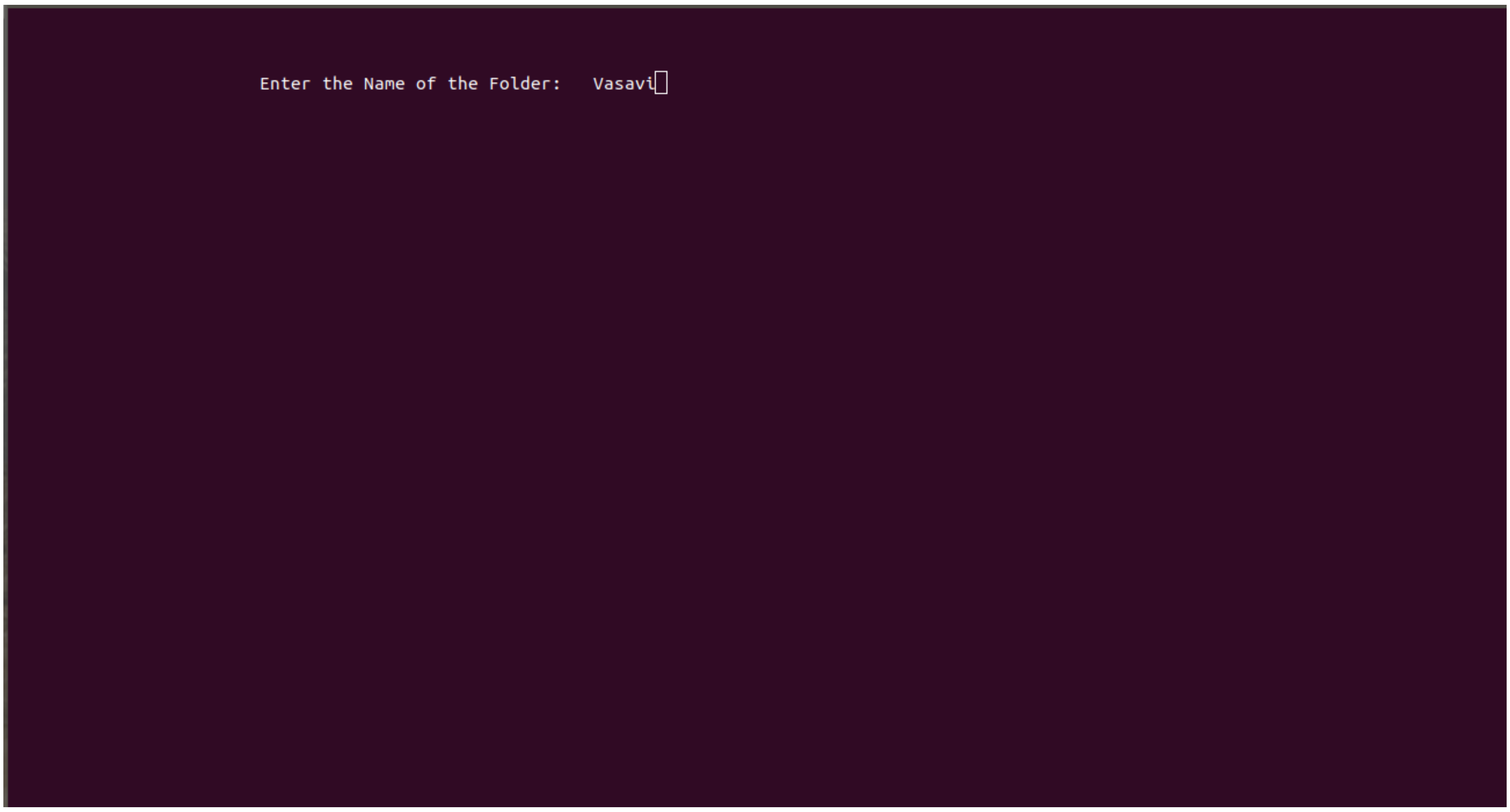


```
                Current Directory:      /home/suman

                                Files Present in Current Directory


    .cache                              .putty                              C_Programs
    .config                             CCC                                 .bash_history
    .profile                            Downloads                           .bashrc
    Match                               .idlerc                             Public
    HTML                                Codeforces                          DAA_lecture.txt
    DAA_Mini_Project                    Pictures                            .ICEauthority
    .local                              Kickstart                           Skynet Protocols
    .gnupg                              Templates                           Chrome Passwords.csv
    HTML - Practice                     .vscode                             .sudo_as_admin_successful
    Spell_Checker                       CA                                  examples.desktop
    Codechef                            Fools_Programming                   snap
    .ssh                                .pki                                .python_history
    .gnome                              OS_Manual.docx                      codejam_2
    Ethical Hacking Course.zip          .pylint.d                           .thunderbird
    CodeJam                             GitHub                              Desktop
    .bash_logout                        Python Programs                     Movies
    Documents                           .mozilla                            Videos
    .zoom                               WPS bypass log                      OS_Mini_Project
    Music                               Send Anywhere Received              Hackerrank
    .ssr


                Press any key to continue:
```

