File Explorer App – Capstone Project

Name: Rudramadhab Rath

Roll Number: 2241018081

Course: Capstone Project

Institution: ITER

Date: 9th November 2025

File Explorer Application

Objective:
Develop a console-based file explorer application in **C++** that interfaces with the **Linux operating system** to manage files and directories efficiently.

Technology Stack

1. Operating System: Ubuntu 22.04 LTS (Linux)
2. Programming Language: C++ (Standard: C++17)
3. Compiler: GNU G++ (g++)
4. Libraries/Utilities: Standard Template Library (STL), File System Library (<filesystem>)
5. Version Control: Git and GitHub

Implementation & Code

```
#include <iostream>

#include <filesystem>
```

```
#include <fstream>
#include <string>
using namespace std;
namespace fs = filesystem;
void listDirectory(const fs::path &path) {
    cout << "\nContents of " << path << ":\n";
    for (const auto &entry : fs::directory_iterator(path)) {
        cout << (entry.is_directory() ? "[DIR] " : "[FILE] ")
             << entry.path().filename().string() << endl;
    }
}
void createFile(const fs::path &path) {
    ofstream file(path);
    if (file) {
        cout << "File created: " << path.filename().string() << endl;
    } else {
        cout << "Error creating file!" << endl;
    }
}

void deleteFile(const fs::path &path) {
    if (fs::exists(path)) {
```

```
        fs::remove_all(path);
        cout << "Deleted: " << path.filename().string() << endl;
    } else {
        cout << "File or directory not found!" << endl;
    }
}
void copyFile(const fs::path &src, const fs::path &dest) {
    try {
        fs::copy(src, dest, fs::copy_options::overwrite_existing |
fs::copy_options::recursive);
        cout << "Copied " << src.filename().string() << " to " << dest.string() << endl;
    } catch (exception &e) {
        cout << "Error copying file: " << e.what() << endl;
    }
}
void moveFile(const fs::path &src, const fs::path &dest) {
    try {
        fs::rename(src, dest);
        cout << "Moved " << src.filename().string() << " to " << dest.string() << endl;
    } catch (exception &e) {
        cout << "Error moving file: " << e.what() << endl;
    }
```

```
}
void searchFile(const fs::path &path, const string &filename) {
    cout << "Searching for \"" << filename << "\" in " << path << "...\n";
    bool found = false;
    for (const auto &entry : fs::recursive_directory_iterator(path)) {
        if (entry.path().filename() == filename) {
            cout << "Found at: " << entry.path() << endl;
            found = true;
        }
    }
    if (!found)
        cout << "No file named \"" << filename << "\" found.\n";
}
void changePermissions(const fs::path &path, fs::perms permissions) {
    try {
        fs::permissions(path, permissions);
        cout << "Permissions updated for: " << path.filename().string() << endl;
    } catch (exception &e) {
        cout << "Error changing permissions: " << e.what() << endl;
    }
}
int main() {
```

```
    fs::path currentPath = fs::current_path();
    string command;
    cout << "===== Simple File Explorer (C++ - LinuxOS) =====\n";
    cout << "Type 'help' to view available commands.\n";

    while (true) {
        cout << "\n[" << currentPath << "]$ ";
        cin >> command;
        if (command == "ls") {
            listDirectory(currentPath);
        }
        else if (command == "cd") {
            string dir;
            cin >> dir;
            fs::path newPath = (dir == "..") ? currentPath.parent_path() : currentPath /
dir;
            if (fs::exists(newPath) && fs::is_directory(newPath)) {
                currentPath = newPath;
            } else {
                cout << "Invalid directory.\n";
            }
        }
```

```
else if (command == "create") {
    string filename;
    cin >> filename;
    createFile(currentPath / filename);
}
else if (command == "delete") {
    string name;
    cin >> name;
    deleteFile(currentPath / name);
}
else if (command == "copy") {
    string src, dest;
    cin >> src >> dest;
    copyFile(currentPath / src, currentPath / dest);
}
else if (command == "move") {
    string src, dest;
    cin >> src >> dest;
    moveFile(currentPath / src, currentPath / dest);
}
else if (command == "search") {
    string name;
```

```
    cin >> name;
    searchFile(currentPath, name);
}
else if (command == "chmod") {
    string filename;
    string mode;
    cin >> filename >> mode;
    try {
        fs::perms permissions = static_cast<fs::perms>(stoi(mode, nullptr, 8));
        changePermissions(currentPath / filename, permissions);
    } catch (...) {
        cout << "Invalid permission format. Use octal (e.g., 644)." << endl;
    }
}
else if (command == "help") {
    cout << "\nAvailable commands:\n"
        << "ls                - List files and directories\n"
        << "cd <dir>          - Change directory\n"
        << "create <file>     - Create a file\n"
        << "delete <file/dir>   - Delete file or directory\n"
        << "copy <src> <dest>   - Copy file or folder\n"
        << "move <src> <dest>   - Move or rename file/folder\n"
```

```
            << "search <filename>    - Search file recursively\n"
            << "chmod <file> <mode>  - Change file permission (octal)\n"
            << "help                 - Show available commands\n"
            << "exit                 - Quit program\n";
        }
        else if (command == "exit") {
            cout << "Exiting File Explorer...\n";
            break;
        }
        else {
            cout << "Unknown command! Type 'help' for options.\n";
        }
    }
    return 0;
}
```

