Project Title:
FILE SYSTEM CHANGE MAKER

Name: Prabhjot Singh

Subject: Programming in C

Batch: B-45

SAP-ID:590023045

Submitted to: Mrs Dolly Dass

Date: 28-11-2025

GitHub Repo for code:

https://github.com/Prabhjotsingh1509/FILE-CHANGE-MAKER-SYSTEM

ABSTRACT

The File System Change Maker is a command-line utility developed in the C programming language to simplify common file system operations.

The system allows users to create folders, create files, rename files, delete files or folders, and list directory contents. Using standard C libraries.

The project provides an efficient and user-friendly way to manipulate files without requiring manual commands.

This tool is intended for users who need frequent file operations for organizational or automation tasks. The system is lightweight, modular, and built with separate header files for each functionality, improving reusability and maintainability.

Problem Definition

File system operations such as creating, deleting, or renaming files normally require manual steps through graphical interfaces or typing commands in a terminal.

These methods can be slow, error-prone, and difficult for beginners.
The purpose of this project is to provide a **single interactive menu-driven utility that performs essential file system operations quickly and reliably**.

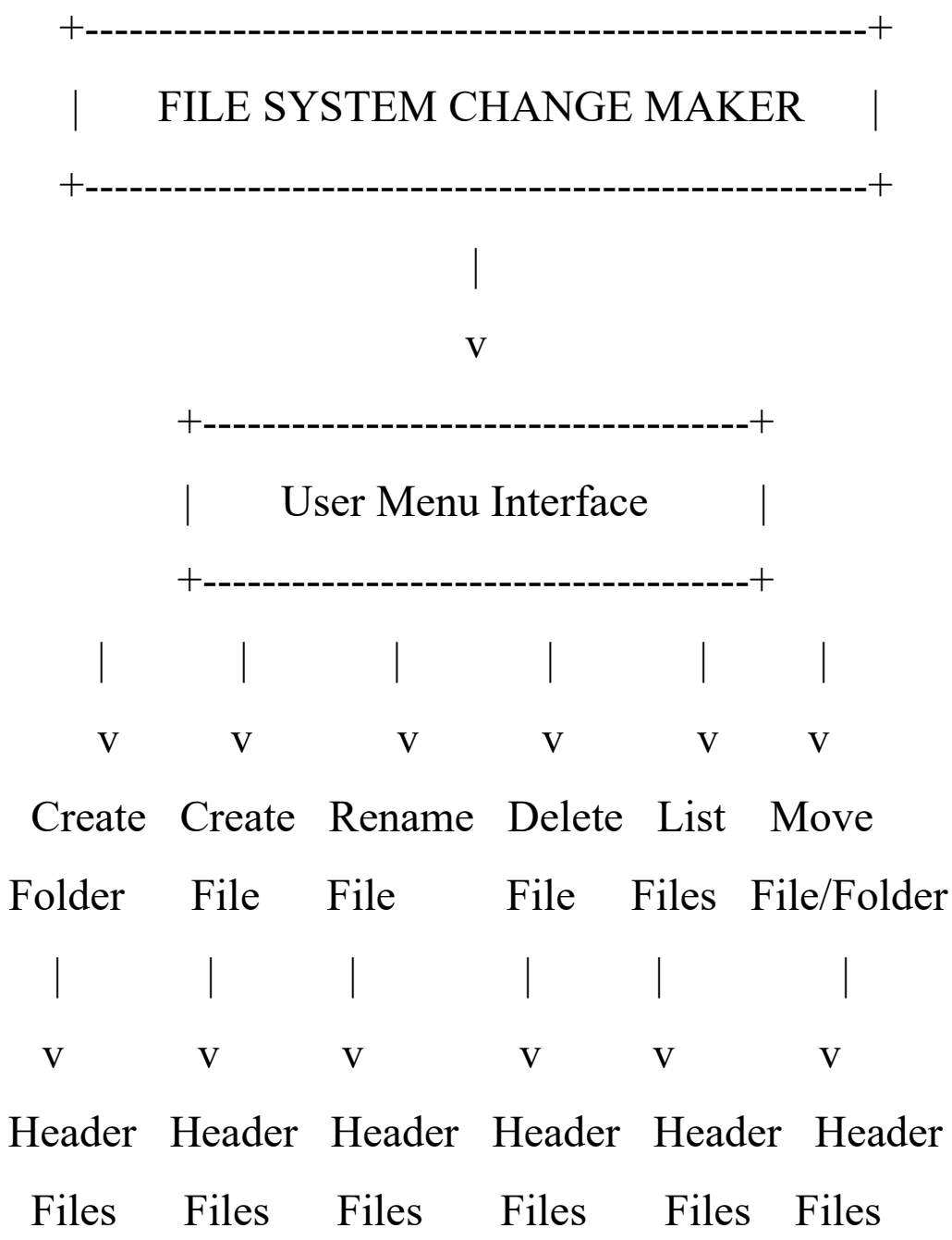
Benefit of Making this Project:

By using simple numbered options, the user can easily perform tasks without knowing OS commands or technical details.