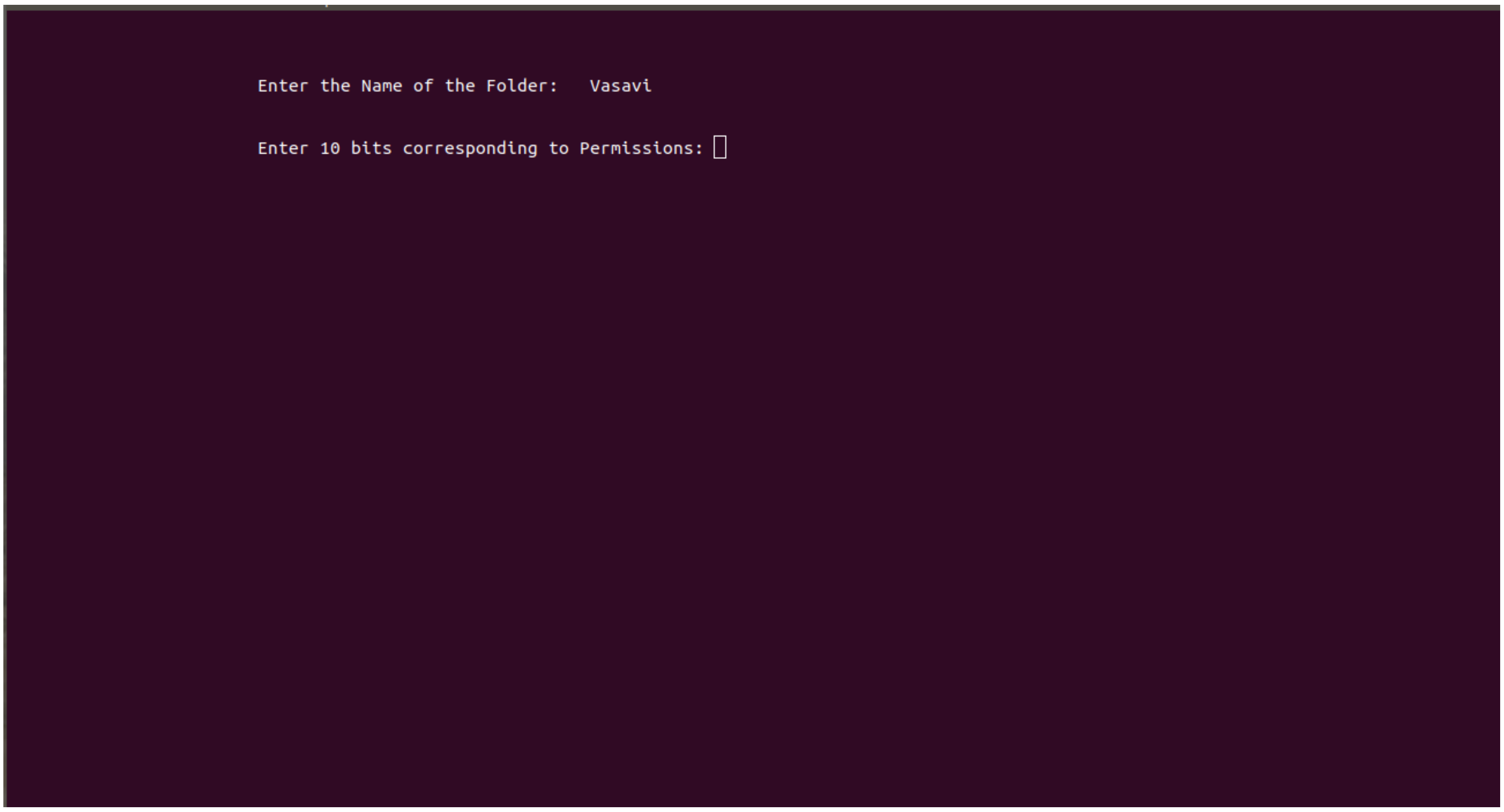


```
Choose one from the below Options

0. Open any file to read

1. Open any file to write

2. Delete a file

3. Rename a file

4. Go to Parent Directory

5. Go to Sub Directory

6. Sort Files based on any Attribute

7. Copy a file

8. Move a file

9. Preview the Files in Current Directory

10. Create a folder with Specified Permissions

                Any other key to Quit

                Integral Choice:        10
```

