Project Report

Basic FILE-EXPLORER for UBUNTU OS

Done by:

1. Azeem Mohamed Husain
   - B22F0759CS142

---

1. Introduction:

- The Basic File Explorer for Ubuntu OS is a command-line tool developed in C language. It allows users to perform various file operations such as creating, listing, copying, moving, deleting, renaming files, searching for files, changing directory, and managing file permissions. This project aims to provide a simple and intuitive interface for basic file management tasks on the Ubuntu operating system.

2. Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>

#define MAX_PATH 1024
#define HOME_PATH "/home/azeemhusain"

void list_files(const char *path);
void copy_file(const char *src, const char *dest);
void move_file(const char *src, const char *dest);
void delete_file(const char *path);
void rename_file(const char *old_name, const char *new_name);
void search_file(const char *path, const char *filename);
void change_permissions(const char *path, mode_t mode);
void create_file(const char *path);
void check_file_permission(const char *path);

void list_files(const char *path) {
struct dirent *entry;
DIR *dp = opendir(path);
```

```c
if (dp == NULL) {
perror("opendir");
return;
}

printf("\nListing files in directory: %s\n", path);
while ((entry = readdir(dp))) {
printf("%s\n", entry->d_name);
}

closedir(dp);
}

void copy_file(const char *src, const char *dest) {
FILE *source, *target;
source = fopen(src, "rb");
if (source == NULL) {
perror("fopen src");
return;
}

struct stat st;
if (stat(dest, &st) == 0 && S_ISDIR(st.st_mode)) {
// If the destination is a directory, ask for the file name and create it in
the destination directory
char filename[MAX_PATH];
printf("Enter the file name with extension to create in the destination: ");
scanf("%s", filename);
char new_dest[MAX_PATH];
if (snprintf(new_dest, sizeof(new_dest), "%s/%s", dest, filename) >=
sizeof(new_dest)) {
fprintf(stderr, "Error: destination path is too long\n");
fclose(source);
return;
}
target = fopen(new_dest, "wb");
} else {
target = fopen(dest, "wb");
}

if (target == NULL) {
fclose(source);
perror("fopen dest");
return;
}

char buffer[BUFSIZ];
```

```
size_t n;

while ((n = fread(buffer, 1, sizeof(buffer), source)) > 0) {
fwrite(buffer, 1, n, target);
}

fclose(source);
fclose(target);
}

void move_file(const char *src, const char *dest) {
struct stat st;
if (stat(dest, &st) == 0 && S_ISDIR(st.st_mode)) {
// If the destination is a directory, append the file name from src to dest
const char *filename = strrchr(src, '/');
if (filename) {
filename++; // Skip the '/'
} else {
filename = src;
}

char new_dest[MAX_PATH];
if (snprintf(new_dest, sizeof(new_dest), "%s/%s", dest, filename) >=
sizeof(new_dest)) {
fprintf(stderr, "Error: destination path is too long\n");
return;
}

if (rename(src, new_dest) != 0) {
perror("rename");
}
} else {
if (rename(src, dest) != 0) {
perror("rename");
}
}
}

void delete_file(const char *path) {
char confirm;
printf("Are you sure you want to delete %s? (y/n): ", path);
scanf(" %c", &confirm);
if (confirm == 'y' || confirm == 'Y') {
if (remove(path) != 0) {
perror("remove");
} else {
printf("File deleted successfully.\n");
}
```

```c
} else {
printf("File deletion cancelled.\n");
}
}

void rename_file(const char *old_name, const char *new_name) {
if (rename(old_name, new_name) != 0) {
perror("rename");
}
}

void search_file(const char *path, const char *filename) {
struct dirent *entry;
DIR *dp = opendir(path);
if (dp == NULL) {
perror("opendir");
return;
}

printf("\nSearching for file: %s in directory: %s\n", filename, path);
int found = 0;
while ((entry = readdir(dp))) {
if (strcmp(entry->d_name, filename) == 0) {
printf("File found: %s/%s\n", path, entry->d_name);
found = 1;
break;
}
}

if (!found) {
printf("File not found.\n");
}

closedir(dp);
}

void change_permissions(const char *path, mode_t mode) {
if (chmod(path, mode) != 0) {
perror("chmod");
} else {
printf("Permissions changed successfully.\n");
}
}

void create_file(const char *path) {
char filename[MAX_PATH];
printf("Enter the file name with extension: ");
scanf("%s", filename);
```

```
// Concatenate the path and filename
char full_path[MAX_PATH];
strcpy(full_path, path);
strcat(full_path, "/");
strcat(full_path, filename);

// Open the file in write mode
FILE *file = fopen(full_path, "w");
if (file == NULL) {
perror("fopen");
return;
}

fclose(file);
printf("File created successfully: %s\n", full_path);
}


void check_file_permission(const char *path) {
struct stat file_stat;
if (stat(path, &file_stat) != 0) {
perror("stat");
return;
}

printf("File permissions for %s:\n", path);
printf("Owner: %s%s%s\n", (file_stat.st_mode & S_IRUSR) ? "read " : "no-read
",
(file_stat.st_mode & S_IWUSR) ? "write " : "no-write ",
(file_stat.st_mode & S_IXUSR) ? "execute" : "no-execute");
printf("Group: %s%s%s\n", (file_stat.st_mode & S_IRGRP) ? "read " : "no-read
",
(file_stat.st_mode & S_IWGRP) ? "write " : "no-write ",
(file_stat.st_mode & S_IXGRP) ? "execute" : "no-execute");
printf("Others: %s%s%s\n", (file_stat.st_mode & S_IROTH) ? "read " : "no-read
",
(file_stat.st_mode & S_IWOTH) ? "write " : "no-write ",
(file_stat.st_mode & S_IXOTH) ? "execute" : "no-execute");
}


int main() {
char cwd[MAX_PATH];
strcpy(cwd, HOME_PATH);
chdir(cwd);

int choice;
char path[MAX_PATH];
char src[MAX_PATH], dest[MAX_PATH], new_name[MAX_PATH], filename[MAX_PATH];
mode_t mode;

