**实验报告：操作系统文件管理**

**实验目的**

本实验旨在实现一个简单的文件管理系统，通过该系统，用户可以执行文件和目录的创建、删除、查找、移动、复制、列出目录内容、改变当前目录等操作。

**实验环境**

- 编程语言：C

- 操作系统：Windows

**系统功能**

**1. 文件控制块结构体 (FCB)**

- 定义了文件控制块结构体，用于存储文件或目录的基本信息，包括名称、路径和是否为目录。

typedef struct {

char name[MAX\_NAME\_LEN];

char path[MAX\_PATH\_LEN]; // 文件路径

int is\_directory;

} FCB;

**2. 全局变量**

- current\_path：存储当前工作目录路径。

- filesystem：模拟文件系统的数组，用于存储文件控制块。

- file\_count：记录文件系统中的文件数量。

char current\_path[MAX\_PATH\_LEN];

FCB filesystem[MAX\_FILES];

int file\_count = 0;

**3. 系统初始化**

- 初始化文件系统，设置当前工作目录。

void initialize() {

file\_count = 0;

GetCurrentDirectory(MAX\_PATH\_LEN, current\_path);

}

**4. 创建文件/目录**

- 根据用户输入创建文件或目录，并更新文件系统。使用 CreateDirectory 创建目录，使用 CreateFile 创建文件。

void create\_file(char name, int is\_directory) {

char full\_path[MAX\_PATH\_LEN];

sprintf(full\_path, "%s\\%s", current\_path, name);

if (is\_directory) {

CreateDirectory(full\_path, NULL);

} else {

HANDLE hFile = CreateFile(full\_path, GENERIC\_WRITE, 0, NULL, CREATE\_NEW, FILE\_ATTRIBUTE\_NORMAL, NULL);

CloseHandle(hFile);

}

}

**5. 删除文件/目录**

- 删除指定名称的文件或目录。使用 RemoveDirectory 删除目录，使用 DeleteFile 删除文件。

void delete\_file(char name) {

char full\_path[MAX\_PATH\_LEN];

sprintf(full\_path, "%s\\%s", current\_path, name);

RemoveDirectory(full\_path) || DeleteFile(full\_path);

}

**6. 查找文件**

- 在当前目录下查找指定名称的文件，并返回其路径。通过 FindFirstFile 和 FindNextFile 进行文件查找。

FCB find\_file(char name) {

char search\_path[MAX\_PATH\_LEN];

sprintf(search\_path, "%s\\", current\_path);

WIN32\_FIND\_DATA findData;

HANDLE hFind = FindFirstFile(search\_path, &findData);

// 查找逻辑省略

}

**7. 移动文件/目录**

- 将文件或目录移动到指定位置。使用 MoveFile 函数进行文件或目录的移动操作。

void move\_file\_or\_directory(char source\_name, char dest\_name) {

char source\_path[MAX\_PATH\_LEN], dest\_path[MAX\_PATH\_LEN];

sprintf(source\_path, "%s\\%s", current\_path, source\_name);

sprintf(dest\_path, "%s\\%s\\%s", current\_path, dest\_name, source\_name);

MoveFile(source\_path, dest\_path);

}

**8. 列出目录内容**

- 列出当前目录下的所有文件和子目录。使用 FindFirstFile 和 FindNextFile 列出目录内容。

void list\_directory\_contents() {

char search\_path[MAX\_PATH\_LEN];

sprintf(search\_path, "%s\\", current\_path);

WIN32\_FIND\_DATA findData;

HANDLE hFind = FindFirstFile(search\_path, &findData);

// 列出逻辑省略

}

**9. 复制目录**

- 递归复制目录及其内容。通过 CreateDirectory 创建目标目录，并使用 CopyFile 复制文件。

void copy\_directory(char source\_name, char dest\_name) {

char source\_path[MAX\_PATH\_LEN], dest\_path[MAX\_PATH\_LEN];

sprintf(source\_path, "%s\\%s", current\_path, source\_name);

sprintf(dest\_path, "%s\\%s", current\_path, dest\_name);

CreateDirectory(dest\_path, NULL);

// 递归复制逻辑省略

}

**10. 改变当前目录**

- 切换到指定的子目录。使用 SetCurrentDirectory 切换目录，并更新 current\_path。

void change\_dir(char path) {

char new\_path[MAX\_PATH\_LEN];

sprintf(new\_path, "%s\\%s", current\_path, path);

