

CAPSTONE PROJECT REPORT

Simple File Explorer Using C++ (Linux/WSL)

Student Name: Prem Sagar Padhy

Registration Number: 2241016021

Course: B.Tech, Computer Science Engineering

Institute: Institute of Technical Education and Research (ITER)

1. Introduction

This project implements a **terminal-based File Explorer** application using **C++17** and the `<filesystem>` library. It allows users to perform essential file system operations such as navigating directories, listing files, creating files, copying, moving, deleting, searching, and modifying file permissions.

The project works in a **Linux / WSL environment** and demonstrates understanding of **system-level programming** and **filesystem management**.

2. Objective

- To gain practical experience with **Linux file operations**.
- To understand and use the modern **C++17 filesystem API**.
- To build a **user-interactive terminal utility**.
- To implement **recursive search** and **permission handling**.
- To simulate a real-world file manager in command-line environment.

3. System Requirements

Component	Requirement
Operating System	Ubuntu / WSL in Windows
Programming Language	C++ (C++17 Standard)
Compiler	g++ (GNU Compiler Collection)
Editor	VS Code / Terminal
Libraries Used	<code><filesystem></code> , <code><chrono></code> , <code><iomanip></code> , <code><sstream></code>

5. Implementation Details

Commands Supported

Command	Function Performed	Example
ls	Lists files and directories	ls
ls -l	Long listing with permissions & size	ls -l
cd <dir>	Change current directory	cd testfolder
create <file>	Create a file	create notes.txt
delete <file/dir>	Delete file or folder	delete notes.txt
copy <src> <dest>	Copy file or folder	copy a.txt b.txt
move <src> <dest>	Move or rename file	move a.txt docs/
search <filename>	Search recursively	search a.txt
chmod <file> <mode>	Change permission (e.g., 644, 755)	chmod a.txt 644
exit	Exit application	exit

6. Source Code

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include <string>
#include <iomanip>
#include <chrono>
#include <sstream>
#include <ctime>

using namespace std;
namespace fs = std::filesystem;

// ----- time helpers -----
std::time_t to_time_t(fs::file_time_type ftime) {
    using namespace std::chrono;
    auto sctp = time_point_cast<system_clock::duration>(
        ftime - fs::file_time_type::clock::now() + system_clock::now()
    );
    return system_clock::to_time_t(sctp);
}

string timeString(std::tm* tm_ptr) {
    if (!tm_ptr) return "0000-00-00 00:00";
    std::ostringstream oss;
```

```

        oss << std::put_time(tm_ptr, "%Y-%m-%d %H:%M");
        return oss.str();
    }

// ----- permission helpers -----
string permString(fs::perms p) {
    auto bit = [&](fs::perms perm){ return (p & perm) != fs::perms::none; };
    string s;
    s += bit(fs::perms::owner_read) ? 'r' : '-';
    s += bit(fs::perms::owner_write) ? 'w' : '-';
    s += bit(fs::perms::owner_exec) ? 'x' : '-';
    s += bit(fs::perms::group_read) ? 'r' : '-';
    s += bit(fs::perms::group_write) ? 'w' : '-';
    s += bit(fs::perms::group_exec) ? 'x' : '-';
    s += bit(fs::perms::others_read) ? 'r' : '-';
    s += bit(fs::perms::others_write) ? 'w' : '-';
    s += bit(fs::perms::others_exec) ? 'x' : '-';
    return s;
}

void chmodOctal(const fs::path& p, const string& oct) {
    if (oct.size() < 3 || oct.size() > 4) throw runtime_error("Invalid mode (use like 644 or 0755)");
    int v = stoi(oct, nullptr, 8);

    fs::perms set = fs::perms::none;
    auto add = [&](int bit, fs::perms perm){ if (v & bit) set |= perm; };

