File Explorer Application (C++, Linux Terminal-Based)

Description

A console-based file explorer is a command-line program written in C that lets users manage their files and directories without a graphical interface. It directly interacts with the Linux operating system APIs to provide essential file operations, navigation, and permissions management – all within the terminal.

Objective

- Main Goal:
  Build a terminal application in C that can list, search, and manipulate files and directories, offering a user-friendly console alternative to graphical file browsers.

Core Features

- File Listing:
  Display contents (files and directories) of the current working directory.
- Directory Navigation:
  Move into subdirectories, go back to parent, or jump to specific paths.
- File Operations:
  - Copy: Duplicate files or folders.
  - Move: Relocate items to other directories.
  - Delete: Remove files or directories with safety checks.
  - Create: Make new files or directories.
- File Search:
  Search for files or folders by name or attribute.
- File Permissions Management:
  View and modify read, write, execute permissions for files/directories.

Workflow / Implementation Steps

Step	Details
Design Structure	Plan core commands, user interface, error handling, and modular code structure
Set Up Coding Env	Install Linux (or use WSL on Windows), gcc compiler, text editor (VSCode, Vim, etc.)
Directory Listing	Use Linux system calls (opendir, readdir, stat) to display contents
Navigation	Allow users to change current directory (chdir, path management)
File Operations	Implement actions for copying (cp logic), moving, deleting (remove), and creating files
Search	Develop search by name or pattern within directories
Permission Ctrl	Check and modify file/directory permissions (chmod, stat)
Testing	Run the explorer, handle errors, improve usability

Code:

