****

**《操作系统》课程设计报告**

**学生姓名：**

**学生学号：**

**班 级：**

**指导教师：**

**完成日期：**

**南通大学信息科学技术学院**

**目 录**

[一. 设计说明 1](#_Toc84197480)

[二. 系统分析 4](#_Toc84197481)

[三. 总体设计 5](#_Toc84197482)

[四. 详细设计 6](#_Toc84197483)

[五. 实现 12](#_Toc84197484)

[六. 总结 15](#_Toc84197485)

[参考文献 16](#_Toc84197486)

[附录 16](#_Toc84197487)

# 一. 设计说明

**1.设计目的**

⑴加深和巩固学生对《操作系统》课程知识的理解，培养学生综合运用操作系统基本原理、方法和设计技术针对特定问题进行系统设计与实现的能力，加强学生的项目经验， 使学生初步具备设计复杂工程问题的解决方案和系统的能力。

⑵培养学生就计算机专业相关问题进行有效交流和沟通的能力，加强学生自主学习和团队合作能力，使学生具备查阅本专业文献、规范撰写设计报告、清晰阐述系统设计过程的能力。

**2.设计内容**

简单文件系统的模拟：设计一个文件系统的模拟系统，使用主存空间模拟磁盘或直接使用磁盘，实现文件系统的相关功能的模拟。

**3.设计要求**

⑴理解文件管理和主存管理的基本原理和方法；

⑵研究课题的解决方案；

⑶模拟基于多级目录的单用户单任务的文件系统；

⑷模拟文件存储空间的分配；

⑸模拟空闲磁盘空间的管理；

⑹模拟多级目录结构的文件管理，实现磁盘格式化、目录创建、目录删除、目录显示、更改当前目录、文件创建、文件打开、文件关闭、文件删除、读文件和写文件等功能；

⑺若采用位示图来管理文件存储空间，并采用显式链接分配方式，则将位示图合并到文件分配表；（\*）

⑻实现退出文件系统功能，将整个虚拟磁盘上的内容以一个文件的形式全部保存到系统真正的磁盘上，下次启动该文件系统时自动装入到主存中的虚拟磁盘上，以供后续使用。（\*）