    // owner
    add(0400, fs::perms::owner_read);
    add(0200, fs::perms::owner_write);
    add(0100, fs::perms::owner_exec);
    // group
    add(0040, fs::perms::group_read);
    add(0020, fs::perms::group_write);
    add(0010, fs::perms::group_exec);
    // others
    add(0004, fs::perms::others_read);
    add(0002, fs::perms::others_write);
    add(0001, fs::perms::others_exec);

    fs::permissions(p, set, fs::perm_options::replace);
}

// ----- listing -----
void listDirectory(const fs::path &path, bool longMode=false) {
    cout << "\nContents of " << path << ":" << endl;
    std::error_code ec;
    for (const auto &entry : fs::directory_iterator(path, ec)) {
        if (ec) { cerr << "Error: " << ec.message() << endl; break; }

        bool isDir = entry.is_directory(ec);
        if (!longMode) {
            cout << (isDir ? "[DIR] " : "[FILE] ")

```

```

        << entry.path().filename().string() << '\n';
    continue;
}

auto statPerms = fs::status(entry.path(), ec).permissions();
auto perms = permString(statPerms);

uintmax_t sz = 0;
if (!isDir) {
    std::error_code szec;
    sz = fs::file_size(entry.path(), szec);
    if (szec) sz = 0;
}

auto lwt_ec = std::error_code{};
auto ftime = fs::last_write_time(entry.path(), lwt_ec);
std::time_t ctt = lwt_ec ? std::time_t{} : to_time_t(ftime);
std::tm* tm_ptr = lwt_ec ? nullptr : std::localtime(&ctt);
string tstr = timeString(tm_ptr);

cout << (isDir ? 'd' : '-') << perms << " "
<< setw(10) << sz << " "
<< tstr << " "
<< entry.path().filename().string() << '\n';
}
}

// ----- file ops -----
void createFile(const fs::path &path) {
    ofstream file(path);
    if (file) cout << "File created: " << path.filename().string() << '\n';
    else cout << "Error creating file!\n";
}

void deleteFile(const fs::path &path) {
    if (fs::exists(path)) {
        fs::remove_all(path);
        cout << "Deleted: " << path.filename().string() << '\n';
    } else {
        cout << "File or directory not found!\n";
    }
}

void copyFile(const fs::path &src, const fs::path &dest) {
    try {
        fs::path realDest = dest;
        if (fs::exists(dest) && fs::is_directory(dest)) {
            realDest = dest / src.filename();
        }
        fs::copy(src, realDest, fs::copy_options::overwrite_existing |
fs::copy_options::recursive);
        cout << "Copied " << src.filename().string() << " to " << realDest.string() <<
'\n';
    } catch (const exception &e) {

```

```

        cout << "Error copying file: " << e.what() << '\n';
    }
}

void moveFile(const fs::path &src, const fs::path &dest) {
    try {
        fs::path realDest = dest;
        if (fs::exists(dest) && fs::is_directory(dest)) {
            realDest = dest / src.filename();
        }
        fs::rename(src, realDest);
        cout << "Moved " << src.filename().string() << " to " << realDest.string() <<
    '\n';
    } catch (const exception &e) {
        cout << "Error moving file: " << e.what() << '\n';
    }
}

void searchFile(const fs::path &path, const string &filename) {
    cout << "Searching for \" " << filename << " \" in " << path << "... \n";
    bool found = false;
    std::error_code ec;
    for (const auto &entry : fs::recursive_directory_iterator(path, ec)) {
        if (ec) { cerr << "Error: " << ec.message() << "\n"; break; }
        if (entry.path().filename() == filename) {
            cout << "Found at: " << entry.path() << '\n';
            found = true;
        }
    }
    if (!found) cout << "No file named \" " << filename << " \" found.\n";
}

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") {
            string rest;
            getline(cin, rest);
            bool longMode = (rest.find("-l") != string::npos);
            listDirectory(currentPath, longMode);
        }
        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, mode; cin >> filename >> mode;
        try {
            chmodOctal(currentPath / filename, mode);
            cout << "Permissions updated for: " << filename << '\n';
        } catch (const exception& e) {
            cout << "chmod error: " << e.what() << '\n';
        }
    }
    else if (command == "help") {
        cout << "\nAvailable commands:\n"
            << "ls"                                - List files and directories\n"
            << "ls -l"                             - Long list (perms/size/time)\n"
            << "cd <dir>"                         - Change directory\n"
            << "create <file>"                     - Create a file\n"
            << "delete <file/dir>"                  - Delete file or directory (recursive)\n"
            << "copy <src> <dest>"                 - Copy file/folder (dest may be a directory)\n"
            << "move <src> <dest>"                  - Move or rename (dest may be a directory)\n"
            << "search <filename>"                   - Search file recursively by name\n"
            << "chmod <file> <mode>"                - Change permission (octal, e.g., 644, 755)\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;

```