while (1) {
printf("\nCurrent Directory: %s\n", cwd);
printf("1. Create File\n");
printf("2. List Files\n");
printf("3. Copy File\n");
printf("4. Move File\n");
printf("5. Delete File\n");
printf("6. Rename File\n");
printf("7. Search File\n");
printf("8. Change Directory\n");
printf("9. Check File Permission\n");
printf("10. Change File Permissions\n");
printf("11. Exit\n");
printf("Enter your choice: ");

if (scanf("%d", &choice) != 1) {
printf("Invalid choice! Please enter a number.\n");
while (getchar() != '\n'); // Clear the input buffer
continue;
}
switch (choice) {
case 1:
create_file(cwd);
break;
case 2:
list_files(cwd);
break;
case 3:
printf("Enter source file path: ");
scanf(" %s", src); // Note the space before %s to consume newline character
printf("Enter destination file path: ");
scanf(" %s", dest); // Note the space before %s to consume newline character
copy_file(src, dest);
break;
case 4:
printf("Enter source file path: ");
scanf(" %s", src); // Note the space before %s to consume newline character
printf("Enter destination file path: ");
scanf(" %s", dest); // Note the space before %s to consume newline character
move_file(src, dest);
break;
case 5:
printf("Enter file path to delete: ");
scanf(" %s", path); // Note the space before %s to consume newline character
delete_file(path);
break;
case 6:
printf("Enter old file name: ");
scanf(" %s", src); // Note the space before %s to consume newline character
printf("Enter new file name: ");
scanf(" %s", new_name); // Note the space before %s to consume newline
character
rename_file(src, new_name);
break;
case 7:
printf("Enter filename to search: ");
scanf(" %s", filename); // Note the space before %s to consume newline
character
search_file(cwd, filename);
break;
case 8:
printf("Enter directory path: ");
scanf(" %s", path); // Note the space before %s to consume newline character
if (chdir(path) == 0){
getcwd(cwd, sizeof(cwd));
} else {
perror("chdir");
}
break;
case 9:
printf("Enter file path: ");
scanf(" %s", path); // Note the space before %s to consume newline character
check_file_permission(path);
break;
case 10:
printf("Enter file path: ");
scanf(" %s", path); // Note the space before %s to consume newline character
printf("Enter permission mode (in octal): ");
scanf("%o", &mode);
change_permissions(path, mode);
break;
case 11:
exit(0);
default:
printf("Invalid choice!\n");
}
}
return 0;
}
```

3. Features:

- Create File: Users can create a new file within the current directory by specifying the file name.
- List Files: This feature lists all the files and directories within the current directory.
- Copy File: Users can copy a file from a source path to a destination path.
- Move File: This feature enables users to move a file from one location to another.
- Delete File: Users can delete a file from the file system. A confirmation prompt is provided before deletion.
- Rename File: Allows users to rename a file.
- Search File: Users can search for a file within the current directory.
- Change Directory: Enables users to navigate through directories.
- Check File Permission: This feature displays the permissions of a specified file.
- Change File Permissions: Users can modify the permissions of a file by specifying the permission mode in octal format.

4. Implementation:

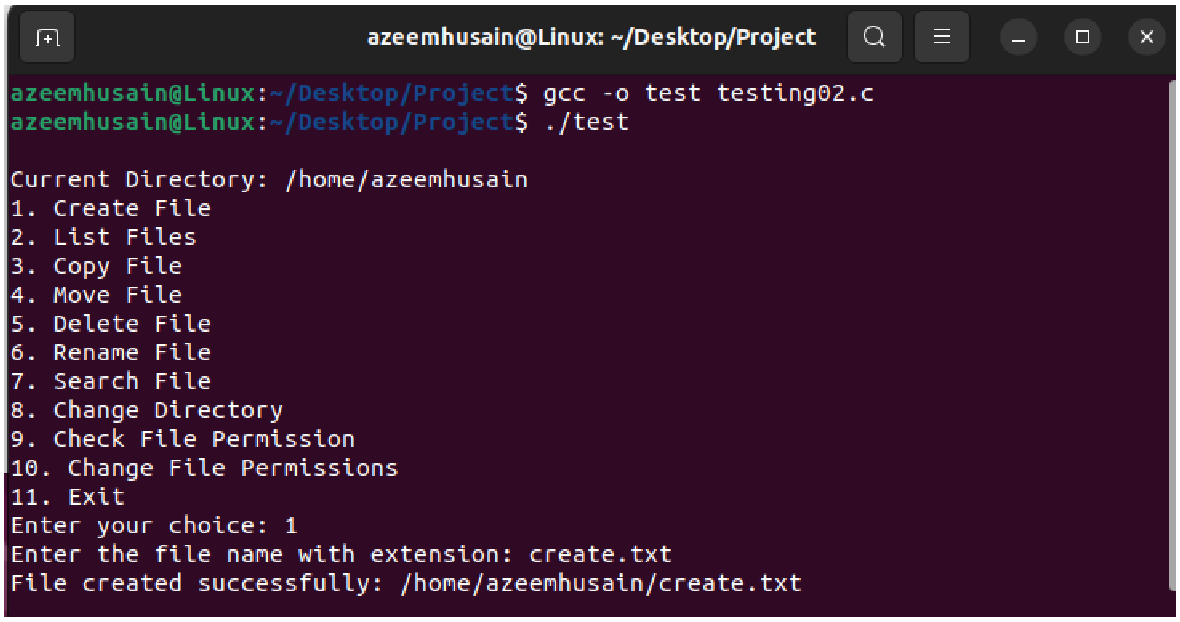
- The project is implemented in C language and utilizes various standard libraries such as stdio.h, stdlib.h, dirent.h, unistd.h, and string.h for file and directory manipulation, input/output operations, and string handling.

- The main functionalities are implemented as individual functions such as create_file, list_files, copy_file, move_file, delete_file, rename_file, search_file, change_directory, check_file_permission, and change_permissions.

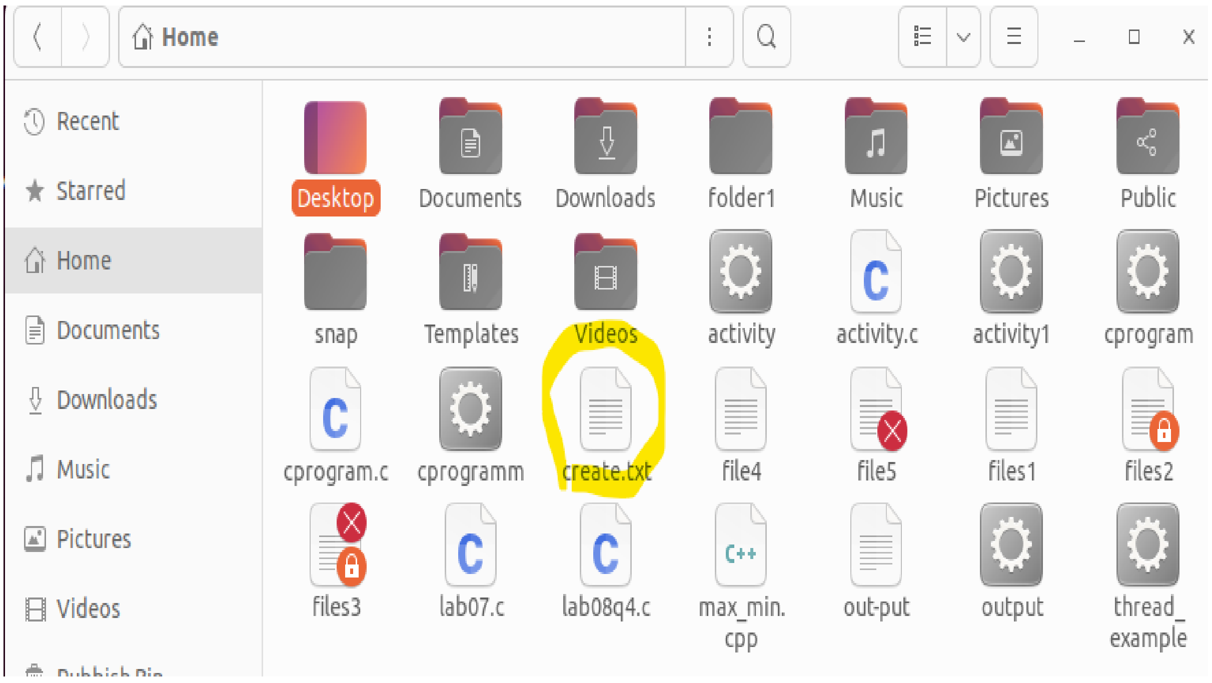
- A command-line interface is provided to interact with the user. The program continuously prompts the user for input, processes the input, and performs the corresponding file operation based on the user's choice.

5. Testing

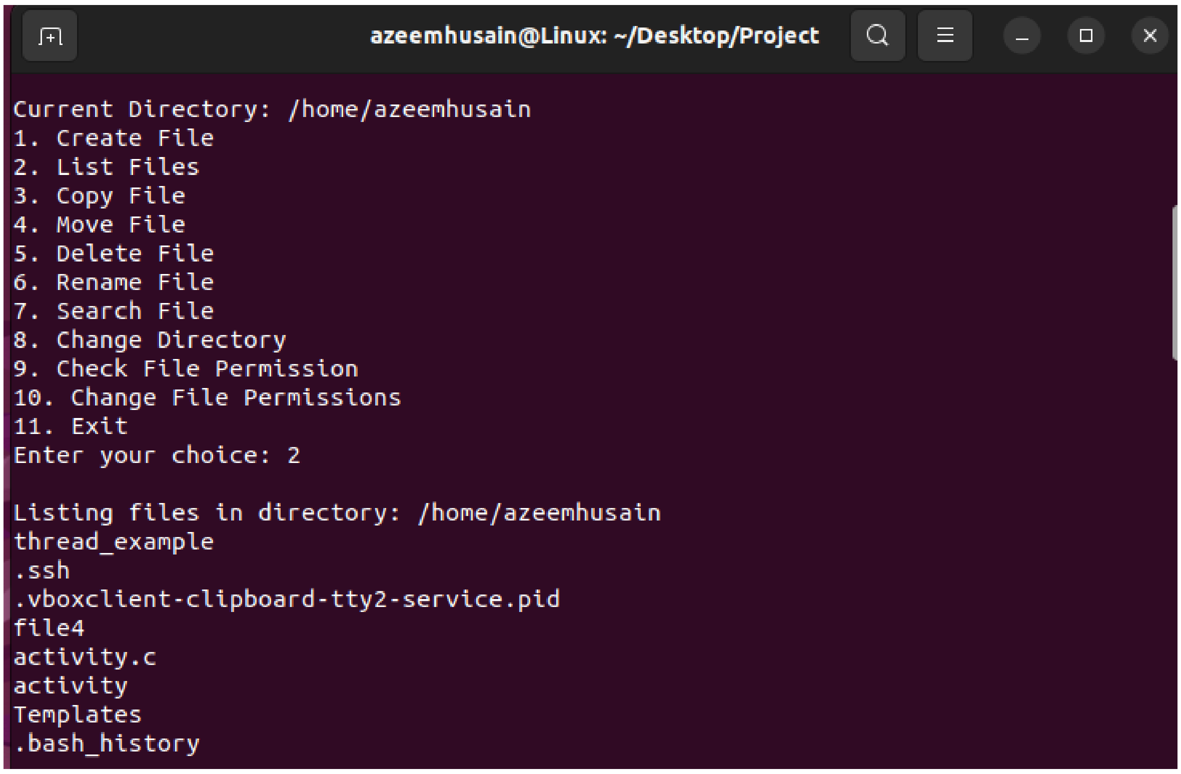
Create File