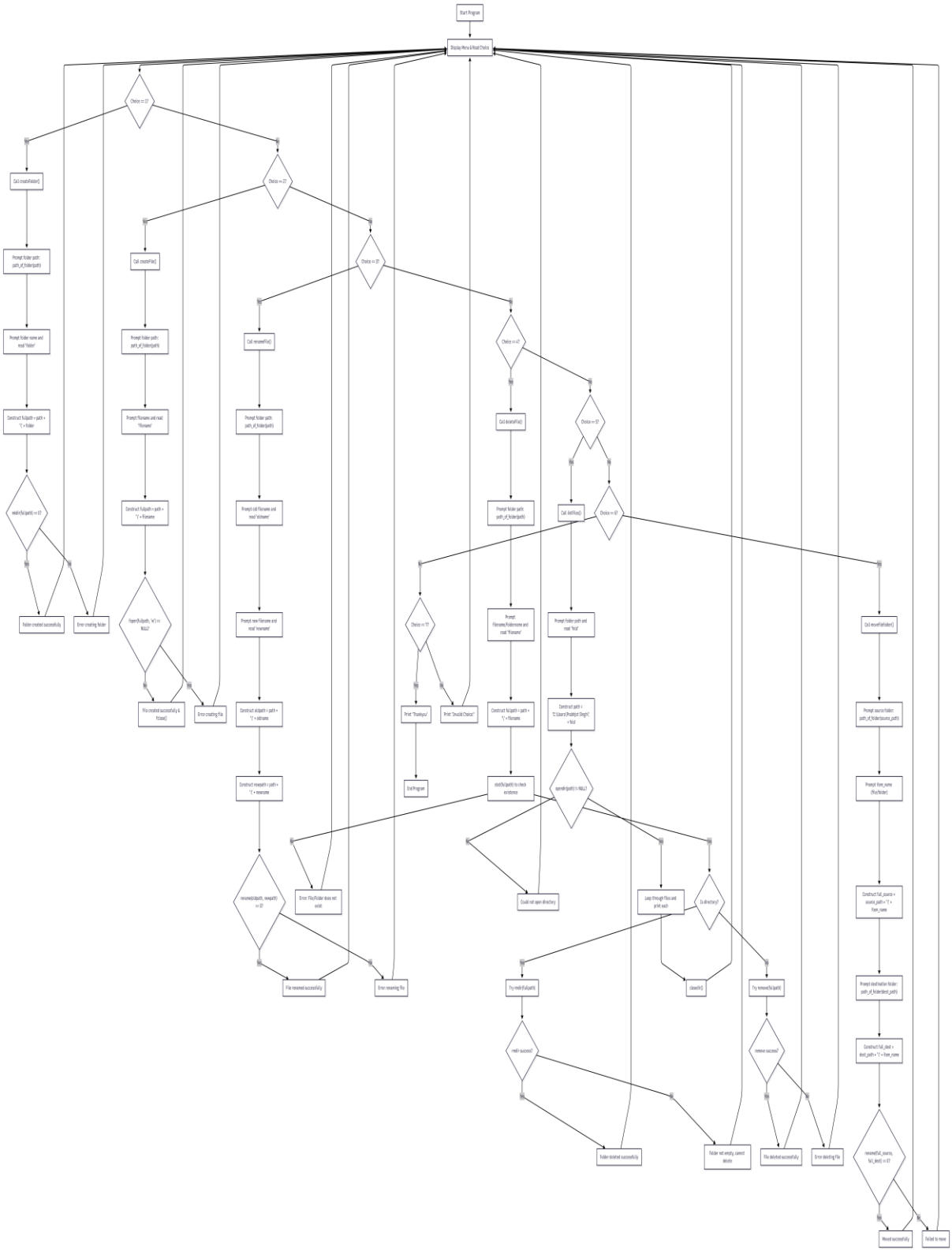
System Design

4.1 System Architecture

(For main)



4.2 Flowchart



4.3 Algorithm

1. Start Program

- Initialize the program and display the main menu repeatedly until the user chooses to exit.

---

2. Display Main Menu Options

1. Create Folder
2. Create File
3. Rename File
4. Delete File or Folder
5. List Files in Directory
6. Move File or Folder
7. Exit

---

3. Accept User Input

- Read the user's choice and store it in choice.