⑼使用磁盘驱动程序，在真实磁盘上完成系统。（\*）

**4.任务分工**

①朱恒宇：负责编写模拟基于多级目录的单用户单任务文件系统，包括采取位示图来管理文件存储空间、设计文件的多级目录结构、并在此基础上实现了多用户多级目录文件系统。

②张家铭：设计总程序大框架以及各函数模块，将位示图与文件分配表合并，与朱恒宇同学协作配合完成该模拟文件系统。

③陈晓庆：收集相关资料，与小组成员密切合作及时交流，撰写实验报告。

# 二. 系统分析

**1、系统功能：**

①文件管理功能：含有文件的创建、打开、读写、关闭和删除等操作。②目录管理功能：含有目录的创建、删除、显示、更改当前目录等操作。

③存储空间管理功能：采用位示图来管理文件存储空间，并采用显式链接分配方式，将位示图合并到文件分配表。分配和释放文件存储空间，以及空闲磁盘空间的管理。

④系统状态管理功能：含有系统启动和关闭，虚拟磁盘内容的保存和加载等操作。

⑤用户权限管理功能：分为超级管理员、管理员以及普通用户，从而可以实现对文件和目录的权限控制。t

**2、系统性能：**

①响应时间：衡量文件系统执行操作所需的时间。通过测量读取和写入文件等常见操作的平均响应时间来评估文件系统的性能。

②吞吐量：衡量文件系统在单位时间内可以处理的操作数量。通过并发访问测试或压力测试来确定文件系统能够处理的最大负载和吞吐量。

③文件系统大小：考虑文件系统本身的大小对性能的影响。文件系统越大，可能导致遍历和索引等操作的速度变慢，从而降低性能。

④空间利用率：评估文件系统对存储空间的有效利用程度。计算已用空间与总空间之比来判断文件系统的空间利用效率。

⑤并发性和并行性：考虑多个进程或线程同时对文件系统进行操作时的性能表现。并发性能较好的文件系统能够有效地处理并发访问请求，并保持数据一致性和操作正确性。

**3、系统约束及其解决方案：**

①约束：无法有效地管理存储空间，出现空闲空间碎片化或存储空间浪费的情况。

解决方案：可以采用位示图、链表或其他方法来管理存储空间分配，确保空闲空间的高效利用。

②约束：文件系统的读取和写入速度较慢，用户等待时间较长。

解决方案：可以通过采用缓存技术、预读取、延迟写入等方法来优化文件系统的访问速度，并且在设计时合理安排数据结构和算法，以提高效率。

③约束：多个并发操作会对文件系统数据的一致性产生影响，存在死锁和数据损坏等风险。

解决方案：可以使用事务管理、锁机制、多版本控制等技术来确保并发访问时的数据一致性，并进行恰当的错误处理和异常处理，以防止死锁和数据损坏。

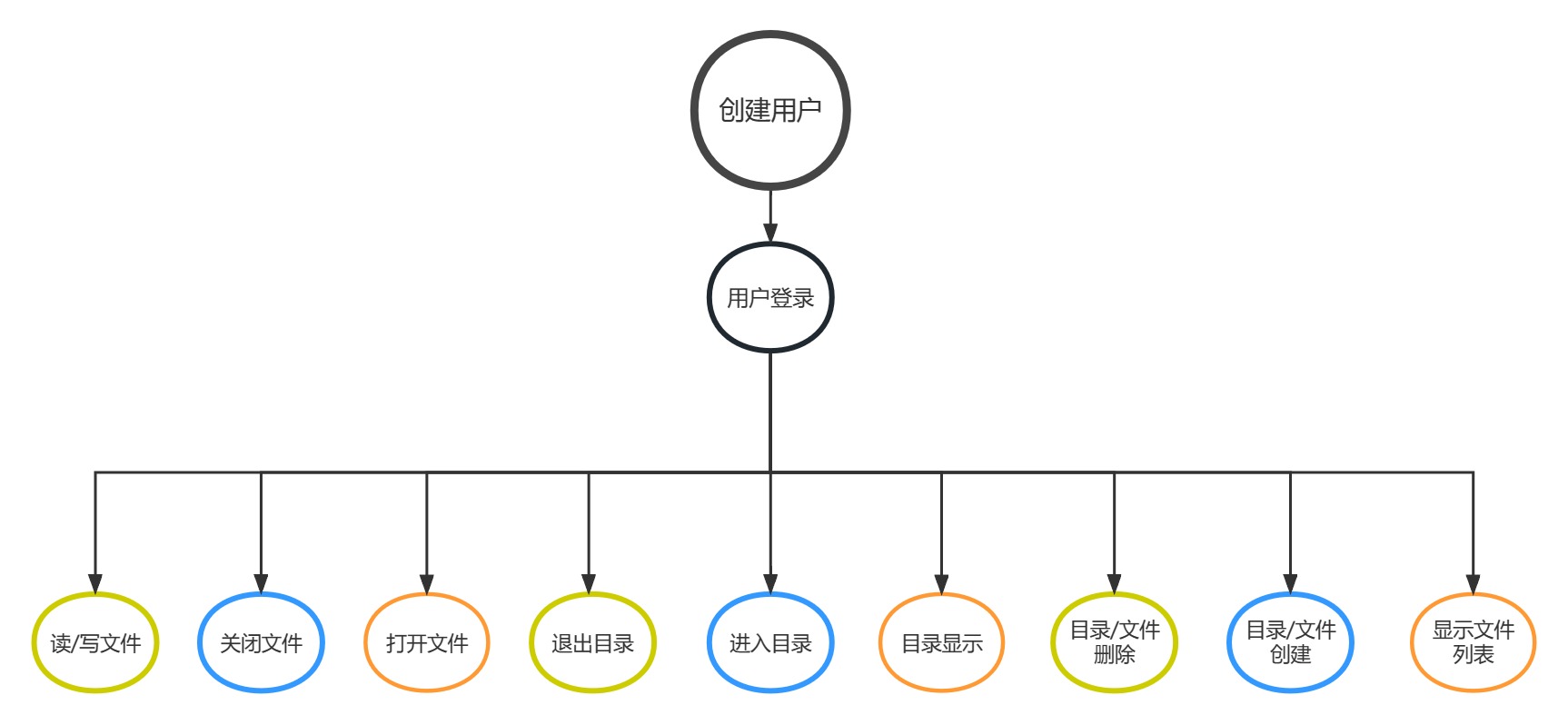
④约束：对文件系统的安全性要求较高，对用户进行身份认证和权限管理，防止未经授权的访问和数据泄露