```
azeemhusain@Linux:~/Desktop/Project$ gcc -o test testing02.c
azeemhusain@Linux:~/Desktop/Project$ ./test

Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 1
Enter the file name with extension: create.txt
File created successfully: /home/azeemhusain/create.txt
```

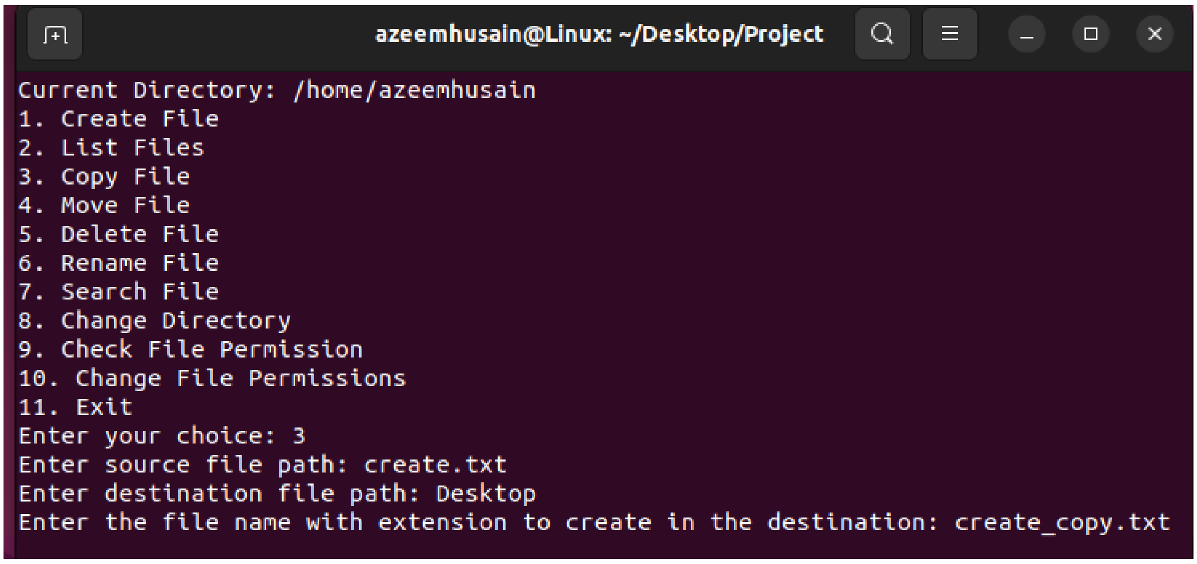


List File

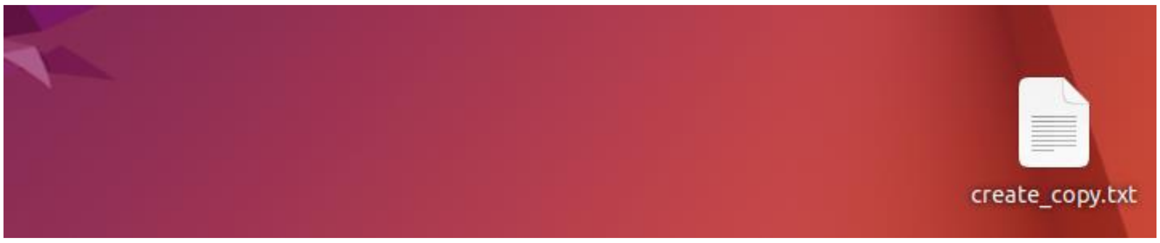


