Fahad M. Al -alawi

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#ifdef \_\_unix\_\_

#include <dirent.h>

#include <sys/stat.h>

#include <unistd.h>

#include <limits.h>

#elif \_WIN32

#include <windows.h>

#include <string>

#endif

void listFiles(const std::string& path, int option) {

std::vector<std::string> files;

#ifdef \_\_unix\_\_

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

if (!dir) {

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

return;

}

struct dirent\* entry;

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

if (entry->d\_name[0] != '.') files.push\_back(entry->d\_name);

}

closedir(dir);

#elif \_WIN32

WIN32\_FIND\_DATA data;

HANDLE hFind = FindFirstFile((path + "\\\*").c\_str(), &data);

if (hFind == INVALID\_HANDLE\_VALUE) {

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

return;

}

do {

if (data.cFileName[0] != '.') files.push\_back(data.cFileName);

} while (FindNextFile(hFind, &data));

FindClose(hFind);

#endif

for (const auto& file : files) {

if (option == 1 || (option == 2 && file.substr(file.find\_last\_of('.') + 1) == "txt") ||

(option == 3 && file.find("moha") != std::string::npos)) {

std::cout << file << std::endl;

}

}

}

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

#ifdef \_\_unix\_\_

if (mkdir(path.c\_str(), 0755) != 0) std::cerr << "Error creating directory: " << path << std::endl;

else std::cout << "Directory created: " << path << std::endl;

#elif \_WIN32

if (CreateDirectory(path.c\_str(), nullptr) || GetLastError() == ERROR\_ALREADY\_EXISTS)

std::cout << "Directory created: " << path << std::endl;

else

std::cerr << "Error creating directory: " << path << std::endl;

#endif

}

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

#ifdef \_\_unix\_\_

if (chdir(path.c\_str()) != 0) std::cerr << "Error changing directory to: " << path << std::endl;

else std::cout << "Changed working directory to: " << path << std::endl;

#elif \_WIN32

if (SetCurrentDirectory(path.c\_str())) std::cout << "Changed working directory to: " << path << std::endl;

else std::cerr << "Error changing directory to: " << path << std::endl;

#endif

}

std::string getCurrentDirectory() {

#ifdef \_\_unix\_\_

char cwd[PATH\_MAX];

return getcwd(cwd, sizeof(cwd)) ? std::string(cwd) : "";

#elif \_WIN32

char cwd[MAX\_PATH];

return GetCurrentDirectory(MAX\_PATH, cwd) ? std::string(cwd) : "";

#endif

}

void displayMenu() {

std::cout << "MAIN MENU\n";

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

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

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

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

}

void handleListFiles() {

std::string path;

int option;

std::cout << "Enter the directory path to list files: ";

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

std::cout << "LIST FILE DETAIL\n";

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

std::cout << "2. List of Extension Files\n";

std::cout << "3. List of Name Pattern Files\n";

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

std::cin >> option;

std::cin.ignore();

listFiles(path, option);

}

void handleChangeDirectory() {

int option;

std::string path, currentPath = getCurrentDirectory();

std::cout << "Current Directory: " << currentPath << std::endl;

std::cout << "Change Directory Menu\n";

std::cout << "1. Step by Step Backward\n";

std::cout << "2. Goto Root Directory\n";

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

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

std::cin >> option;

std::cin.ignore();

switch (option) {

case 1: {

#ifdef \_\_unix\_\_

path = currentPath.substr(0, currentPath.find\_last\_of('/'));

#elif \_WIN32

path = currentPath.substr(0, currentPath.find\_last\_of('\\'));

#endif

break;

}

case 2:

#ifdef \_\_unix\_\_

path = "/";

#elif \_WIN32

path = "C:\\";

#endif

break;

case 3:

std::cout << "Please enter the Directory Name: ";

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

#ifdef \_\_unix\_\_

if (path[0] != '/') path = currentPath + "/" + path;

#elif \_WIN32

if (path[1] != ':') path = currentPath + "\\" + path;

#endif

break;

default:

std::cerr << "Invalid option!" << std::endl;

return;

}

changeDirectory(path);

}

int main() {

int option;

do {

displayMenu();

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

std::cin >> option;

std::cin.ignore();

switch (option) {

case 1:

handleListFiles();

break;

case 2: {

std::string path;

std::cout << "Enter the directory path to create: ";

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

createDirectory(path);

break;

}

case 3:

handleChangeDirectory();

break;

case 4:

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

break;

default:

std::cerr << "Invalid option! Please try again.\n";

break;

}

std::cout << "Press any key to continue...";

std::cin.get();

} while (option != 4);

return 0;

}