SetCurrentDirectory(new\_path);

GetCurrentDirectory(MAX\_PATH\_LEN, current\_path);

}

**11. 显示当前路径**

- 显示当前工作目录的路径。使用 GetCurrentDirectory 获取当前路径并打印。

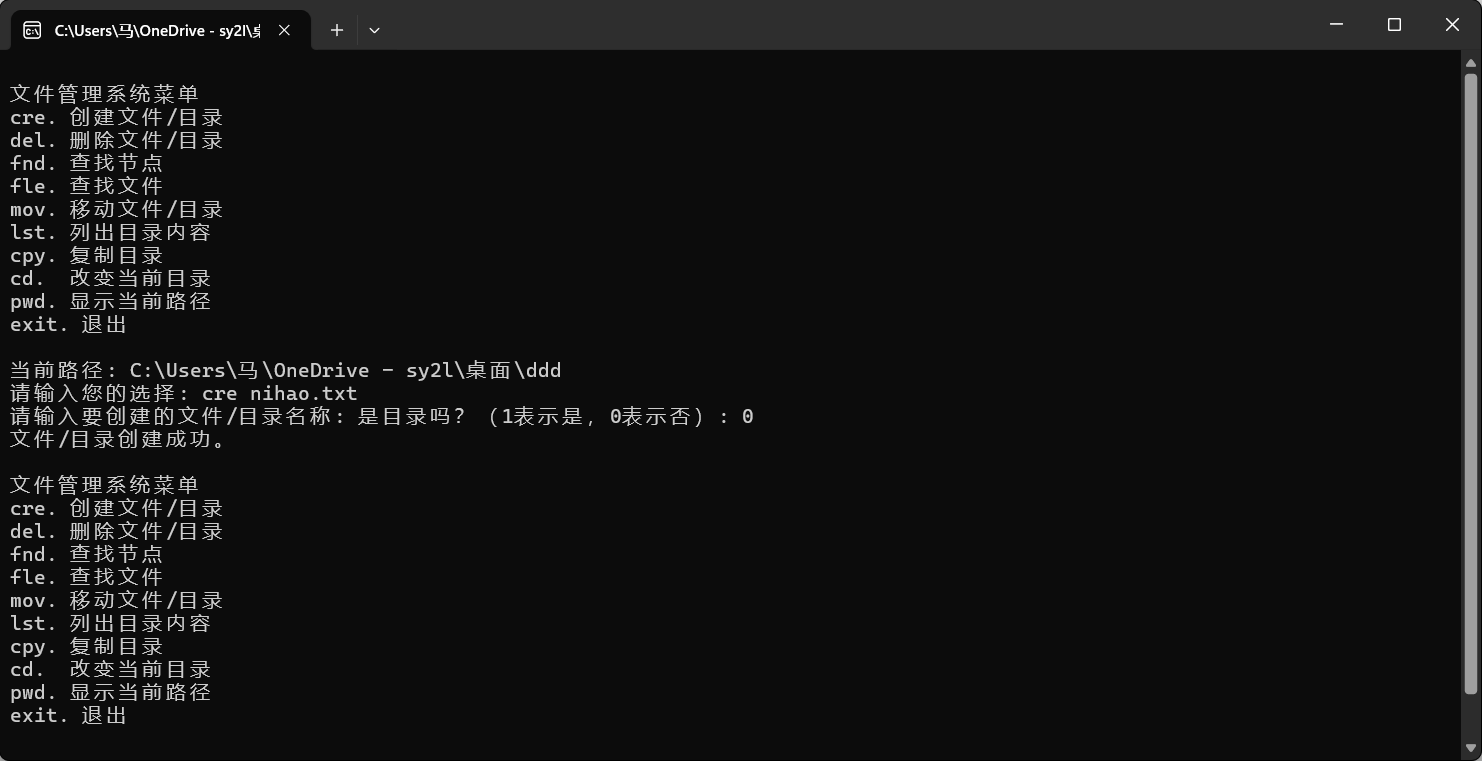