```
}
```

7. Output Screenshots

- Program Start

```
premsagar@PremSagar:~$ ls
2241016021 2241016418 2241016420 C DOS_2241004161 DOSass4 Rajat a.out a.txt capstone check_char.sh q1.c q2.c q2i.c q2j.c snap
premsagar@PremSagar:~$ cd capstone
premsagar@PremSagar:~/capstone$ ls
A1-file-explorer
premsagar@PremSagar:~/capstone$ cd A1-file-explorer
premsagar@PremSagar:~/capstone/A1-file-explorer$ ls
CMakeLists.txt build filex main.cpp prem.txt
premsagar@PremSagar:~/capstone/A1-file-explorer$ gedit main.cpp

(gedit:1239): IBUS-WARNING **: 19:13:00.970: Failed to mkdir /home/premsagar/snap/gedit/697/.config/ibus/bus: Not a directory
premsagar@PremSagar:~/capstone/A1-file-explorer$ g++ -std=gnu++17 -O2 -Wall -Wextra -pedantic main.cpp -o filex
premsagar@PremSagar:~/capstone/A1-file-explorer$ ./filex
```

- Listing Files (ls)

```
premsagar@PremSagar:~/capstone/A1-file-explorer$ ./filex
===== Simple File Explorer (C++ - LinuxOS) =====
Type 'help' to view available commands.

[~/home/premsagar/capstone/A1-file-explorer]"$ ls

Contents of "/home/premsagar/capstone/A1-file-explorer":
[FILE] filex
[DIR] .git
[DIR] build
[FILE] main.cpp
[DIR] .vscode
[FILE] prem.txt
[FILE] CMakeLists.txt
```

- Long Listing (ls -l)

```
[~/home/premsagar/capstone/A1-file-explorer]"$ ls -l

Contents of "/home/premsagar/capstone/A1-file-explorer":
-rwxr-xr-x    67640  2025-11-08 19:13  filex
drwxr-xr-x     0   2025-11-08 19:05  .git
drwxr-xr-x     0   2025-11-08 14:33  build
-rw-r--r--    8560  2025-11-08 18:48  main.cpp
drwxr-xr-x     0   2025-11-08 18:23  .vscode
-rw-r--r--     0   2025-11-08 18:49  prem.txt
-rw-r--r--     333  2025-11-08 18:21  CMakeLists.txt
```

- Creating File (create demo.txt)

```
[~/home/premsagar/capstone/A1-file-explorer]"$ create demo.txt
File created: demo.txt
```

- **Searching File (search demo.txt)**

```
["/home/premsagar/capstone/A1-file-explorer"]$ search demo.txt
Searching for "demo.txt" in "/home/premsagar/capstone/A1-file-explorer"...
Found at: "/home/premsagar/capstone/A1-file-explorer/demo.txt"
```

- **Changing Permissions (chmod demo.txt 644)**

```
["/home/premsagar/capstone/A1-file-explorer"]$ chmod demo.txt 644
Permissions updated for: demo.txt
```

- **Copying File (copy demo.txt copy_demo.txt)**

```
["/home/premsagar/capstone/A1-file-explorer"]$ copy demo.txt copy_demo.txt
Copied demo.txt to /home/premsagar/capstone/A1-file-explorer/copy_demo.txt
```

- **Moving File (move copy.txt testfolder/)**

```
["/home/premsagar/capstone/A1-file-explorer"]$ move copy_demo.txt movetest
Moved copy_demo.txt to /home/premsagar/capstone/A1-file-explorer/movetest/copy_demo.txt
```

- **Deleting File (delete demo.txt)**

```
["/home/premsagar/capstone/A1-file-explorer"]$ delete demo.txt
Deleted: demo.txt
```

8. Conclusion

This project successfully demonstrates the development of a **Linux-based File Explorer** using C++17.

It enabled hands-on practice with:

- Filesystem traversal
- Recursive search
- Permission handling
- Command-based user interface

The project fulfills all requirements of the **A1 Capstone Module** and builds strong fundamentals in system-level programming.