Screenshot: Creation of New folder

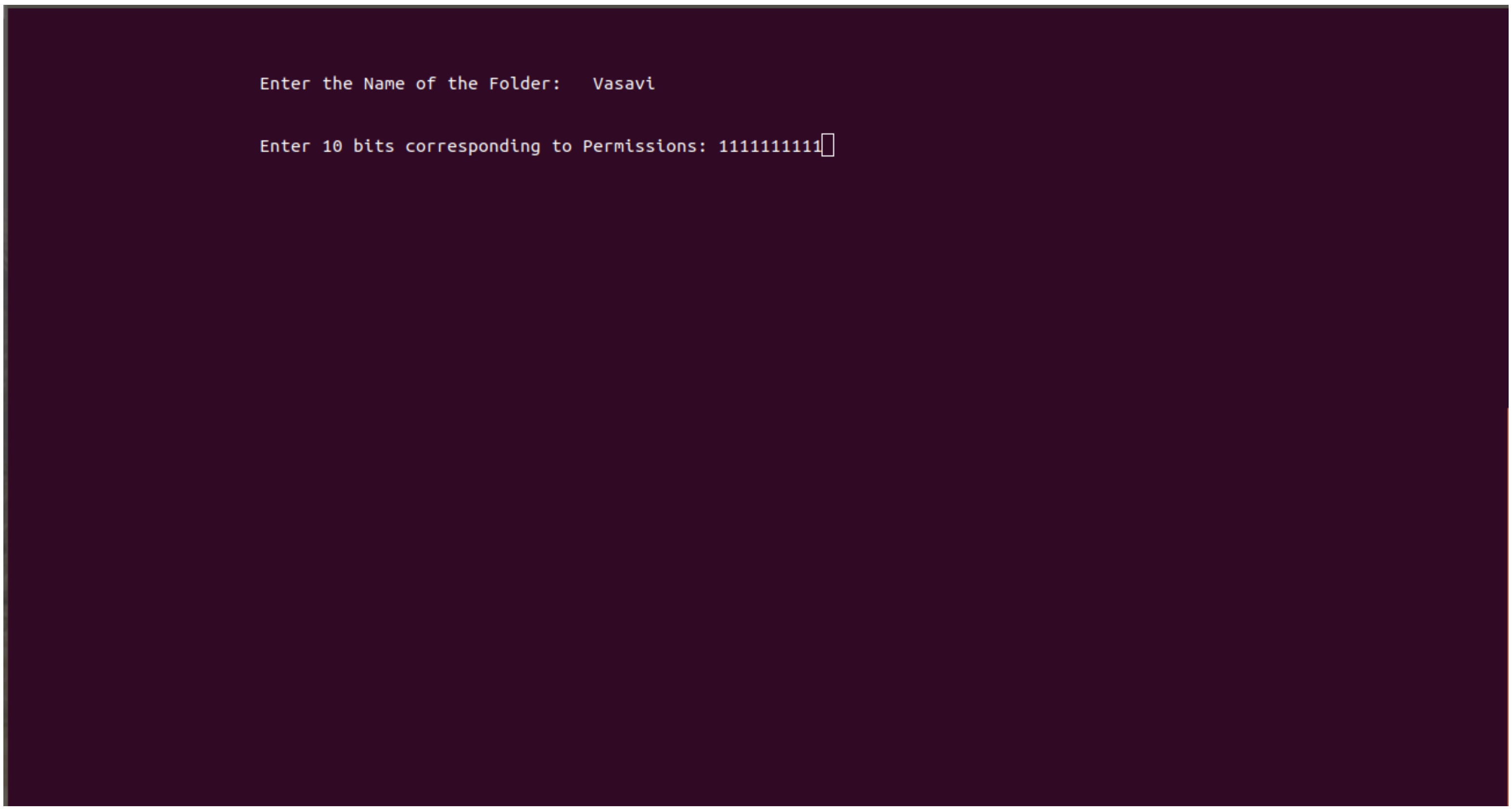


```
        Enter the Name of the Folder:   Vasavi
```