void display\_current\_path() {

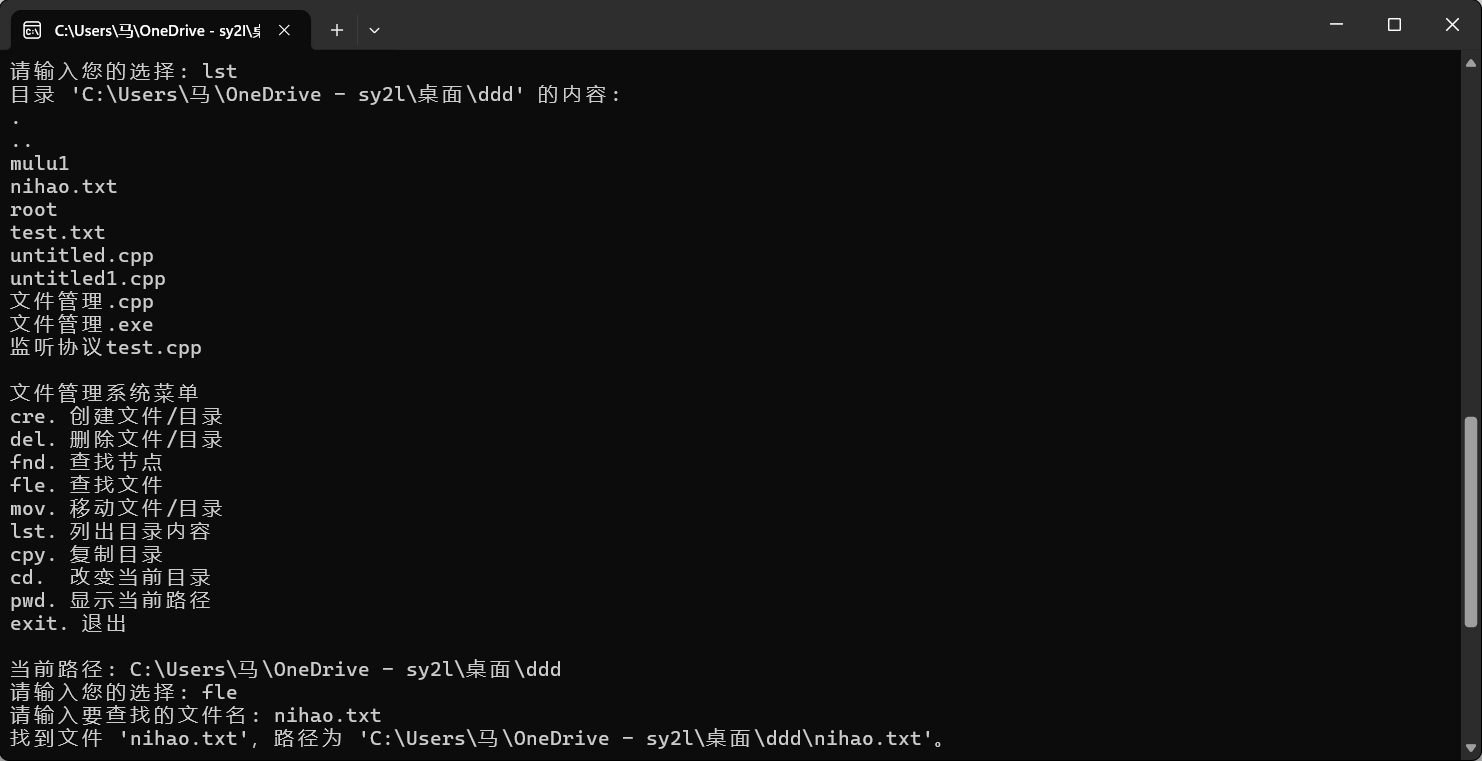
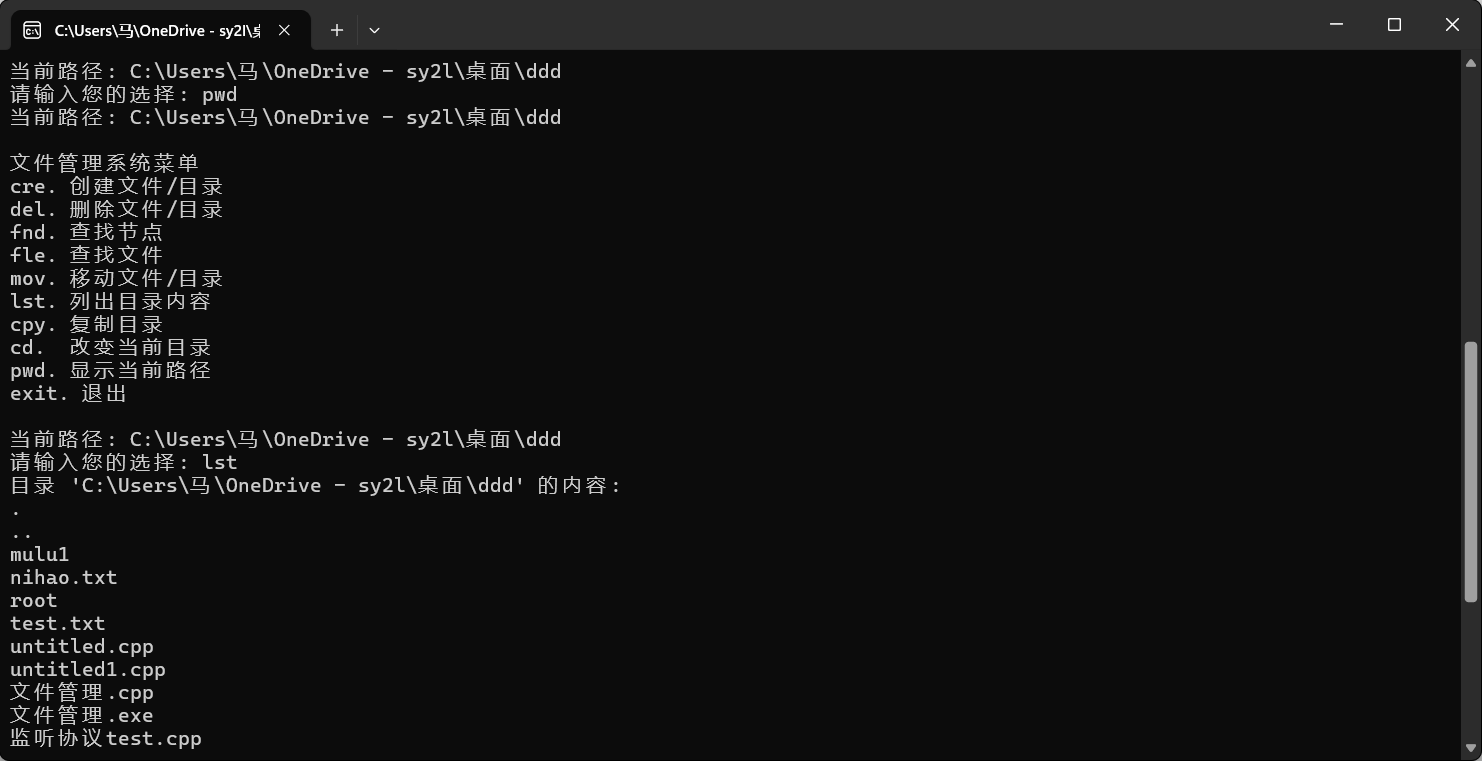
GetCurrentDirectory(MAX\_PATH\_LEN, current\_path);