```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 2

Listing files in directory: /home/azeemhusain
thread_example
.ssh
.vboxclient-clipboard-tty2-service.pid
file4
activity.c
activity
Templates
.bash_history
```

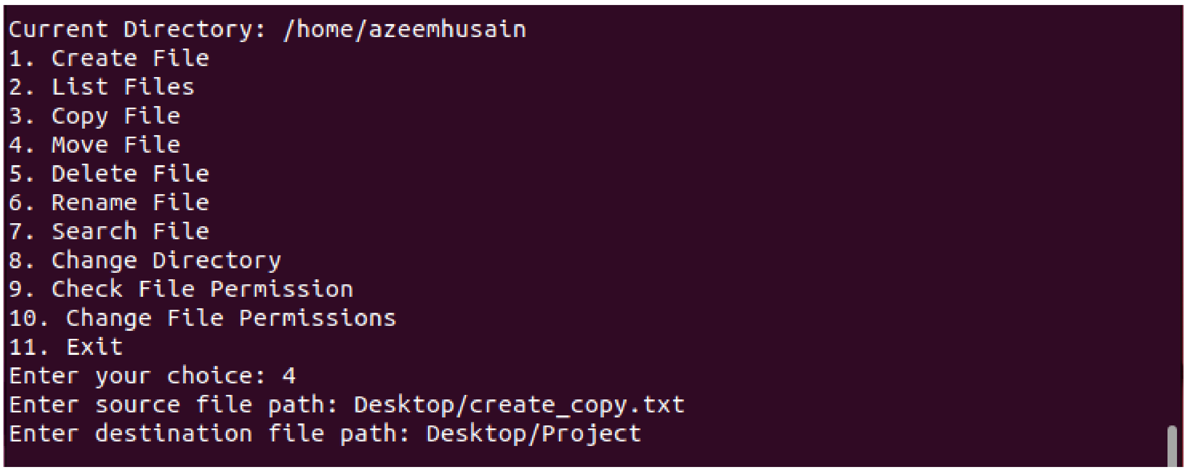
Copy File



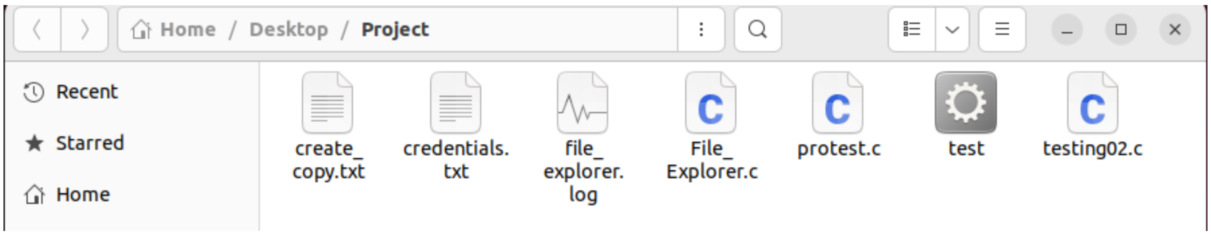
```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 3
Enter source file path: create.txt
Enter destination file path: Desktop
Enter the file name with extension to create in the destination: create_copy.txt
```



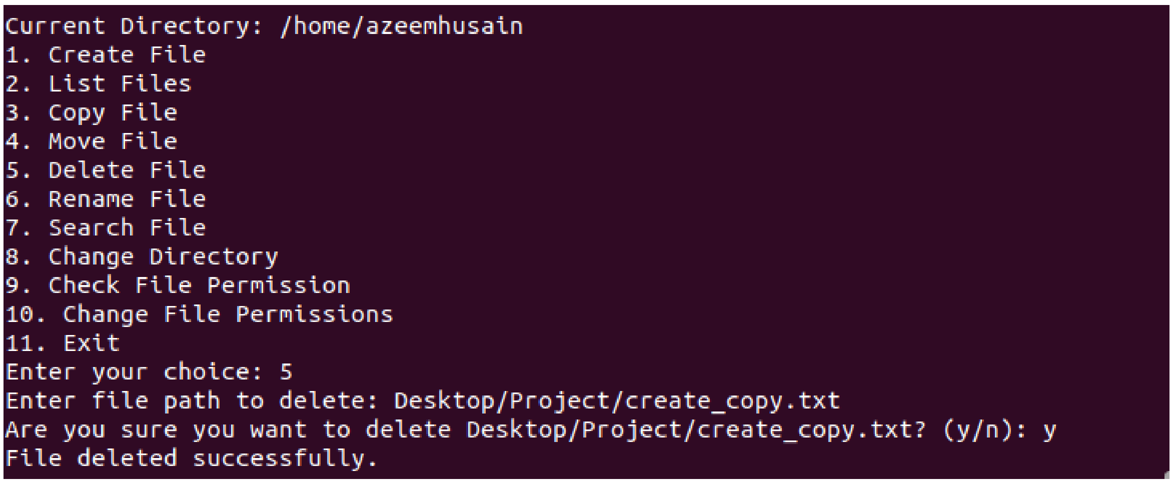