Results and Observations

```
gitul@RudramadhabRath23H2:/mnt/c/Users/asus/Desktop/wipro$ g++ main.cpp -o file_explorer -std=c++17
gitul@RudramadhabRath23H2:/mnt/c/Users/asus/Desktop/wipro$ ls
a.out  file_explorer  main  main.cpp
gitul@RudramadhabRath23H2:/mnt/c/Users/asus/Desktop/wipro$ ./file_explorer
===== Simple File Explorer (C++ - LinuxOS) =====
Type 'help' to view available commands.

["/mnt/c/Users/asus/Desktop/wipro"]$ help

Available commands:
ls                  - List files and directories
cd <dir>            - Change directory
create <file>       - Create a file
delete <file/dir>   - Delete file or directory
copy <src> <dest>   - Copy file or folder
move <src> <dest>   - Move or rename file/folder
search <filename>   - Search file recursively
chmod <file> <mode> - Change file permission (octal)
help                - Show available commands
exit                - Quit program

["/mnt/c/Users/asus/Desktop/wipro"]$ ls

Contents of "/mnt/c/Users/asus/Desktop/wipro":
[FILE] a.out
[FILE] file_explorer
[FILE] main
[FILE] main.cpp

["/mnt/c/Users/asus/Desktop/wipro"]$ create test.txt
File created: test.txt

["/mnt/c/Users/asus/Desktop/wipro"]$ ls

Contents of "/mnt/c/Users/asus/Desktop/wipro":
[FILE] a.out
[FILE] file_explorer
[FILE] main
[FILE] main.cpp
[FILE] test.txt

["/mnt/c/Users/asus/Desktop/wipro"]$ delete test.txt
Deleted: test.txt

["/mnt/c/Users/asus/Desktop/wipro"]$ ls

Contents of "/mnt/c/Users/asus/Desktop/wipro":
[FILE] a.out
[FILE] file_explorer
[FILE] main
[FILE] main.cpp

["/mnt/c/Users/asus/Desktop/wipro"]$ exit
Exiting File Explorer...
gitul@RudramadhabRath23H2:/mnt/c/Users/asus/Desktop/wipro$
```

The application achieved its goal of providing an efficient, console-based file management system in C++ on Linux, demonstrating proper command execution, user interaction, and reliable performance.

Conclusion`

The **File Explorer Application** successfully met its objective of providing a command-line-based file management system in Linux using C++.
It enabled users to perform essential file operations such as creating, deleting, moving, copying, and searching through an interactive and user-friendly terminal interface.
The project demonstrated practical implementation of Linux system commands

and C++ file handling, showcasing strong integration between system-level operations and programming logic.

GitHub Repository

Public Repository Link: https://github.com/Mr-Coder23/RUDRAMADHAB_RATH/tree/main/wipro