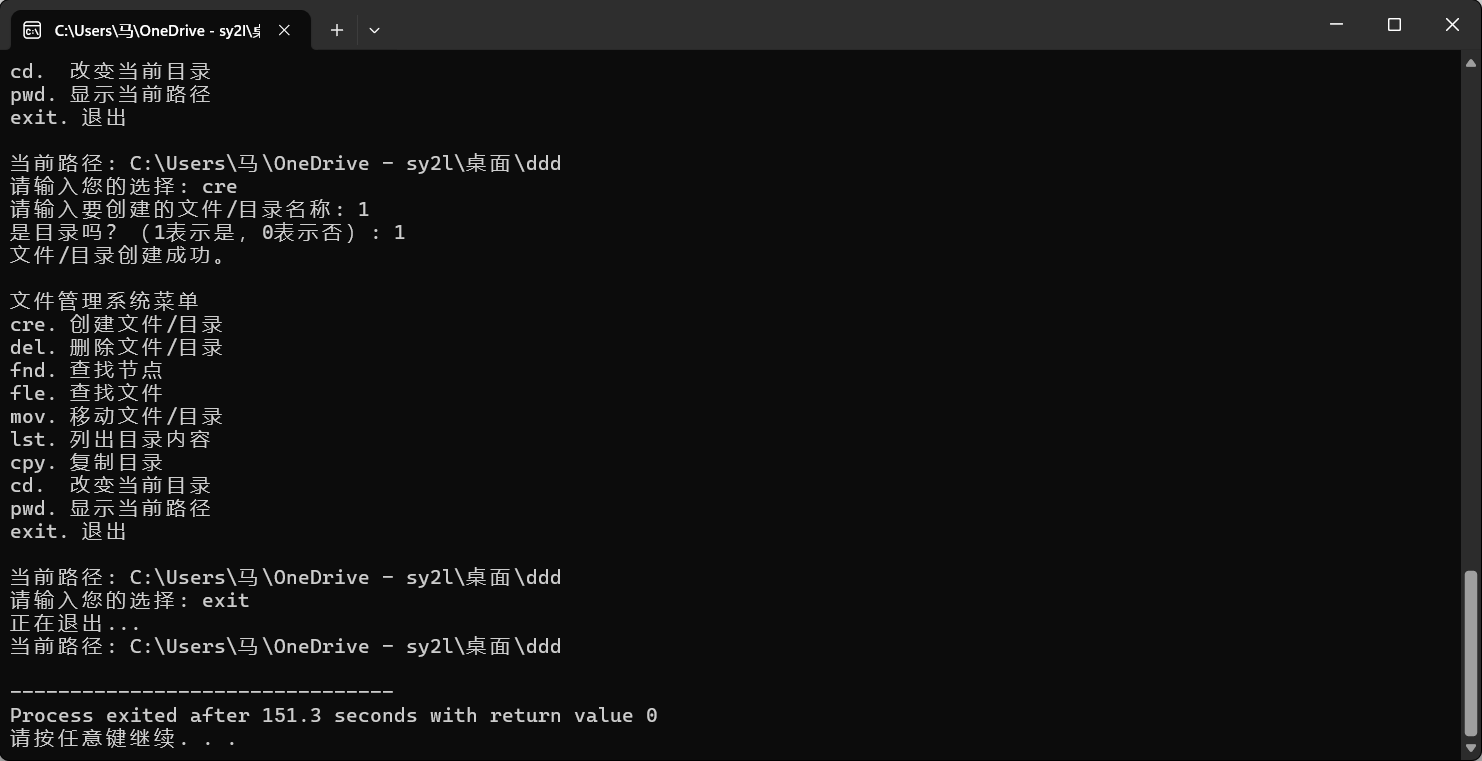