Move File



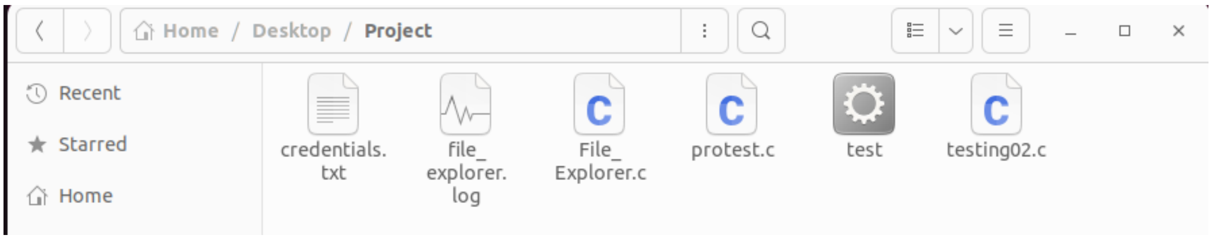
```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 4
Enter source file path: Desktop/create_copy.txt
Enter destination file path: Desktop/Project
```



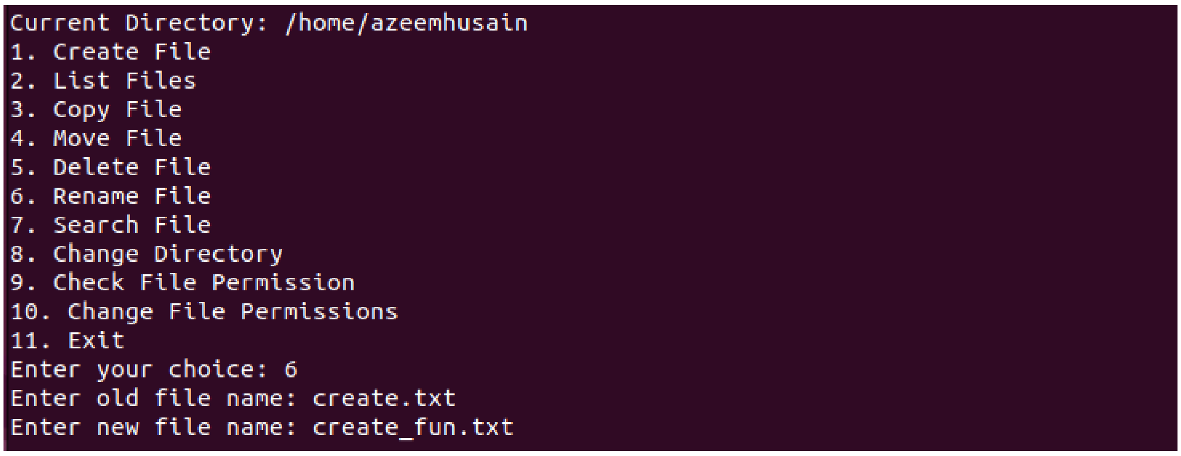
Delete File



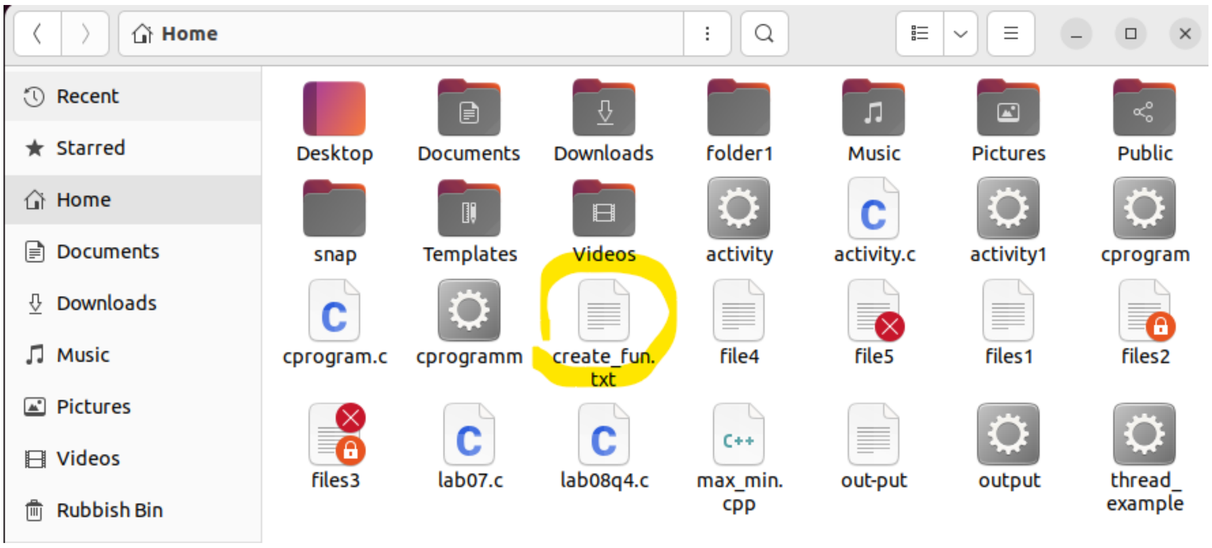
```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 5
Enter file path to delete: Desktop/Project/create_copy.txt
Are you sure you want to delete Desktop/Project/create_copy.txt? (y/n): y
File deleted successfully.
```



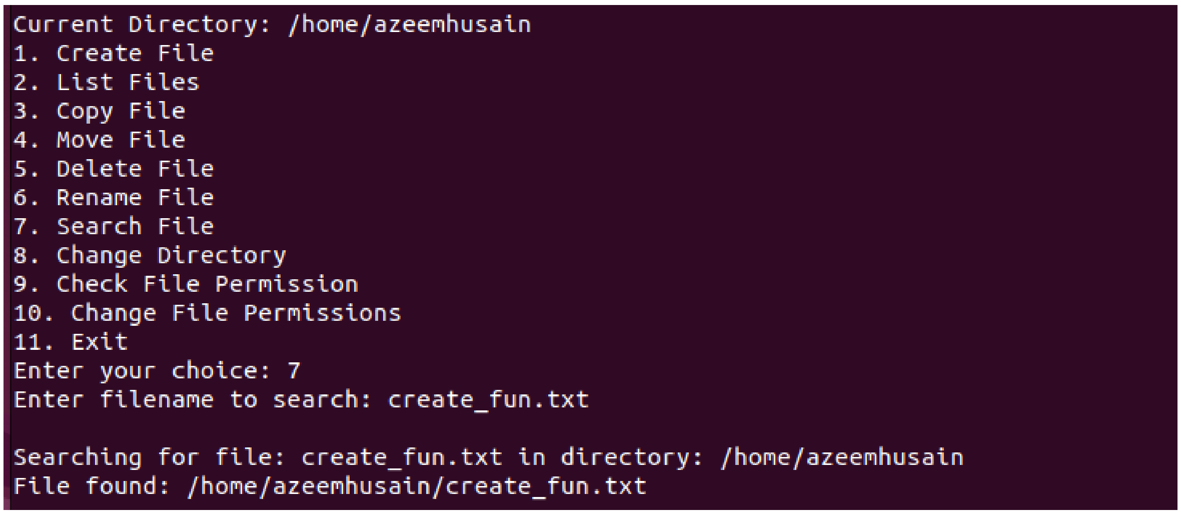
Rename



