TITLE :-FILE EXPLORER APPLICATION (CAPSTONE PROJECT 1)

Batch: Wipro COE Embedded Batch-3

Name - Aniket Nayak

Registration Number: 2241014056

Course: B.Tech

Branch: CSE (AI & ML)

Semester: 7th

Institution: Siksha 'O' Anusandhan, ITER

Date: 09-November-2025

INTRODUCTION

Managing files and directories efficiently is an essential aspect of any operating system, especially for system users and developers working in a Linux environment. This project aims to provide a console-based file explorer application written in C++, enabling users to perform core file operations directly from the terminal. By leveraging modern C++ and the Linux file system, the application streamlines common tasks such as navigating directories, manipulating files, and managing permissions.

OBJECTIVE

The objective of this project is to develop a robust, menu-driven file explorer for Linux, using the C++ programming language and its standard libraries. The tool is designed to allow users to perform essential file and directory operations—including listing, creating, deleting, copying, moving, searching, and managing permissions—through a simple, interactive console interface. This application not only enhances productivity but also familiarizes users with system-level programming concepts and Linux filesystem management.

APPLICATION FEATURES

Feature	**Description**
List files	Display all files/directories in the current working directory.
Change directory	Move between directories interactively.
Create file/directory	Create new files and directories as needed.
Delete file/directory	Remove files or entire directories recursively.
Copy/Move operations	Copy or move files and directories with safety checks.
Search file	Search for files recursively within the current directory tree.
Show permissions	Display detailed file or directory permission settings.
Set permissions	Update permissions (read/write/execute) for files and folders.

DEVELOPMENT ENVIRONMENT :

- **Language:** C++
- **Libraries:** C++17 <filesystem>, <fstream>, <iostream>, <string>
- **Platform:** Linux (tested with WSL)
- **Editor:** Visual Studio Code
- **Compiler:** GCC/G++ (C++17 standard).

IMPLEMENTATION OF FILE EXPLORER APPLICATION IN C++ :

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include <string>

// Use filesystem namespace for convenience
namespace fs = std::filesystem;
using namespace std;

// List all files and directories at the given path
void listFiles(const fs::path &path) {
    cout << "\nFiles in Directory: " << path << "\n";
    try {
        for (const auto &entry : fs::directory_iterator(path)) {
            cout << (entry.is_directory() ? "[DIR] " : "[FILE] ")
                 << entry.path().filename().string() << "\n";
        }
    } catch (const fs::filesystem_error &e) {
        cerr << "Error: " << e.what() << endl;
    }
}

// Change current working directory if path exists
```

```
void changeDirectory(fs::path &currentPath, const string &dirName) {
    fs::path newPath = currentPath / dirName;
    if (fs::exists(newPath) && fs::is_directory(newPath)) {
        currentPath = fs::canonical(newPath);
    } else {
        cout << "Directory not found.\n";
    }
}

// Create a new file at the specified path
void createFile(const fs::path &filePath) {
    ofstream file(filePath);
    if (file) cout << "File created: " << filePath.filename().string()
<< "\n";
    else cout << "Failed to create file.\n";
}

// Create a new directory at the specified path
void createDirectory(const fs::path &dirPath) {
    if (fs::create_directory(dirPath))
        cout << "Directory created: " << dirPath.filename().string()
<< "\n";
    else
        cout << "Failed to create directory.\n";
}

// Delete a file or directory (recursively for directories)
void deleteItem(const fs::path &path) {
    try {
        if (fs::is_directory(path)) fs::remove_all(path);   // Delete
directory recursively
```

```
        else if (fs::is_regular_file(path)) fs::remove(path); //
Delete regular file
        else cout << "Path not found.\n";
        cout << "Deleted.\n";
    } catch (const fs::filesystem_error &e) {
        cerr << "Error deleting: " << e.what() << endl;
    }
}

// Copy a file or directory to a destination
void copyItem(const fs::path &source, const fs::path &destination) {
    try {
        if (fs::is_directory(source))
            fs::copy(source, destination,
fs::copy_options::recursive);
        else
            fs::copy_file(source, destination,
fs::copy_options::overwrite_existing);
        cout << "Copied successfully.\n";
    } catch (const fs::filesystem_error &e) {
        cerr << "Error copying: " << e.what() << endl;
    }
}

// Move a file or directory to a new destination
void moveItem(const fs::path &source, const fs::path &destination) {
    try {
        fs::rename(source, destination);
        cout << "Moved successfully.\n";
    } catch (const fs::filesystem_error &e) {
        cerr << "Error moving: " << e.what() << endl;
    }
```

```
}

// Search for a file by name in the current directory and
subdirectories
void searchFile(const fs::path &path, const string &fileName) {
    try {
        for (const auto &entry :
fs::recursive_directory_iterator(path)) {
            if (entry.path().filename() == fileName)
                cout << "Found: " << entry.path() << "\n";
        }
    } catch (const fs::filesystem_error &e) {
        cerr << "Error searching: " << e.what() << endl;
    }
}