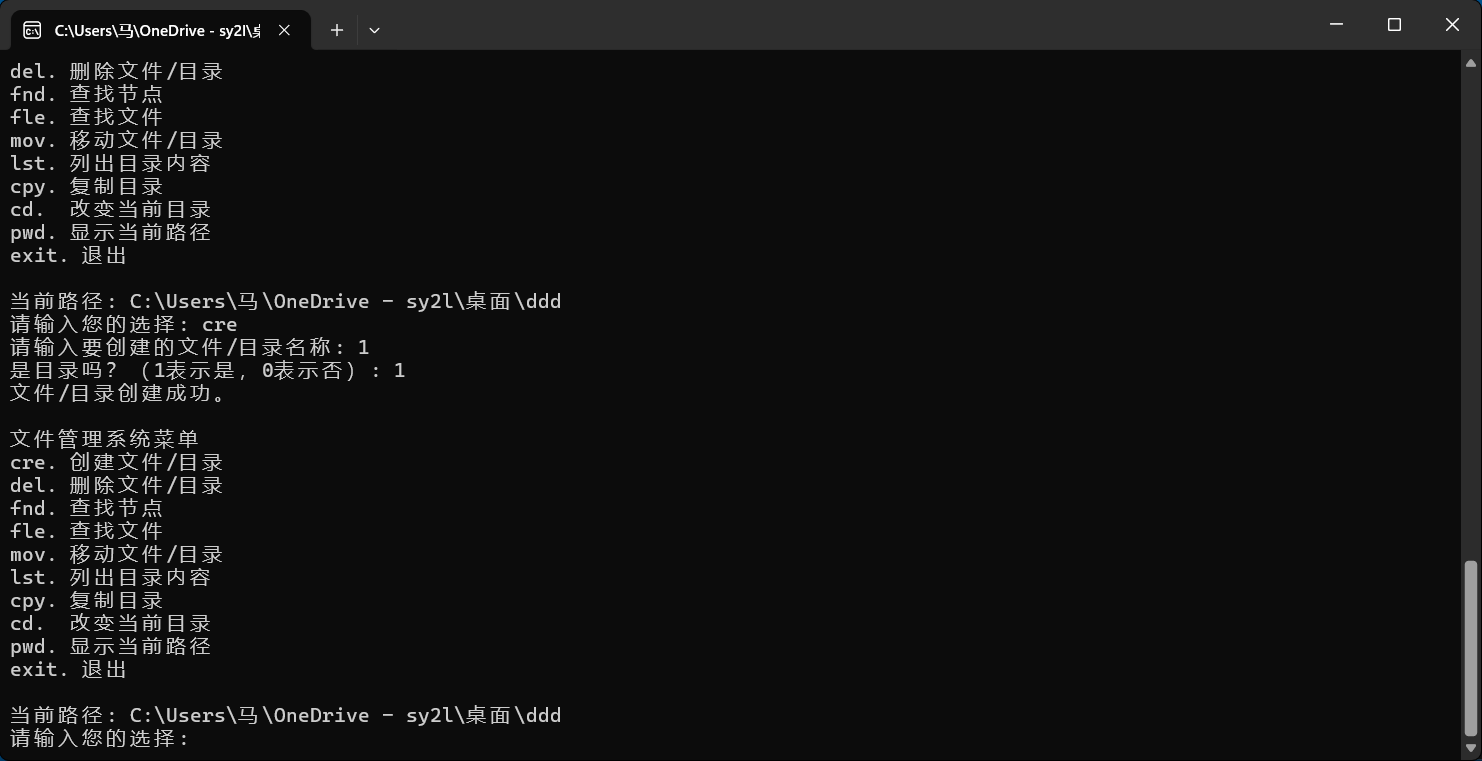
printf("当前路径: %s\n", current\_path);