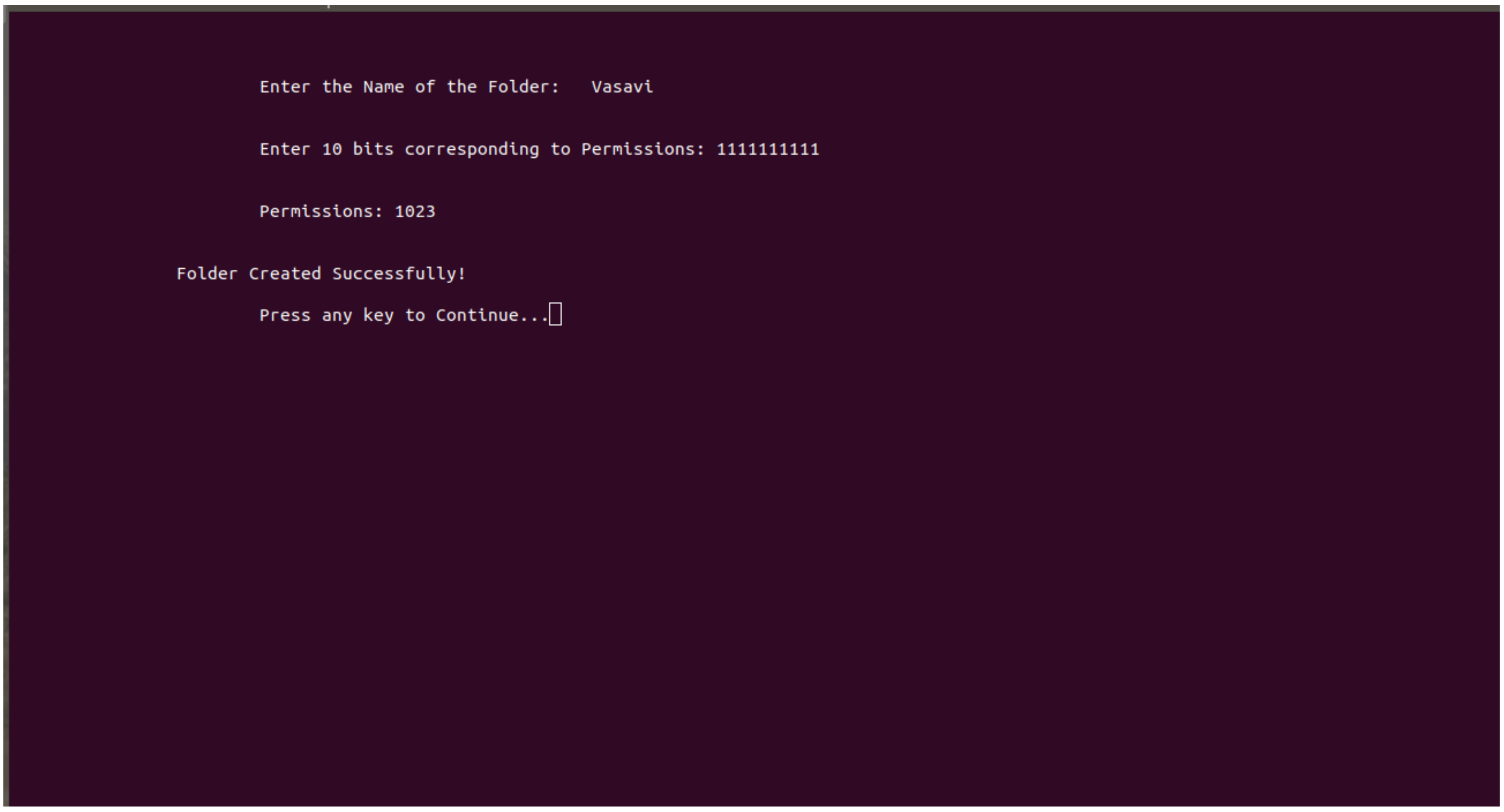


```
Enter the Name of the Folder:   Vasavi

Enter 10 bits corresponding to Permissions:
```