terminal_explorer.cpp

```cpp
#include <iostream>
#include <filesystem>
#include <vector>
#include <string>
#include <iomanip>
#include <cstring>
#include <fstream>
#include <algorithm>
#include <chrono>
#include <ctime>

#ifdef _WIN32
#include <windows.h>
#include <lmcons.h>
#else
#include <unistd.h>
#include <sys/stat.h>
#=include <sys/types.h>
#include <pwd.h>
#include <grp.h>
#endif

namespace fs = std::filesystem;

class FileExplorer {
private:
    fs::path currentPath;
    std::vector<fs::directory_entry> currentEntries;

    void listDirectory() {
        currentEntries.clear();
        for (const auto& entry : fs::directory_iterator(currentPath)) {
```

```
130         }
131     }
132
133     void createDirectory(const std::string& name) {
134         try {
135             fs::create_directory(currentPath / name);
136             listDirectory();
137         } catch (const std::exception& e) {
138             std::cerr << "Error creating directory: " << e.what() << std::endl;
139         }
140     }
141
142     void removeFile(const std::string& name) {
143         try {
144             fs::path target = currentPath / name;
145             if (fs::is_directory(target)) {
146                 fs::remove_all(target);
147             } else {
148                 fs::remove(target);
149             }
150             listDirectory();
151         } catch (const std::exception& e) {
152             std::cerr << "Error removing: " << e.what() << std::endl;
153         }
154     }
155
156     void copyFile(const std::string& src, const std::string& dest) {
157         try {
158             fs::path sourcePath = (src[0] == '/') ? src : currentPath / src;
159             fs::path destPath = (dest[0] == '/') ? dest : currentPath / dest;
160
161             if (fs::is_directory(sourcePath)) {
```

```
            if (fs::is_directory(sourcePath)) {
                fs::create_directories(destPath);
                fs::copy(sourcePath, destPath, fs::copy_options::recursive);
            } else {
                fs::copy(sourcePath, destPath, fs::copy_options::overwrite_existing);
            }
            listDirectory();
        } catch (const std::exception& e) {
            std::cerr << "Error copying: " << e.what() << std::endl;
        }
    }

    void moveFile(const std::string& src, const std::string& dest) {
        try {
            fs::path sourcePath = (src[0] == '/') ? src : currentPath / src;
            fs::path destPath = (dest[0] == '/') ? dest : currentPath / dest;

            fs::rename(sourcePath, destPath);
            listDirectory();
        } catch (const std::exception& e) {
            std::cerr << "Error moving: " << e.what() << std::endl;
        }
    }

    void search(const std::string& pattern) {
        std::cout << "Searching for: " << pattern << " in " << currentPath << "\n";
        size_t count = 0;

        for (const auto& entry : fs::recursive_directory_iterator(currentPath)) {
            std::string filename = entry.path().filename().string();
            if (filename.find(pattern) != std::string::npos) {
                std::cout << entry.path().string() << std::endl;
```

```
                count++;
            }
        }

        std::cout << "\nFound " << count << " matches.\n";
    }
};

int main() {
    FileExplorer explorer;
    std::string command;

    while (true) {
        explorer.list();

        std::cout << "\n> ";
        std::getline(std::cin, command);

        if (command == "exit") {
            break;
        } else if (command.substr(0, 3) == "cd ") {
            explorer.changeDirectory(command.substr(3));
        } else if (command.substr(0, 6) == "mkdir ") {
            explorer.createDirectory(command.substr(6));
        } else if (command.substr(0, 3) == "rm ") {
            explorer.removeFile(command.substr(3));
        } else if (command.substr(0, 3) == "cp ") {
            size_t space = command.find(' ', 3);
            if (space != std::string::npos) {
                std::string src = command.substr(3, space - 3);
                std::string dest = command.substr(space + 1);
                explorer.copyFile(src, dest);
```

```
        std::string color = is_directory(entry) ? "\033[1;34m" : "\033[0m";
        std::cout << " " << color << entry.path().filename().string() << "\033[0m" << std::endl;
    }

public:
    FileExplorer() {
        currentPath = fs::current_path();
        listDirectory();
    }

    void list() {
#ifdef _WIN32
        system("cls");
#else
        system("clear");
#endif
        std::cout << "Current directory: " << currentPath << "\n\n";

        for (const auto& entry : currentEntries) {
            printFileInfo(entry);
        }

        std::cout << "\nCommands: cd <dir>, mkdir <name>, rm <name>, cp <src> <dest>, mv <src> <dest>
    }

    void changeDirectory(const std::string& path) {
        fs::path newPath = currentPath / path;
        if (fs::exists(newPath) && fs::is_directory(newPath)) {
            currentPath = fs::canonical(newPath);
            listDirectory();
        } else {
            std::cerr << "Directory not found: " << path << std::endl;
```

```
        std::string color = is_directory(entry) ? "\033[1;34m" : "\033[0m";
        std::cout << " " << color << entry.path().filename().string() << "\033[0m" << std::endl;
    }

public:
    FileExplorer() {
        currentPath = fs::current_path();
        listDirectory();
    }

    void list() {
#ifdef _WIN32
        system("cls");
#else
        system("clear");
#endif
        std::cout << "Current directory: " << currentPath << "\n\n";

        for (const auto& entry : currentEntries) {
            printFileInfo(entry);
        }

        std::cout << "\nCommands: cd <dir>, mkdir <name>, rm <name>, cp <src> <dest>, mv <src> <dest>
    }

    void changeDirectory(const std::string& path) {
        fs::path newPath = currentPath / path;
        if (fs::exists(newPath) && fs::is_directory(newPath)) {
            currentPath = fs::canonical(newPath);
            listDirectory();
        } else {
            std::cerr << "Directory not found: " << path << std::endl;
```

```
        } else if (command.substr(0, 3) == "mv ") {
            size_t space = command.find(' ', 3);
            if (space != std::string::npos) {
                std::string src = command.substr(3, space - 3);
                std::string dest = command.substr(space + 1);
                explorer.moveFile(src, dest);
            }
        } else if (command.substr(0, 7) == "search ") {
            explorer.search(command.substr(7));
            std::cout << "Press Enter to continue...";
            std::cin.ignore();
        }
    }

    return 0;
}