// Display permissions for a given file or directory
void showPermissions(const fs::path &path) {
    auto perms = fs::status(path).permissions();
    cout << "Permissions for " << path.filename().string() << ": ";
    cout << ((perms & fs::perms::owner_read) != fs::perms::none ? "r"
: "-");
    cout << ((perms & fs::perms::owner_write) != fs::perms::none ? "w"
: "-");
    cout << ((perms & fs::perms::owner_exec) != fs::perms::none ? "x"
: "-");
    cout << ((perms & fs::perms::group_read) != fs::perms::none ? "r"
: "-");
    cout << ((perms & fs::perms::group_write) != fs::perms::none ? "w"
: "-");
    cout << ((perms & fs::perms::group_exec) != fs::perms::none ? "x"
: "-");
    cout << ((perms & fs::perms::others_read) != fs::perms::none ? "r"
: "-");
```

```
    cout << ((perms & fs::perms::others_write) != fs::perms::none ?
"w" : "-");
    cout << ((perms & fs::perms::others_exec) != fs::perms::none ? "x"
: "-");
    cout << "\n";
}

// Set new permissions for a file or directory
void setPermissions(const fs::path &path, fs::perms permissions) {
    fs::permissions(path, permissions);
    cout << "Permissions updated.\n";
}

// Main function: menu-driven interface for file explorer
int main() {
    fs::path currentPath = fs::current_path(); // Start at current
working directory
    int choice;
    while (true) {
        // Display menu options
        cout << "\nCurrent Directory: " << currentPath << "\n";
        cout << "1. List files\n2. Change directory\n3. Create
file\n4. Create directory\n";
        cout << "5. Delete file/directory\n6. Copy file/directory\n7.
Move file/directory\n";
        cout << "8. Search file\n9. Show permissions\n10. Set
permissions\n0. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();
        if (choice == 0) break; // Exit on 0

        string name, dest;
```

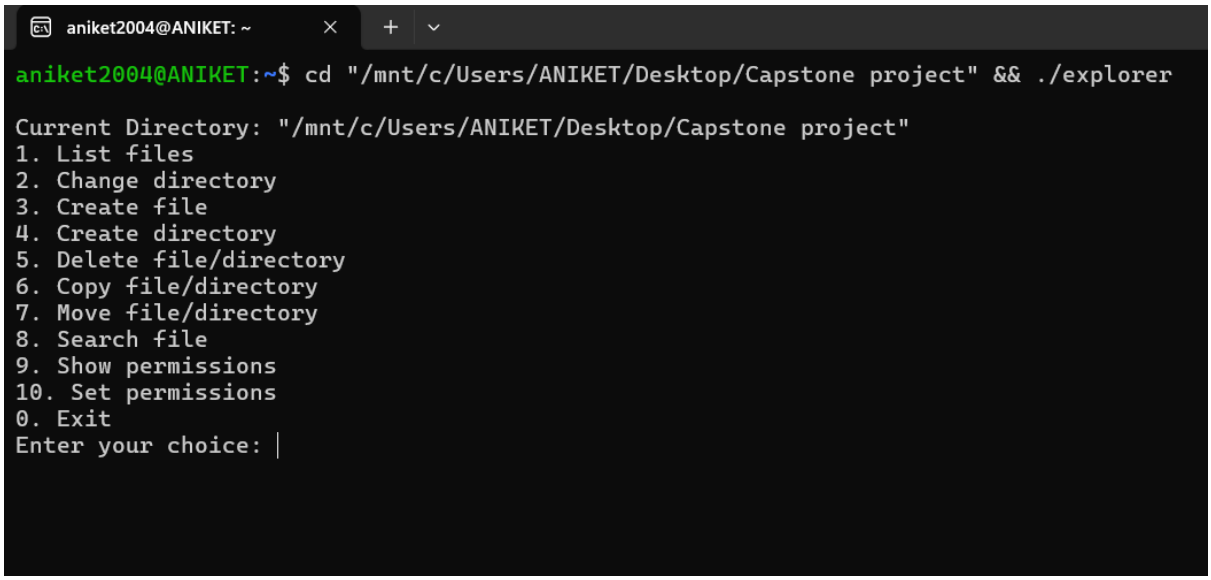
```
switch (choice) {
    case 1: listFiles(currentPath); break;
    case 2:
        cout << "Enter directory name: ";
        getline(cin, name);
        changeDirectory(currentPath, name);
        break;
    case 3:
        cout << "Enter file name: ";
        getline(cin, name);
        createFile(currentPath / name);
        break;
    case 4:
        cout << "Enter directory name: ";
        getline(cin, name);
        createDirectory(currentPath / name);
        break;
    case 5:
        cout << "Enter file/directory name: ";
        getline(cin, name);
        deleteItem(currentPath / name);
        break;
    case 6:
        cout << "Enter source name: ";
        getline(cin, name);
        cout << "Enter destination name: ";
        getline(cin, dest);
        copyItem(currentPath / name, currentPath / dest);
        break;
    case 7:
        cout << "Enter source name: ";
```