Screenshot: Specifying permissions for newly created Folder

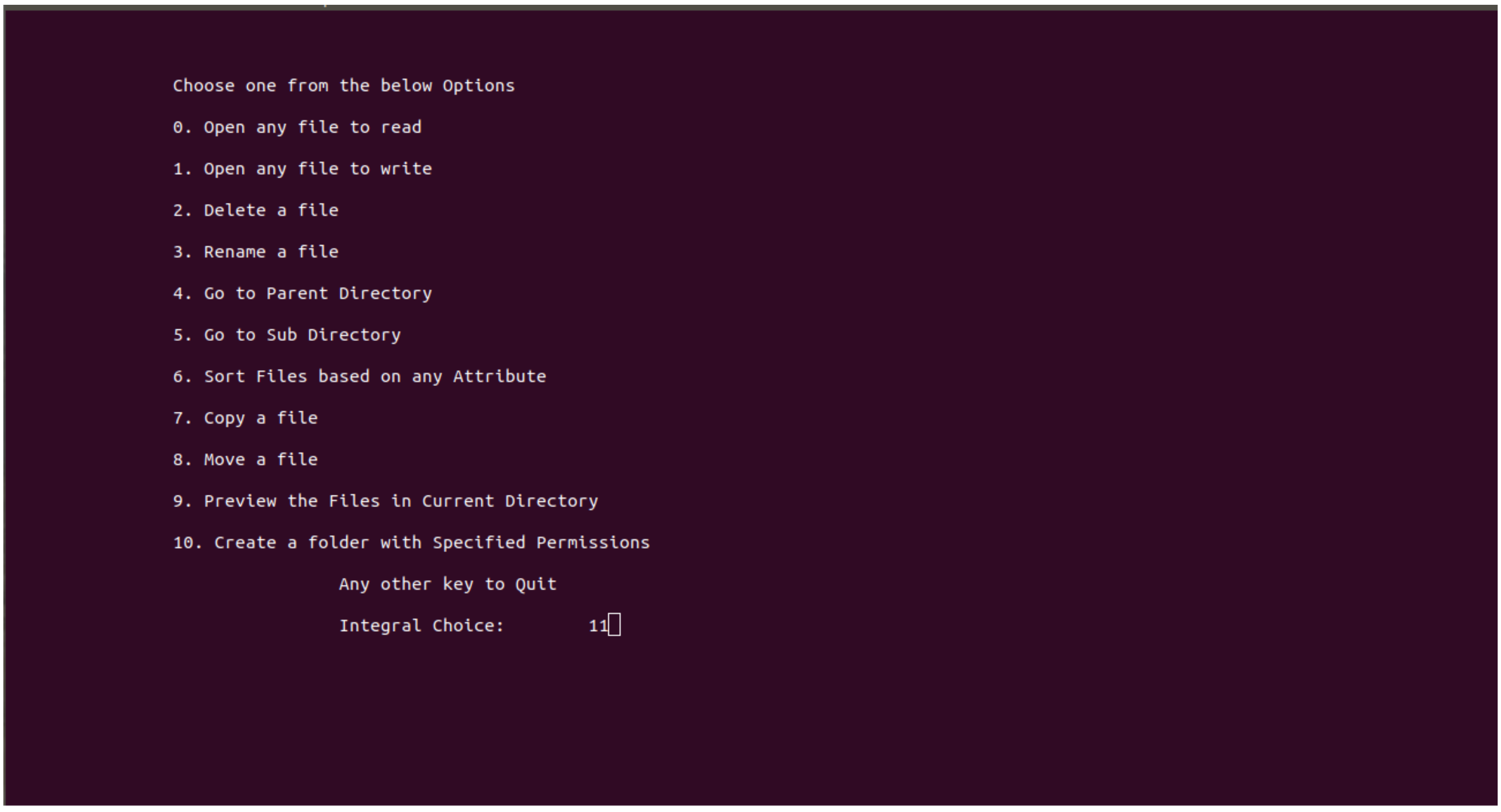


```
Enter the Name of the Folder:   Vasavi

Enter 10 bits corresponding to Permissions: 1111111111
```