```

Output

```
-rwxrwxrwx user     group     3912237 C-A-P-Y-B-A-R-A.mp3
-rwxrwxrwx user     group       93769 COA_LAB-4.pdf
-rwxrwxrwx user     group       93769 COA_LAB-5.pdf
-rwxrwxrwx user     group     5222586 CS345-Algorithms-II-Algorithm-Design-by-Jon-Kleinberg-Eva-Tardos.pdf
-rwxrwxrwx user     group     1668254 Campus Process.pdf
-rwxrwxrwx user     group     4920544 Can you handle pressure.mp4
-rwxrwxrwx user     group     4029741 Cat [Youtube_ Feline Music].mp3
-rwxrwxrwx user     group      517494 Ch-21.pdf
-rwxrwxrwx user     group      610981 Ch-22.pdf
-rwxrwxrwx user     group      941545 Ch-24.pdf
-rwxrwxrwx user     group      946040 Ch-25.pdf
-rwxrwxrwx user     group      567922 Ch-26.pdf
-rwxrwxrwx user     group     1387388 Chapter 2 - thinking like an economist.PDF
-rwxrwxrwx user     group         489 ChatGPT Image Apr 6, 2025, 09_45_24 AM.png
-rwxrwxrwx user     group     1104440 ChatGPT Installer.exe
-rwxrwxrwx user     group     1373744 ChromeSetup.exe
-rwxrwxrwx user     group      321600 Course Outcomes and lesson plan (PME) sem-2.pdf
-rwxrwxrwx user     group   119549160 CursorUserSetup-x64-1.5.7.exe
-rwxrwxrwx user     group    18038784 DB.Browser.for.SQLite-3.12.2-win64.msi
-rwxrwxrwx user     group     1832038 DBMS Assignment-output.pdf
-rwxrwxrwx user     group      334433 DSA Assignment-2-1 (1).pdf
-rwxrwxrwx user     group      334433 DSA Assignment-2-1 (2).pdf
-rwxrwxrwx user     group      334433 DSA Assignment-2-1.pdf
-rwxrwxrwx user     group           0 DT20245990226_Application_Form.pdf
-rwxrwxrwx user     group     9055819 DashingImage.jpg
-rwxrwxrwx user     group      732396 Default_a_scenary_of_mountains_and_rising_sun_reddish_yellowda_2_651de51f-3005-4366-be0a-e6c2e888156a_1 (1).jpg
-rwxrwxrwx user     group      732396 Default_a_scenary_of_mountains_and_rising_sun_reddish_yellowda_2_651de51f-3005-4366-be0a-e6c2e888156a_1 (2).jpg
-rwxrwxrwx user     group      732396 Default_a_scenary_of_mountains_and_rising_sun_reddish_yellowda_2_651de51f-3005-4366-be0a-e6c2e888156a_1.jpg
-rwxrwxrwx user     group        4734 Document (4) (3).pdf
-rwxrwxrwx user     group      407312 DreamShaper_v5_a_creapy_mummy_frontview_angle_highly_detaile_3.jpg
-rwxrwxrwx user     group      373140 DreamShaper_v5_a_warrior_face_strong_looking_straight_to_fron_2 (1).jpg
-rwxrwxrwx user     group      633548 DreamShaper_v5_a_warrior_face_strong_looking_straight_to_fron_2 (2).jpg
-rwxrwxrwx user     group      351287 DreamShaper_v5_a_warrior_face_strong_looking_straight_to_fron_2.jpg
-rwxrwxrwx user     group      462421 DreamShaper_v5_centered_isometric_mural_graffiti_composition_s_2.jpg
-rwxrwxrwx user     group      453987 DreamShaper_v5_centered_isometric_mural_graffiti_composition_s_3.jpg
-rwxrwxrwx user     group      391044 DreamShaper_v5_create_image_of_a_man_with_Indian_tradational_c_3.jpg
-rwxrwxrwx user     group      520620 DreamShaper_v5_dog_is_jumping_in_eyes_0.jpg
-rwxrwxrwx user     group      551484 DreamShaper_v5_dog_is_jumping_in_eyes_1.jpg
-rwxrwxrwx user     group      524674 DreamShaper_v5_make_the_selected_image_more_charismatic_1.jpg
-rwxrwxrwx user     group      564285 DreamShaper_v5_shah_rukh_khan_2.jpg
-rwxrwxrwx user     group      425074 DreamShaper_v5_shah_rukh_khan_and_priyanka_chopra_0.jpg
```

Conclusion

The File Explorer Application in C for the Linux terminal enables users to efficiently manage files and directories without relying on graphical tools. By implementing core commands like listing contents, navigation, file operations, search, and permission management, you develop a solid grasp of Linux system calls and low-level programming concepts. This project offers hands-on experience with OS-level tasks, making it a practical exercise for building foundational skills in systems programming.

Future Enhancements

- User Interface Improvements: Add support for colored output, tree-like directory views, and interactive menus to enhance user experience.
- File Preview Features: Integrate file type detection and previews for text and image files.
- Batch Operations: Enable bulk file copying, moving, and deletion via pattern matching or selection.
- Bookmarking & History: Implement directory bookmarking and history tracking for quick navigation.
- Remote Access: Extend file operations to networked or remote directories using protocols like SFTP or SSH.
- Multi-threading: Make certain file operations concurrent for better performance on large datasets.

References

- Linux Programming with C
- man pages for opendir, readdir, stat, chmod, unlink
- Official GNU C Library documentation
- TutorialsPoint: File Handling in C
- GeeksforGeeks: Directory Handling in C