```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 6
Enter old file name: create.txt
Enter new file name: create_fun.txt
```

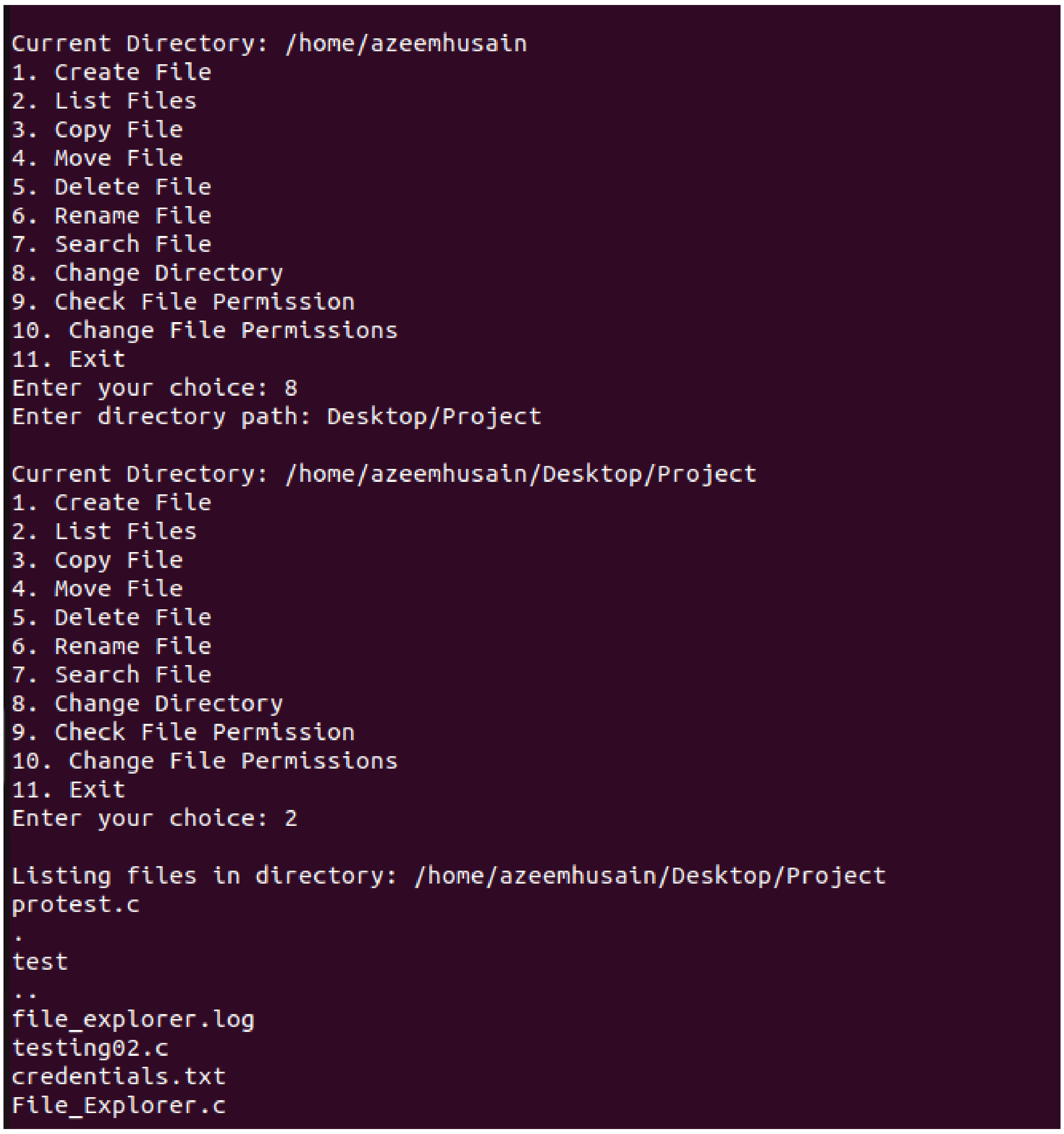


Search File



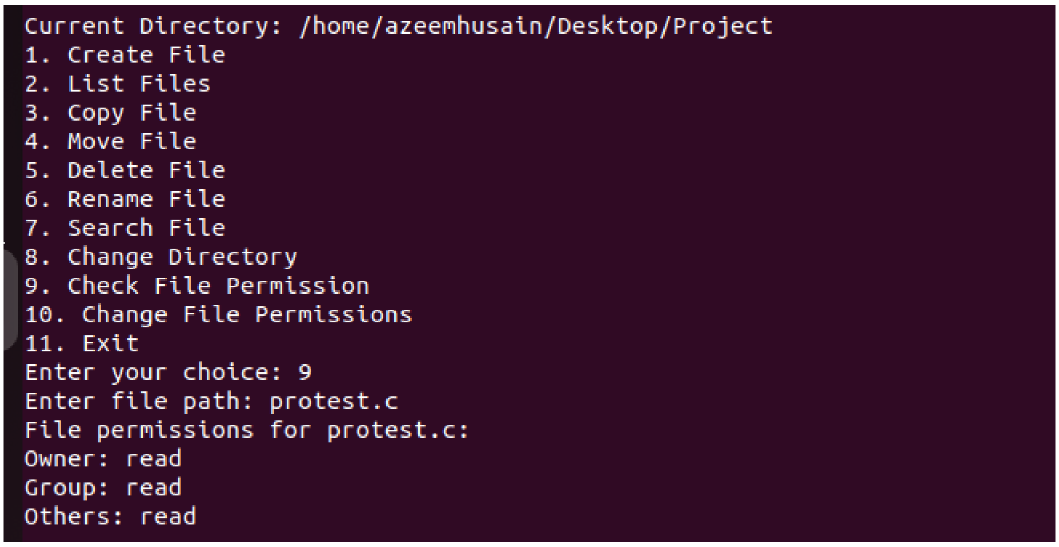
```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 7
Enter filename to search: create_fun.txt