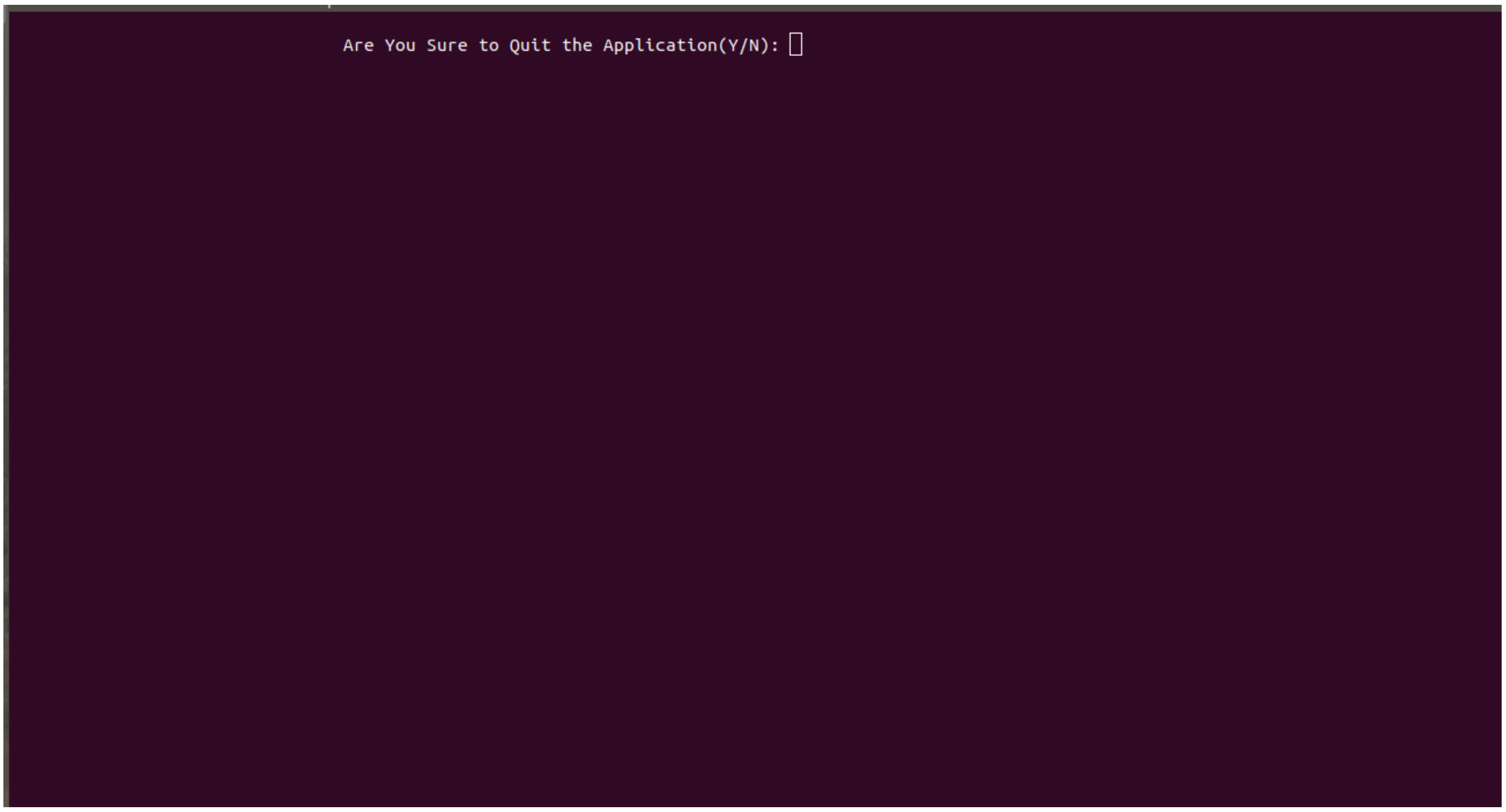


```
        Enter the Name of the Folder:   Vasavi

        Enter 10 bits corresponding to Permissions: 1111111111

        Permissions: 1023

Folder Created Successfully!
        Press any key to Continue...
```