}

**实验部分过程截图**







**实验结果**

通过上述函数的实现和调用，我们成功地创建了一个基本的文件管理系统，用户可以通过菜单选项执行不同的文件管理操作，如创建、删除、查找、移动、复制文件和目录，列出目录内容，以及改变当前目录等。本实验通过实现一个简单的文件管理系统，理解了文件系统的基本操作和实现方式。尽管该系统较为简单，但为进一步深入学习文件系统提供了良好的基础。

**源代码**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#define MAX\_PATH\_LEN 100

#define MAX\_NAME\_LEN 50

// 文件控制块结构体

typedef struct {

char name[MAX\_NAME\_LEN];

char path[MAX\_PATH\_LEN]; // 文件路径

int is\_directory;

// 可以添加其他需要的信息

} FCB;

// 全局变量

char current\_path[MAX\_PATH\_LEN]; // 当前工作目录路径，默认为空

// 函数声明

void initialize();

void create\_file(char \*name, int is\_directory);

void delete\_file(char \*name);

FCB\* find\_node(char \*name);

FCB\* find\_file(char \*name);

void move\_file\_or\_directory(char \*source\_name, char \*dest\_name);

void list\_directory\_contents();

void copy\_directory(char \*source\_name, char \*dest\_name);

void change\_dir(char \*path);

void go\_back();

void display\_current\_path();

// 示例的简单文件系统数据结构

#define MAX\_FILES 100

FCB filesystem[MAX\_FILES];

int file\_count = 0;

int main() {

int choice;

char name[MAX\_NAME\_LEN];

initialize(); // 初始化文件系统

FCB \*file; // 在此处声明变量

char option[5];

do {

printf("\n文件管理系统菜单\n");

printf("cre. 创建文件/目录\n");

printf("del. 删除文件/目录\n");

printf("fnd. 查找节点\n");

printf("fle. 查找文件\n");

printf("mov. 移动文件/目录\n");

printf("lst. 列出目录内容\n");

printf("cpy. 复制目录\n");

printf("cd. 改变当前目录\n");

printf("pwd. 显示当前路径\n");

printf("exit. 退出\n");

printf("\n");

display\_current\_path(); // 显示当前路径

printf("请输入您的选择: ");

scanf("%s", option);

if (strcmp(option, "cre") == 0) {

printf("请输入要创建的文件/目录名称: ");

scanf("%s", name);

printf("是目录吗？（1表示是，0表示否）: ");

int is\_directory;

scanf("%d", &is\_directory);

create\_file(name, is\_directory);

