#include <iostream> // For input/output operations

#include <filesystem> // For filesystem operations (C++17)

#include <vector> // For storing file info

#include <string> // For string handling

namespace fs = std::filesystem;

// Function to list files based on user choice

void listFiles() {

std::cout << "\n=== List Files ===\n";

std::cout << "1. List all files\n";

std::cout << "2. List files by extension\n";

std::cout << "3. List files by name pattern\n";

std::cout << "Enter your choice (1-3): ";

int option;

std::cin >> option;

// Gather all files in current directory

std::vector<fs::directory\_entry> files;

for (const auto& entry : fs::directory\_iterator(fs::current\_path())) {

if (entry.is\_regular\_file()) {

files.push\_back(entry);

}

}

if (files.empty()) {

std::cout << "No files found in current directory.\n";

return;

}

switch (option) {

case 1: {

std::cout << "\nAll files:\n";

for (const auto& file : files) {

std::cout << " - " << file.path().filename().string() << "\n";

}

break;

}

case 2: {

std::cout << "Enter extension (e.g., .cpp): ";

std::string ext;

std::cin >> ext;

std::cout << "\nFiles with extension '" << ext << "':\n";

for (const auto& file : files) {

if (file.path().extension() == ext) {

std::cout << " - " << file.path().filename().string() << "\n";

}

}

break;

}

case 3: {

std::cout << "Enter filename pattern: ";

std::string pattern;

std::cin >> pattern;

std::cout << "\nFiles matching pattern '" << pattern << "':\n";

for (const auto& file : files) {

if (file.path().filename().string().find(pattern) != std::string::npos) {

std::cout << " - " << file.path().filename().string() << "\n";

}

}

break;

}

default:

std::cout << "Invalid option selected.\n";

break;

}

}

// Function to create a new directory

void createDirectory() {

std::cout << "Enter directory name to create: ";

std::string dirName;

std::cin >> dirName;

if (fs::create\_directory(dirName)) {

std::cout << "Directory '" << dirName << "' successfully created.\n";

} else {

std::cout << "Failed to create directory. It may already exist.\n";

}

}

// Function to change current working directory

void changeDirectory() {

std::cout << "\nChange Directory Options:\n";

std::cout << "1. Go to parent directory\n";

std::cout << "2. Go to root directory\n";

std::cout << "3. Enter specific path\n";

std::cout << "Select an option (1-3): ";

int option;

std::cin >> option;

switch (option) {

case 1: {

fs::current\_path("..");

std::cout << "Current directory: " << fs::current\_path() << "\n";

break;

}

case 2: {

fs::path rootPath = fs::current\_path().root\_path();

fs::current\_path(rootPath);

std::cout << "Now at root directory: " << fs::current\_path() << "\n";

break;

}

case 3: {

std::cout << "Enter the path: ";

std::string path;

std::cin >> path;

if (fs::exists(path) && fs::is\_directory(path)) {

fs::current\_path(path);

std::cout << "Changed directory to: " << fs::current\_path() << "\n";

} else {

std::cout << "Invalid path.\n";

}

break;

}

default:

std::cout << "Invalid option.\n";

break;

}

}

int main() {

while (true) {

std::cout << "\n=== File Management Menu ===\n";

std::cout << "1. List Files\n";

std::cout << "2. Create Directory\n";

std::cout << "3. Change Directory\n";

std::cout << "4. Exit\n";

std::cout << "Your choice (1-4): ";

int choice;

std::cin >> choice;

x

switch (choice) {

case 1:

listFiles();

break;

case 2:

createDirectory();

break;

case 3:

changeDirectory();

break;

case 4:

std::cout << "Exiting. Goodbye!\n";

return 0;

default:

std::cout << "Invalid choice. Please try again.\n";

}

}

}