Screenshot: Folder creation successful with specified permissions

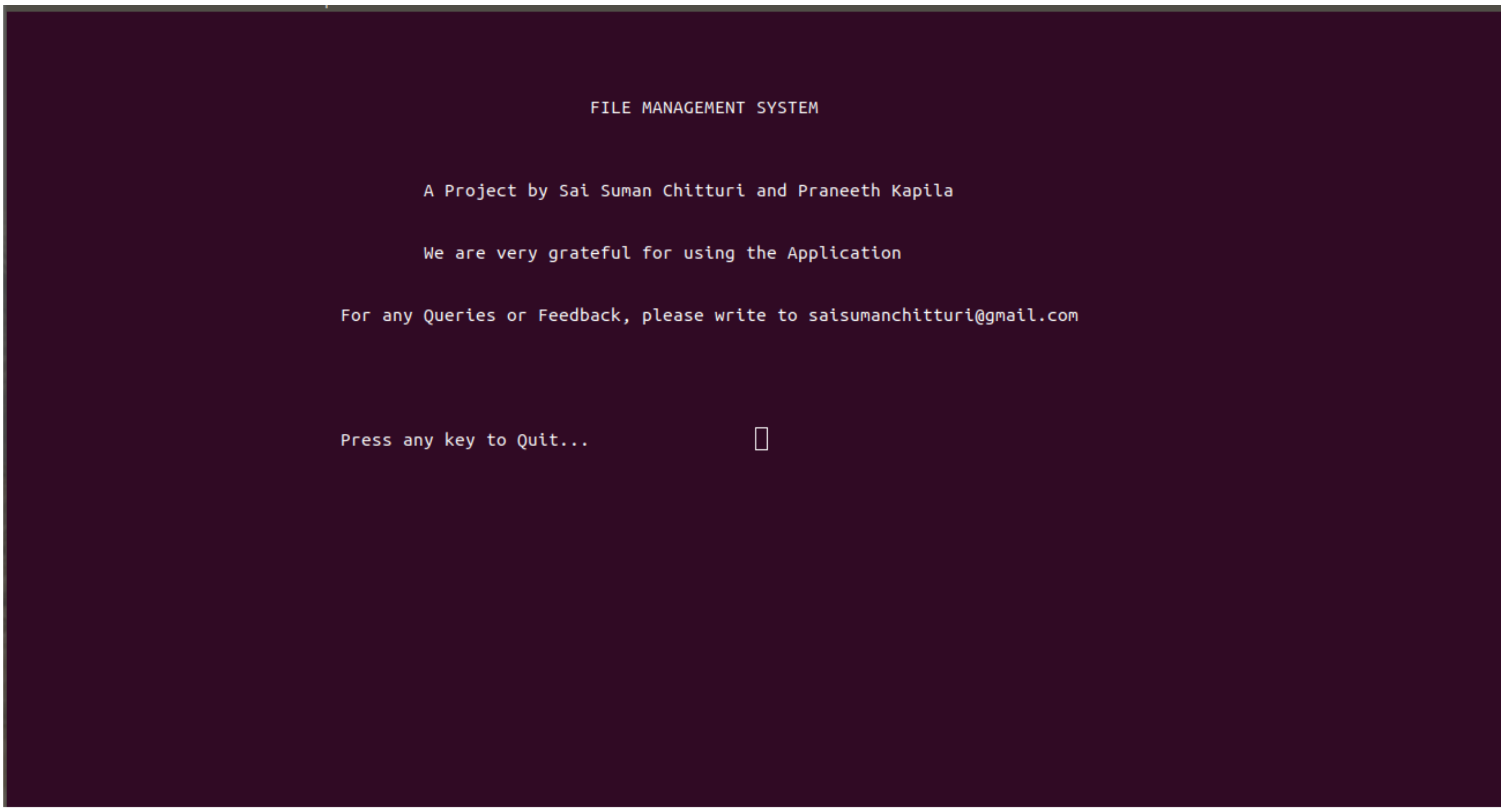


```
Choose one from the below Options
0. Open any file to read
1. Open any file to write
2. Delete a file
3. Rename a file
4. Go to Parent Directory
5. Go to Sub Directory
6. Sort Files based on any Attribute
7. Copy a file
8. Move a file
9. Preview the Files in Current Directory
10. Create a folder with Specified Permissions
                Any other key to Quit
                Integral Choice:        11
```



```
Are You Sure to Quit the Application(Y/N): 
```

Screenshot: Quit screen and Conclusion



```
                        FILE MANAGEMENT SYSTEM


        A Project by Sai Suman Chitturi and Praneeth Kapila

        We are very grateful for using the Application

For any Queries or Feedback, please write to saisumanchitturi@gmail.com



Press any key to Quit...
```

References:

Internet Documents

Professional Internet Site

[1]. GeeksforGeeks | A computer science portal for geeks.

Available: https://www.geeksforgeeks.org/

Professional Internet Site

[2]. Stack Overflow – Where Developers Learn, Share, & Build Careers

Available: https://stackoverflow.com/questions/

Professional Internet Site

[3]. IBM Knowledge Center – Home of IBM product documentation

Available: https://www.ibm.com/support/knowledgecenter/

General Discussion forum

[4]. Tutorialspoint

Available: https://www.tutorialspoint.com/

Electronic Documents

Official C99 standard documentation

[5]. Available: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf

Official VS Code Documentation for configuring GCC on Linux

[6]. Available: https://code.visualstudio.com/docs/cpp/config-linux