} else if (strcmp(option, "del") == 0) {

printf("请输入要删除的文件/目录名称: ");

scanf("%s", name);

delete\_file(name);

} else if (strcmp(option, "fnd") == 0) {

printf("请输入要查找的节点名称: ");

scanf("%s", name);

if (find\_node(name) != NULL)

printf("找到节点。\n");

else

printf("未找到节点。\n");

} else if (strcmp(option, "fle") == 0) {

printf("请输入要查找的文件名: ");

scanf("%s", name);

file = find\_file(name);

if (file != NULL) {

printf("找到文件 '%s'，路径为 '%s'。\n", file->name, file->path);

} else {

printf("未找到文件。\n");

}

} else if (strcmp(option, "mov") == 0) {

printf("请输入源文件/目录名称: ");

scanf("%s", name);

printf("请输入目标文件/目录名称: ");

char dest\_name[MAX\_NAME\_LEN];

scanf("%s", dest\_name);

move\_file\_or\_directory(name, dest\_name);

} else if (strcmp(option, "lst") == 0) {

list\_directory\_contents();

} else if (strcmp(option, "cpy") == 0) {

printf("请输入源文件/目录名称: ");

scanf("%s", name);

printf("请输入目标文件/目录名称: ");

scanf("%s", name); // 注意这里的输入，应该是 dest\_name

copy\_directory(name, name);

} else if (strcmp(option, "cd") == 0) {

printf("请输入要切换到的目录路径（相对于当前路径）: ");

scanf("%s", name);

change\_dir(name);

} else if (strcmp(option, "pwd") == 0) {

display\_current\_path();

} else if (strcmp(option, "exit") == 0) {

printf("正在退出...\n");

} else {

printf("无效选择，请重新输入。\n");

}

} while (strcmp(option, "exit") != 0);

display\_current\_path(); // 最后显示当前路径

return 0;

}

void initialize() {

// 初始化文件系统，这里简化为清空文件系统信息

file\_count = 0;

GetCurrentDirectory(MAX\_PATH\_LEN, current\_path); // 获取当前工作目录

}

void create\_file(char \*name, int is\_directory) {

// 创建文件或目录的函数实现

char full\_path[MAX\_PATH\_LEN];

sprintf(full\_path, "%s\\%s", current\_path, name);

if (is\_directory) {

if (!CreateDirectory(full\_path, NULL)) {

printf("创建目录失败。\n");

return;

}

} else {

HANDLE hFile = CreateFile(full\_path, GENERIC\_WRITE, 0, NULL, CREATE\_NEW, FILE\_ATTRIBUTE\_NORMAL, NULL);

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

printf("创建文件失败。\n");

return;

}

CloseHandle(hFile);

}

printf("文件/目录创建成功。\n");

}

void delete\_file(char \*name) {

// 删除文件或目录的函数实现

char full\_path[MAX\_PATH\_LEN];

sprintf(full\_path, "%s\\%s", current\_path, name);

if (RemoveDirectory(full\_path) || DeleteFile(full\_path)) {

printf("文件/目录删除成功。\n");

} else {

printf("删除失败或文件/目录不存在。\n");

}

}

FCB\* find\_node(char \*name) {

// 在文件系统中查找特定名称的节点（文件或目录）

char full\_path[MAX\_PATH\_LEN];

sprintf(full\_path, "%s\\%s", current\_path, name);

WIN32\_FIND\_DATA findData;

HANDLE hFind = FindFirstFile(full\_path, &findData);

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

FindClose(hFind);

return &filesystem[0]; // 返回一个虚拟的 FCB 指针，仅为示例

}

return NULL;

}

FCB\* find\_file(char \*name) {

// 在文件系统中查找特定名称的文件，并输出其路径

char search\_path[MAX\_PATH\_LEN];

sprintf(search\_path, "%s\\\*", current\_path);