解决方案：可以采用身份验证机制、加密技术等手段实现安全性和权限管理，确保只有合法用户才能访问和修改文件系统中的数据。

# 三. 总体设计

系统总体结构主要包括用户登录模块、新建用户模块、新建目录模块、显示当前路径下文件模块、读写文件模块、创建或删除目录/文件模块、以及目录后退模块。

主流程图



**1、登录模块：**

用户输入用户名和密码，在user文件中查找是否有此用户，核对密码。正确则登陆成功，当前目录设定到当前用户文件夹下。

**2、新建用户模块：**

先根据提示输入用户名，再判断此用户名是否已存在，若存在则重新输入，若没有，则用户创建成功。函数结束。

**3、初始化空闲区链表以及文件索引链模块：**

本函数模块在程序用运行后，对空闲区进行初始化。

**4、新建目录模块：**

输入参数为要建立的类型，f为文件，d为文件夹。并输入权限(0只读,1读写)。在创建目录时，并检测是否有重名的目录，若有则创建失败；没有重名的就在空闲区添加，否则，直接在文件中添加。

**5、显示当前路径下文件模块：**

查找当前目录下的各种文件并列举出来。

**6、目录后退模块：**

检测当前是否处于根目录下，不是则返回到上一目录；若当前处于根目录下，则无法后退。

**7、打开文件模块：**

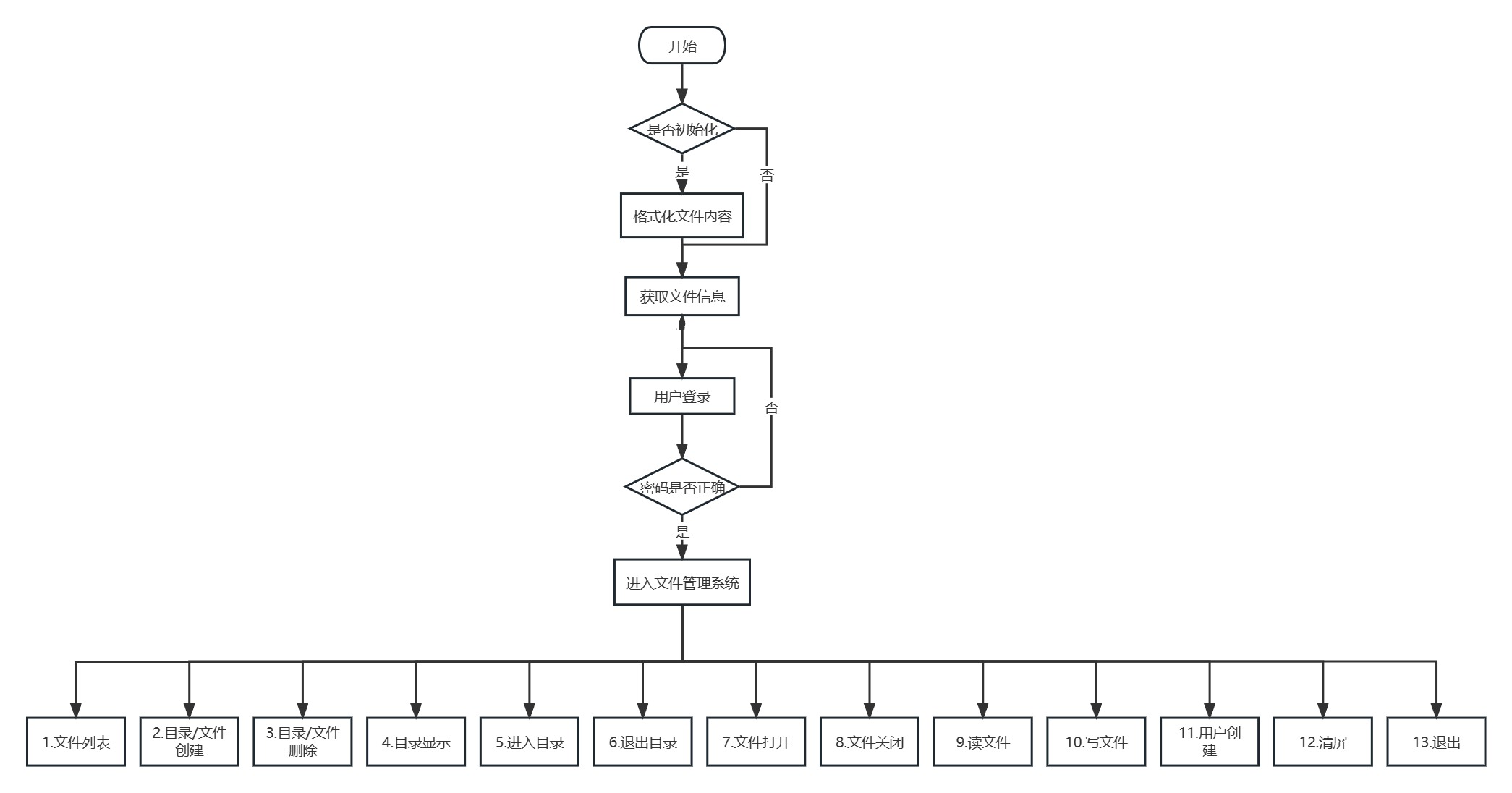
输入文件名，若当前目录中存在该文件，则检测当前文件是否已经打开，若未打开则将其添加到打开链表中。