```
                getline(cin, name);
                cout << "Enter destination name: ";
                getline(cin, dest);
                moveItem(currentPath / name, currentPath / dest);
                break;
            case 8:
                cout << "Enter file name to search: ";
                getline(cin, name);
                searchFile(currentPath, name);
                break;
            case 9:
                cout << "Enter file/directory name: ";
                getline(cin, name);
                showPermissions(currentPath / name);
                break;
            case 10:
                cout << "Enter file/directory name: ";
                getline(cin, name);
                cout << "Enter permission mode (e.g., 644): ";
                int perm;
                cin >> oct >> perm;
                setPermissions(currentPath / name,
static_cast<fs::perms>(perm));
                cin.ignore();
                break;
            default:
                cout << "Invalid choice.\n";
        }
    }
    cout << "Exiting File Explorer...\n";
    return 0;
```

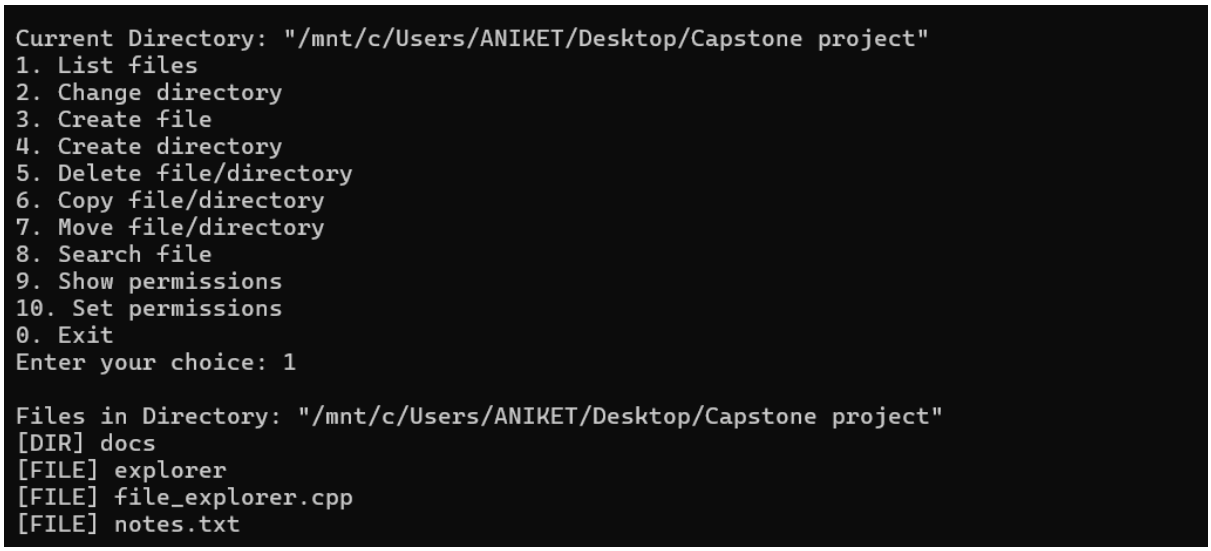
}

PROGRAM TESTING AND RESULTS:

1. **Application Launch and Main Menu**



2. **Listing Files in Directory**



3. **Changing Directory**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 2
Enter directory name: docs
```

4. **Creating a New File**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project/docs"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 3
Enter file name: aniket_files.txt
File created: aniket_files.txt
```

5. **Creating a New Directory**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project/docs"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 4
Enter directory name: test_folder
Directory created: test_folder
```

6. **Deleting a File or Directory**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project/docs"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 5
Enter file/directory name: aniket_files.txt
Deleted.
```

7. **Copying a File or Directory**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 6
Enter source name: notes.txt
Enter destination name: notes_backup.txt
Copied successfully.
```

8. **Moving a File or Directory**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 7
Enter source name: notes.txt
Enter destination name: docs/notes.txt
Moved successfully.
```

9. **Searching for a File**

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 8
Enter file name to search: file_explorer.cpp
Found: "/mnt/c/Users/ANIKET/Desktop/Capstone project/file_explorer.cpp"
```

10. Displaying Permissions of File/Directory

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 9
Enter file/directory name: file_explorer.cpp
Permissions for file_explorer.cpp: rwxrwxrwx
```

11. Setting Permissions of File/Directory

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 10
Enter file/directory name: notes_backup.txt
Enter permission mode (e.g., 644): 600
Permissions updated.
```

12. Exiting the File Explorer Application

```
Current Directory: "/mnt/c/Users/ANIKET/Desktop/Capstone project"
1. List files
2. Change directory
3. Create file
4. Create directory
5. Delete file/directory
6. Copy file/directory
7. Move file/directory
8. Search file
9. Show permissions
10. Set permissions
0. Exit
Enter your choice: 0
Exiting File Explorer...
aniket2004@ANIKET:/mnt/c/Users/ANIKET/Desktop/Capstone project$
```

CONCLUSION

This File Explorer Application in C++ works well for managing files and folders in Linux through the command line. It lets users do many important tasks like creating, deleting, moving, and searching files easily. This project helped me learn how to use C++ to work with the file system and how to write programs that interact with the operating system. Overall, this is a useful tool that can be improved more in the future.