Searching for file: create_fun.txt in directory: /home/azeemhusain
File found: /home/azeemhusain/create_fun.txt
```

Change Directory



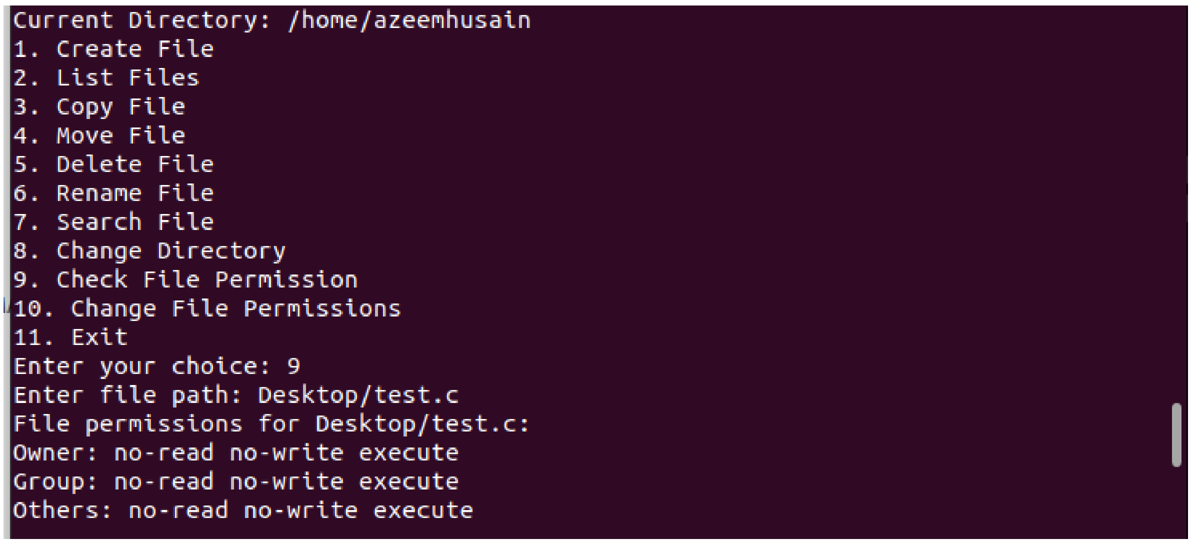
```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 8
Enter directory path: Desktop/Project

Current Directory: /home/azeemhusain/Desktop/Project
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 2

Listing files in directory: /home/azeemhusain/Desktop/Project
protest.c
.
test
..
file_explorer.log
testing02.c
credentials.txt
File_Explorer.c
```

Check File Permission

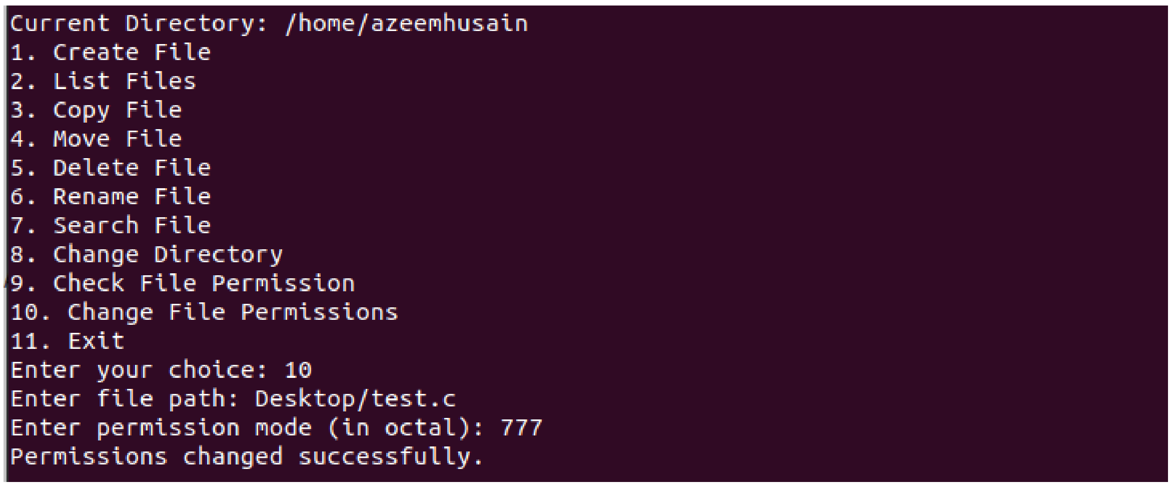