**8、关闭文件模块：**

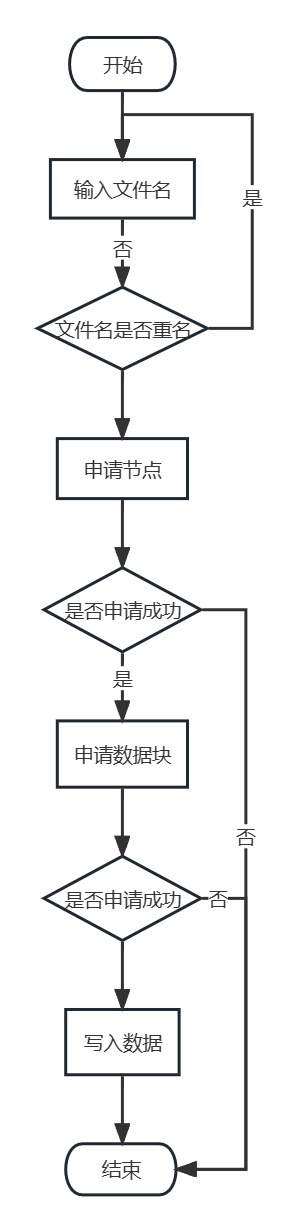
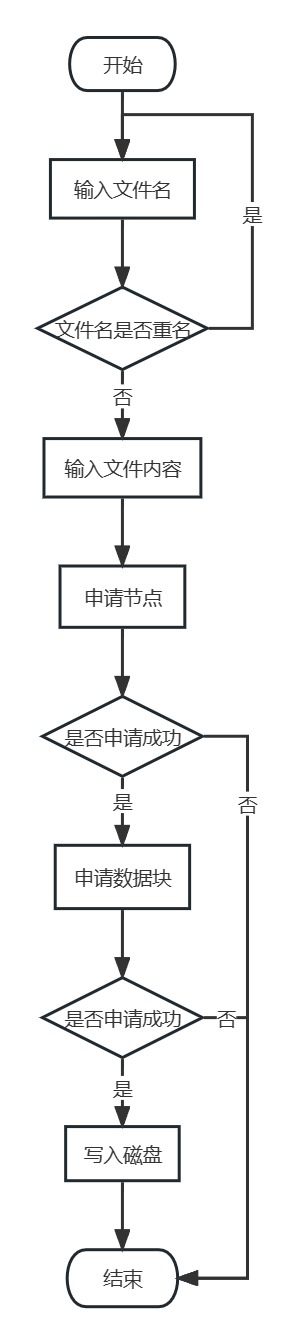
输入文件名，若当前目录中存在该文件，则检测当前文件是否已经打开，若已经打开则将其添加关闭。

