#include <iostream>

#include <string>

#include <dirent.h>

#include <unistd.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <cstring>

#include <vector>

#include <fstream>

#include <sstream> // Include this for std::istringstream

class FileExplorer {

private:

std::string currentDirectory;

public:

FileExplorer() {

char buffer[1024];

if (getcwd(buffer, sizeof(buffer)) != nullptr) {

currentDirectory = std::string(buffer);

}

}

void start() {

std::string command;

while (true) {

std::cout << currentDirectory << " > ";

std::getline(std::cin, command);

if (command == "exit") break;

executeCommand(command);

}

}

private:

void executeCommand(const std::string& command) {

if (command == "list") {

listFiles();

} else if (command.substr(0, 3) == "cd ") {

changeDirectory(command.substr(3));

} else if (command.substr(0, 4) == "copy") {

auto args = parseCommand(command);

if (args.size() == 3) {

copyFile(args[1], args[2]);

}

} else if (command.substr(0, 4) == "move") {

auto args = parseCommand(command);

if (args.size() == 3) {

moveFile(args[1], args[2]);

}

} else if (command.substr(0, 6) == "delete") {

auto args = parseCommand(command);

if (args.size() == 2) {

deleteFile(args[1]);

}

} else if (command.substr(0, 6) == "create") {

auto args = parseCommand(command);

if (args.size() == 2) {

createFile(args[1]);

}

} else if (command.substr(0, 6) == "search") {

auto args = parseCommand(command);

if (args.size() == 2) {

searchFiles(currentDirectory, args[1]);

}

} else if (command.substr(0, 4) == "chmod") {

auto args = parseCommand(command);

if (args.size() == 3) {

setPermissions(args[1], args[2]);

}

} else {

std::cout << "Unknown command: " << command << std::endl;

}

}

void listFiles() {

DIR\* dir = opendir(currentDirectory.c\_str());

if (dir == nullptr) {

std::cerr << "Error opening directory!" << std::endl;

return;

}

struct dirent\* entry;

while ((entry = readdir(dir)) != nullptr) {

std::cout << entry->d\_name << std::endl;

}

closedir(dir);

}

void changeDirectory(const std::string& newDir) {

if (chdir(newDir.c\_str()) == 0) {

char buffer[1024];

if (getcwd(buffer, sizeof(buffer)) != nullptr) {

currentDirectory = std::string(buffer);

}

} else {

std::cerr << "Error: Cannot change directory to " << newDir << std::endl;

}

}

void copyFile(const std::string& source, const std::string& destination) {

std::ifstream src(source, std::ios::binary);

std::ofstream dest(destination, std::ios::binary);

dest << src.rdbuf();

src.close();

dest.close();

std::cout << "File copied to " << destination << std::endl;

}

void moveFile(const std::string& source, const std::string& destination) {

if (rename(source.c\_str(), destination.c\_str()) == 0) {

std::cout << "File moved to " << destination << std::endl;

} else {

std::cerr << "Error moving file!" << std::endl;

}

}

void deleteFile(const std::string& path) {

if (unlink(path.c\_str()) == 0) {

std::cout << "File deleted: " << path << std::endl;

} else {

std::cerr << "Error deleting file!" << std::endl;

}

}

void createFile(const std::string& fileName) {

std::ofstream file(fileName);

if (file) {

std::cout << "File created: " << fileName << std::endl;

} else {

std::cerr << "Error creating file!" << std::endl;

}

file.close();

}

void searchFiles(const std::string& directory, const std::string& pattern) {

DIR\* dir = opendir(directory.c\_str());

if (dir == nullptr) {

std::cerr << "Error opening directory!" << std::endl;

return;

}

struct dirent\* entry;

while ((entry = readdir(dir)) != nullptr) {

if (strstr(entry->d\_name, pattern.c\_str()) != nullptr) {

std::cout << "Found: " << entry->d\_name << std::endl;

}

if (entry->d\_type == DT\_DIR && strcmp(entry->d\_name, ".") != 0 && strcmp(entry->d\_name, "..") != 0) {

searchFiles(directory + "/" + entry->d\_name, pattern);

}

}

closedir(dir);

}

void setPermissions(const std::string& path, const std::string& mode) {

mode\_t perm = std::stoi(mode, 0, 8);

if (chmod(path.c\_str(), perm) == 0) {

std::cout << "Permissions set to " << mode << std::endl;

} else {

std::cerr << "Error setting permissions!" << std::endl;

}

}

std::vector<std::string> parseCommand(const std::string& command) {

std::vector<std::string> tokens;

std::string token;

std::istringstream tokenStream(command);

while (std::getline(tokenStream, token, ' ')) {

tokens.push\_back(token);

}

return tokens;

}

};

int main() {

FileExplorer explorer;

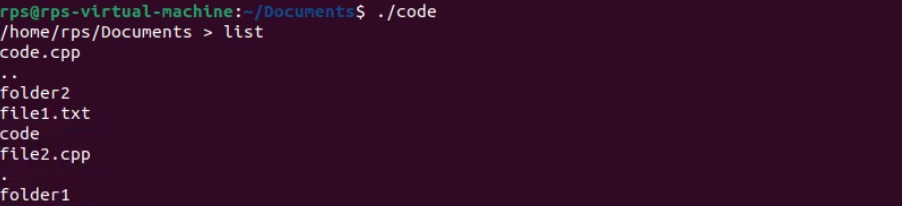
explorer.start();

return 0;

}

**Output:**

File name is code.cpp:

**Listing Files in the Current Directory**:

**Changing Directory:**



**Creating a New File:**



**Copying a File:**



**Moving (Renaming) a File:**



**Deleting a File:**



**Searching for a File:**



**Exiting the Program:**



**Conclusion:**

With this code, you can use simple commands that resemble Unix/Linux shell commands to communicate with your file system. This software can be tested locally by compiling and executing it. The files and directories in your working directory will determine the final product.