---

4. Execute Operation (Using switch-case)

Case 1: Create Folder

1. Call path_of_folder() to get the parent folder path.
2. Ask for new folder name.
3. Combine path and folder name to create full path.
4. Call mkdir() to create the folder.
5. Print success or error message.

Case 2: Create File

1. Call path_of_folder() to get the folder path.

2. Ask for filename.
3. Combine path and filename to create full path.
4. Call fopen() with "w" mode to create the file.
5. Print success or error message.

Case 3: Rename File

1. Call path_of_folder() to get the folder path.
2. Ask for old filename and new filename.
3. Combine folder path with old and new filenames.
4. Call rename() function.
5. Print success or error message.

Case 4: Delete File or Folder

1. Call path_of_folder() to get folder path
   Ask for filename or folder name to delete.
2. Check if the path exists using stat().

3. If folder → call rmdir() (only works if folder is empty).
4. If file → call remove().
5. Print success or error message.

Case 5: List Files in Directory

1. Ask for folder name.
2. Combine folder name with base path (C:\Users\Prabhjot Singh\).
3. Open folder using opendir().
4. Loop through files using readdir() and display names.
5. Print error if folder cannot be opened.

Case 6: Move File or Folder

1. Call path_of_folder() to get source folder path.
2. Ask for the file/folder name to move.
3. Combine to form full source path.
4. Call path_of_folder() to get destination folder path.

5. Combine to form full destination path.
6. Call rename(full_source, full_dest) to move.
7. Print success or error message.

Case 7: Exit

Terminate the program with a thank you message.

5. Loop

- After completing any operation (except Exit), return to Step 2 and display the menu again.

6.End Program

Exit gracefully when user chooses option 7.

Implementation Details (with snippets)

The File System Change Maker is implemented in C language using a modular approach. Each operation has its own header and source file. The program interacts with the user through a menu-driven interface, performs file operations, and loops until the user exits.

Main Menu

The main menu displays the available operations and calls the corresponding functions using switch-case:

```c
int main(void)
{
    int choice;

    while (1)
    {
        printf("\n===== FILE SYSTEM CHANGE MAKER =====\n");
        printf("1. Create Folder\n");
        printf("2. Create File\n");
        printf("3. Rename File\n");
        printf("4. Delete File or Folder\n");
        printf("5. List Files in Directory\n");
        printf("6. move the file or folder\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();
        switch (choice)
        {
        case 1:
            createFolder();
            break;
        case 2:
            createFile();
            break;
        case 3:
            renameFile();
            break;
        case 4:
            deleteFile();
            break;
        case 5:
            listFiles();
            break;
        case 6:
            movefilefolder();
            break;
        case 7:
            printf("Thankyou");
            return 0;
            break;
        default:
            printf("Invalid choice!\n");
        }
    }
}
```

Create Folder

Creates a new folder at the specified path :

```c
void createFolder(void)
{
    char path[100], folder[50], fullpath[200];

    path_of_folder(path);
    fflush(stdin);
    printf("Enter new folder name: ");
    scanf("%s", folder);

    sprintf(fullpath, "%s\\%s", path, folder);


    if (mkdir(fullpath) == 0)
        printf("Folder '%s' created successfully at: %s\n", folder, fullpath);
    else
        printf("Error creating folder.\n");
}
```

Create File

Creates a new file inside the selected folder:

```c
void createFile(void) {
    char path[200];
    char filename[50];
    char fullpath[200];

    path_of_folder(path);

    FILE *fp;
    printf("Enter filename: ");
    scanf("%s", filename);

    sprintf(fullpath,"%s\\%s",path,filename);
    fp = fopen(fullpath, "w");
    if (fp == NULL)
        printf("Error creating file.\n");
    else {
        printf("File '%s' created successfully.\n", filename);
        fclose(fp);
    }
}
```

Rename File

Renames an existing file:

```c
void renameFile(void)
{
    char path[50];
    char filename[100];
    char oldpath[200],newpath[200];
    char oldname[100], newname[100];

    path_of_folder(path);

    printf("Enter old filename: ");
    fgets(oldname,100,stdin);
    oldname[strcspn(oldname, "\n")] = '\0';

    fflush(stdin);
    printf("Enter new filename: ");
    fgets(newname,100,stdin);
    newname[strcspn(newname, "\n")] = '\0';

    sprintf(oldpath,"%s\\%s",path,oldname);
    sprintf(newpath,"%s\\%s",path,newname);

    if (rename(oldpath, newpath) == 0)
        printf("File renamed successfully.\n");
    else
        printf("Error renaming file.\n");
}
```