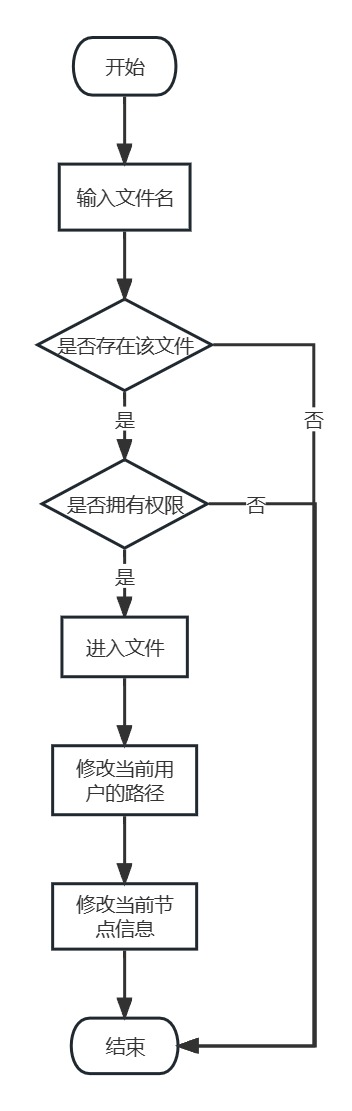
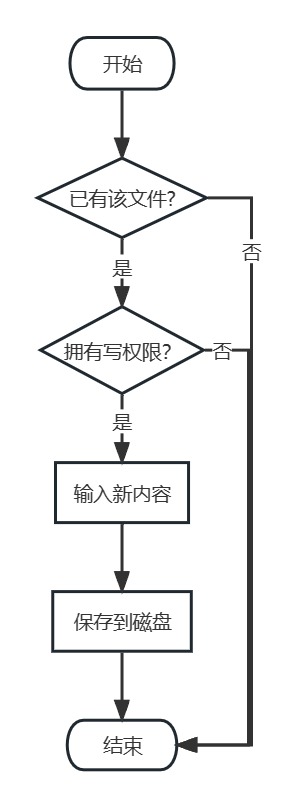
# 四. 详细设计

**1、主函数**

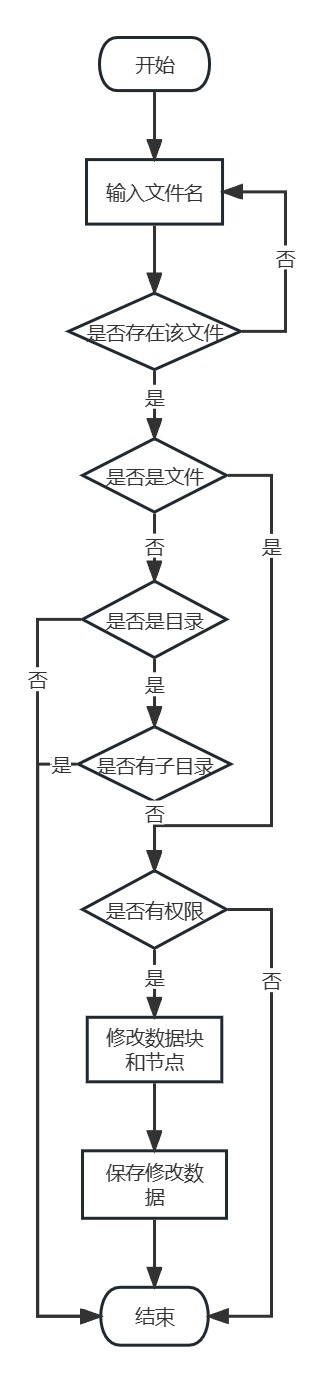
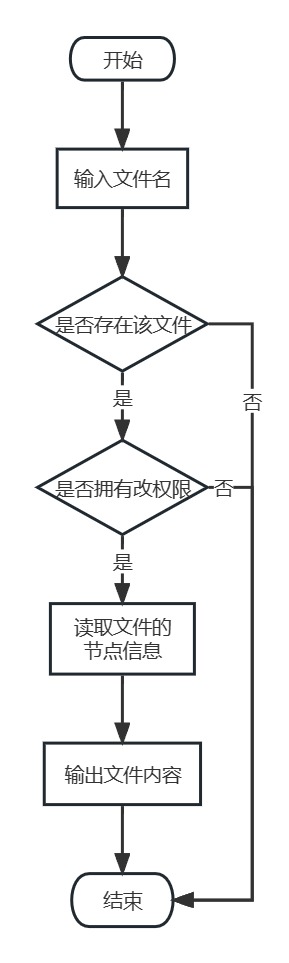


**2、创建文件 3、创建文件夹**

****

**4、写文件 5、打开文件**

**6、读文件 7、删除文件**



**以用户登录模块示例**

bool user\_login() {

char name[10];

        char pass[10];

        cout << "\n\t\t\t    ○用户名:";

cin >> name;

cout << "\t\t\t    ○密  码:";

cin >> pass;

if ((strcmp("root", name) == 0) && (strcmp("123456", pass) == 0))//管理员

       {

            usernowpride = 1;

            strcpy(usernowname, "root");

            return true;

        }

        FILE \*fp = NULL;

struct user actuser;

       if (!(fp = fopen("user", "ab+"))) {

cout << "Error:用户表错误\n";

            return false;