WIN32\_FIND\_DATA findData;

HANDLE hFind = FindFirstFile(search\_path, &findData);

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

printf("无法查找文件。\n");

return NULL;

}

static FCB found\_file; // 静态变量存储找到的文件信息

do {

if (!(findData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) &&

strcmp(findData.cFileName, name) == 0) {

// 找到文件，输出路径

strcpy(found\_file.name, findData.cFileName);

sprintf(found\_file.path, "%s\\%s", current\_path, findData.cFileName);

found\_file.is\_directory = 0; // 不是目录

FindClose(hFind);

return &found\_file;

}

} while (FindNextFile(hFind, &findData) != 0);

FindClose(hFind);

return NULL;

}

void move\_file\_or\_directory(char \*source\_name, char \*dest\_name) {

// 移动文件或目录的函数实现

char source\_path[MAX\_PATH\_LEN], dest\_path[MAX\_PATH\_LEN];

sprintf(source\_path, "%s\\%s", current\_path, source\_name);

sprintf(dest\_path, "%s\\%s\\%s", current\_path, dest\_name, source\_name); // 目标路径包含源文件名

if (MoveFile(source\_path, dest\_path)) {

printf("文件/目录移动成功。\n");

} else {

printf("移动失败或文件/目录不存在。\n");

}

}

void list\_directory\_contents() {

// 列出目录内容的函数实现

char search\_path[MAX\_PATH\_LEN];

sprintf(search\_path, "%s\\\*", current\_path);

WIN32\_FIND\_DATA findData;

HANDLE hFind = FindFirstFile(search\_path, &findData);

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

printf("无法列出目录内容。\n");

return;

}

printf("目录 '%s' 的内容:\n", current\_path);

do {

printf("%s\n", findData.cFileName);

} while (FindNextFile(hFind, &findData) != 0);

FindClose(hFind);

}

void copy\_directory(char \*source\_name, char \*dest\_name) {

// 复制目录的函数实现

char source\_path[MAX\_PATH\_LEN], dest\_path[MAX\_PATH\_LEN];

sprintf(source\_path, "%s\\%s", current\_path, source\_name);

sprintf(dest\_path, "%s\\%s", current\_path, dest\_name);

if (!CreateDirectory(dest\_path, NULL)) {

printf("创建目标目录失败。\n");

return;

}

char search\_path[MAX\_PATH\_LEN];

sprintf(search\_path, "%s\\\*", source\_path);

WIN32\_FIND\_DATA findData;

HANDLE hFind = FindFirstFile(search\_path, &findData);

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

printf("无法复制目录内容。\n");

return;

}

do {

if (strcmp(findData.cFileName, ".") != 0 && strcmp(findData.cFileName, "..") != 0) {

char source\_file[MAX\_PATH\_LEN], dest\_file[MAX\_PATH\_LEN];

sprintf(source\_file, "%s\\%s", source\_path, findData.cFileName);

sprintf(dest\_file, "%s\\%s", dest\_path, findData.cFileName);

if (findData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) {

copy\_directory(source\_file, dest\_file);

} else {

CopyFile(source\_file, dest\_file, FALSE);

}

}

} while (FindNextFile(hFind, &findData) != 0);

FindClose(hFind);

printf("目录复制成功。\n");

}

void change\_dir(char \*path) {

// 改变当前工作目录的函数实现

char new\_path[MAX\_PATH\_LEN];

sprintf(new\_path, "%s\\%s", current\_path, path);

if (SetCurrentDirectory(new\_path)) {

GetCurrentDirectory(MAX\_PATH\_LEN, current\_path); // 更新当前路径

printf("当前目录切换到 '%s'。\n", current\_path);

} else {

printf("切换目录失败或目录不存在。\n");

}

}

void go\_back() {

// 返回上一级目录的函数实现

if (!SetCurrentDirectory("..")) {

printf("返回上级目录失败。\n");

return;

}

GetCurrentDirectory(MAX\_PATH\_LEN, current\_path); // 更新当前路径

printf("当前目录切换到 '%s'。\n", current\_path);

}

void display\_current\_path() {

// 显示当前工作目录路径的函数实现

GetCurrentDirectory(MAX\_PATH\_LEN, current\_path);

printf("当前路径: %s\n", current\_path);

}