```
Current Directory: /home/azeemhusain/Desktop/Project
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 9
Enter file path: protest.c
File permissions for protest.c:
Owner: read
Group: read
Others: read
```

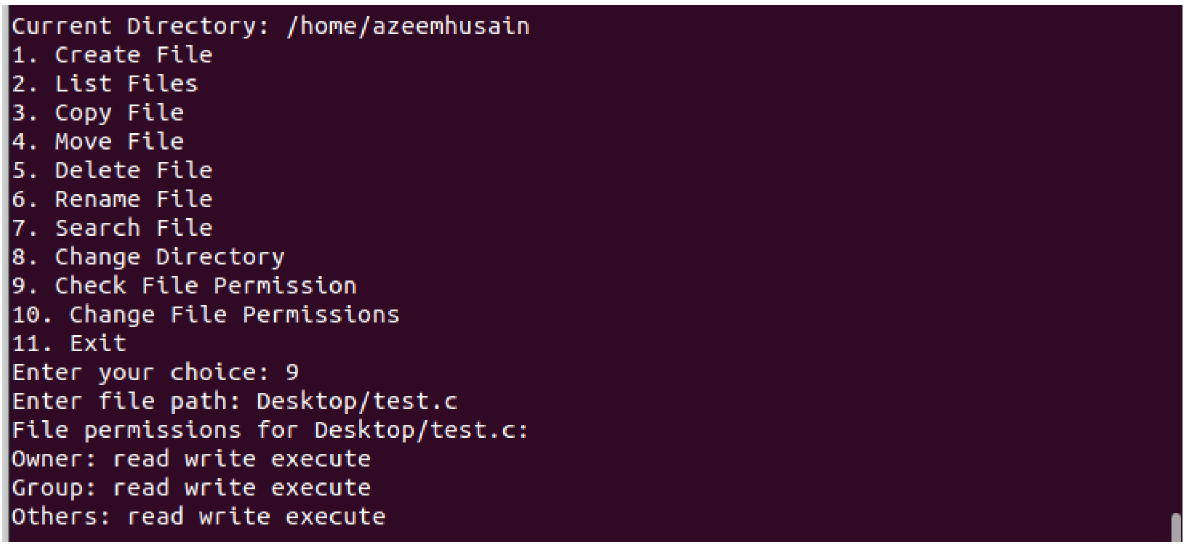
Change File Permissions



```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 9
Enter file path: Desktop/test.c
File permissions for Desktop/test.c:
Owner: no-read no-write execute
Group: no-read no-write execute
Others: no-read no-write execute
```

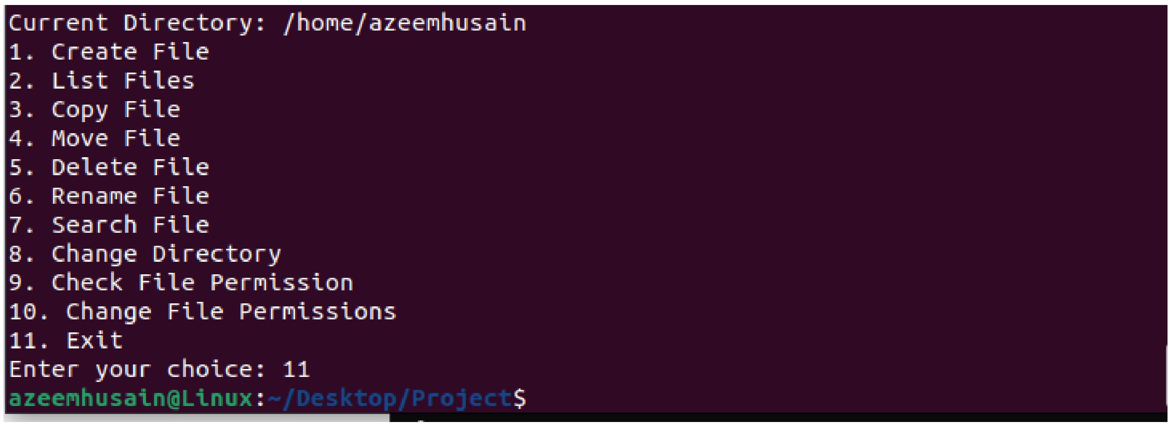


```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 10
Enter file path: Desktop/test.c
Enter permission mode (in octal): 777
Permissions changed successfully.
```



```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 9
Enter file path: Desktop/test.c
File permissions for Desktop/test.c:
Owner: read write execute
Group: read write execute
Others: read write execute
```

Exit



```
Current Directory: /home/azeemhusain
1. Create File
2. List Files
3. Copy File
4. Move File
5. Delete File
6. Rename File
7. Search File
8. Change Directory
9. Check File Permission
10. Change File Permissions
11. Exit
Enter your choice: 11
azeemhusain@Linux:~/Desktop/Project$
```

6. Usage:

- ❖ Users can compile the source code using a C compiler such as GCC on Ubuntu OS. Once compiled, the executable can be run from the terminal. The user is presented with a menu of options to choose from. They can then select the desired operation by entering the corresponding number.

7. Conclusion:

- The Basic File Explorer for Ubuntu OS provides a simple yet effective solution for basic file management tasks on the Ubuntu operating system. It offers essential functionalities such as creating, listing, copying, moving, deleting files, and more, all accessible through a user-friendly command-line interface. While the project currently focuses on basic functionalities, there is potential for further enhancement and expansion to support additional features and improvements in the future.