Delete File or Folder

Deletes a file or folder (folder must be empty):

```c
void deleteFile(void) {

    char path[200];
    char fullpath[200];
    char filename[50];

    struct stat s;
    path_of_folder(path);

    printf("Enter filename or foldername to delete: ");
    scanf("%s", filename);
    sprintf(fullpath, "%s\\%s", path, filename);
    if (stat(fullpath, &s) != 0) {
        printf("Error: File or folder does not exist.\n");
        return;
    }


    if (s.st_mode & S_IFDIR)
    {

        if (rmdir(fullpath) == 0) {
            printf("Folder deleted successfully.\n");
        } else {
            printf("Folder is not empty. Cannot delete with rmdir().\n");
        }

    } else
    {
        if (remove(fullpath) == 0)
            printf("File deleted successfully.\n");
        else
            printf("Error deleting file.\n");
    }
}
```

List Files in Directory

Displays all files in the selected folder:

```c
void listFiles()
{
    char fold[50],path[50];
    DIR *p;
    struct dirent *dir;
    printf("Enter the folder name:");
    scanf("%s",&fold);
    fflush(stdin);
    sprintf(path, "C:\\Users\\Prabhjot Singh\\%s", fold);
    p=opendir(path);
    if (p)
    {
        printf("\nFiles in '%s':\n", path);
        while ((dir = readdir(p)) != NULL)
            printf("%s\n", dir->d_name);
        closedir(p);
    } else
    {
        printf("Could not open directory.\n");
    }

}
```

Move File or Folder

Moves a file or folder from source to destination:

```c
void movefolder(void)
{
    char source_path[200];
    char dest_path[200];
    char item_name[100];

    printf("\n===== MOVE FILE OR FOLDER =====\n");
    path_of_folder(source_path);

    if (strlen(source_path) == 0)
    {
        printf("Invalid source path.\n");
        return;
    }

    printf("Enter the name of the file/folder to move: ");
    fgets(item_name, sizeof(item_name), stdin);
    item_name[strcspn(item_name, "\n")] = '\0';

    char full_source[300];
    sprintf(full_source, "%s\\%s", source_path, item_name);

    printf("\nEnter DESTINATION folder name inside C:\\Users\\Prabhjot Singh\\ :\n");
    path_of_folder(dest_path);

    if (strlen(dest_path) == 0)
    {
        printf("Invalid destination path.\n");
        return;
    }

    char full_dest[300];
    sprintf(full_dest, "%s\\%s", dest_path, item_name);

    if (rename(full_source, full_dest) == 0)
    {
        printf("\nMoved successfully!\n");
        printf("From: %s\n", full_source);
        printf("To:   %s\n", full_dest);
    }
    else
    {
        printf("\nFailed to move.\n");
        perror("Error");
    }
}
```

Notes on Implementation

- Each module uses path_of_folder() for consistency in getting folder paths.
- Modular design improves readability and maintainability.
- Error handling ensures the program handles invalid paths, non-existent files, and non-empty folders gracefully.
- rename() is used both for **renaming** and **moving** files/folders

Testing and Results

All the proof in screenshots in the assets folder

Test Case	Input	Expected Output	Result
1. Create Folder	Folder name: test	“Folder created successfully”	Passed
2. Create File	Filename: sample.txt	File appears in directory	Passed
3. Rename File	sample.txt → new.txt	File renamed successfully	Passed
4. Delete Item	new.txt	File removed from directory	Passed
5. List Files	—	Displays correct files and folders	Passed
6. Move Folder	Folder name: Prabh , Source folder: Desktop, Destination folder: Document	Folder appears in destination folder	Passed

Conclusion & Future Work

Conclusion

The File System Change Maker successfully performs all essential file system operations, including:

- Creating folders and files
- Renaming files
- Deleting files and folders
- Listing directory contents
- Moving files and folders

The system uses a simple, menu-driven interface, making it intuitive and easy to use, even for beginners learning file handling in C. The program is lightweight, modular, and efficient, with proper error handling to manage invalid paths and operations.

Overall, the project demonstrates practical file system management in C and provides a solid foundation for further enhancements.

Future Work

The current system can be extended with the following features:

1. Graphical User Interface (GUI):

Implement a user-friendly GUI using GTK, WinAPI, or Qt for easier interaction.

2. File Copy and Advanced Move Operations:

Allow copying files/folders and batch move operations.

3. File Search Functionality:

Enable users to search files/folders by name, type, or size.

4. Logging System:

Maintain a log of all operations performed for audit or tracking purposes.

5. Permission Handling:

Implement read/write/execute permissions to control access to files and folders.

6. Additional Features:

Support for nested folder operations, undo/redo operations, and integration with cloud storage.