    }

        rewind(fp);

while (!feof(fp)) {

fread(&actuser, sizeof(struct user), 1, fp);

    if ((strcmp(actuser.name, name) == 0) && (strcmp(actuser.pass, pass)  == 0)) {

            usernowpride = actuser.pride;//记录当前用户的权限

            strcpy(usernowname, actuser.name);//记录当前用户的主路径

            nowlevel = -1;//记录当前目录层次

            fclose(fp);

            //设置路径

            if (strcmp(usernowpath, "\\") != 0)       //不是根目录就添加斜杠

            {

                strcat(usernowpath, "\\");

            }

            strcat(usernowpath, usernowname);

            return true;

        }

    }

        cout << "Error:用户名或密码无效，请核对后再输入\n";

fclose(fp);

return false;

}

**代码分析：**

上段代码定义了字符数组name和pass，用于存储用户输入的用户名和密码。提示用户输入用户名和密码，并将用户输入存储到相应的字符数组中。

检查用户输入的用户名和密码是否与超级管理员"root"和密码"123456"匹配。如果匹配，则将当前用户的权限设置为管理员权限（1），将当前用户名设置为"root"，并返回true。同时声明一个文件指针fp，初始化为NULL。打开名为"user"的文件，以二进制追加和读取的方式打开。如果打开失败，则输出错误信息并返回false。将文件指针重新定位到文件的起始位置。然后进入一个循环，读取文件中的每个用户记录。每当读取到一个用户记录后，比较读取到的用户名和密码与输入的用户名和密码是否匹配。如果匹配，则记录当前用户的权限、用户名和主路径，并关闭文件。设置当前目录层次为-1，并将当前用户的主路径添加到用户当前路径中。若返回true表示登录成功。如果循环结束仍未找到匹配的用户记录，则输出错误信息并关闭文件，返回false表示登录失败。

# 五. 实现

1、此次课程设计在win11系统下进行，所用开发环境为Visual Studio Code，编程语言为C++。

2、通过分析课程设计要求，具体设计出如下数据结构:

//用户类型定义

struct user

{

    char name[10];//用户名

    int pride;//用户权限，1为管理员，0为普通用户

    char pass[10];//用户密码

};

//定义空白区项

struct freeb

{

           int number;

           struct freeb \* next;

};

//定义文件打开项

struct fileb

{

int parent;//所在父节点

          char name[15];//文件名

          int pride;//读写权限，0只读，1读写

          int  rex;//读写状态，0为没有，1读2写

struct fileb \* next;

};

//定义文件索引项

struct findex

{

         char name[15];

         int number;

         int parent;

         char kind;

         struct findex \* next;

};

//定义目录表项

struct dreitem

{

         char name[25];//目录或者文件名称

         int share; //共享数

         int parent;//上层目录

         int pride;//文件操作权限，0只读，1读写

         int empty;//是否是空闲块，0为空闲块，1为非空

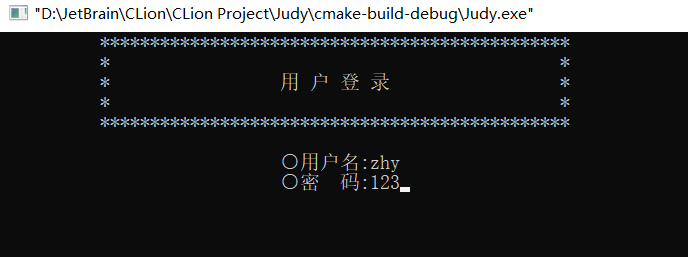
         char kind;//类型，文件为f，文件夹为d

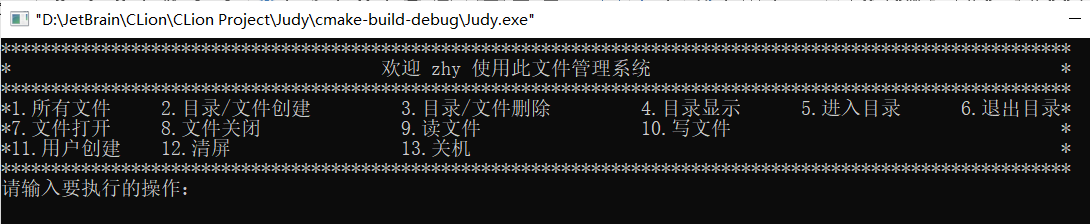
};

3.运行截图

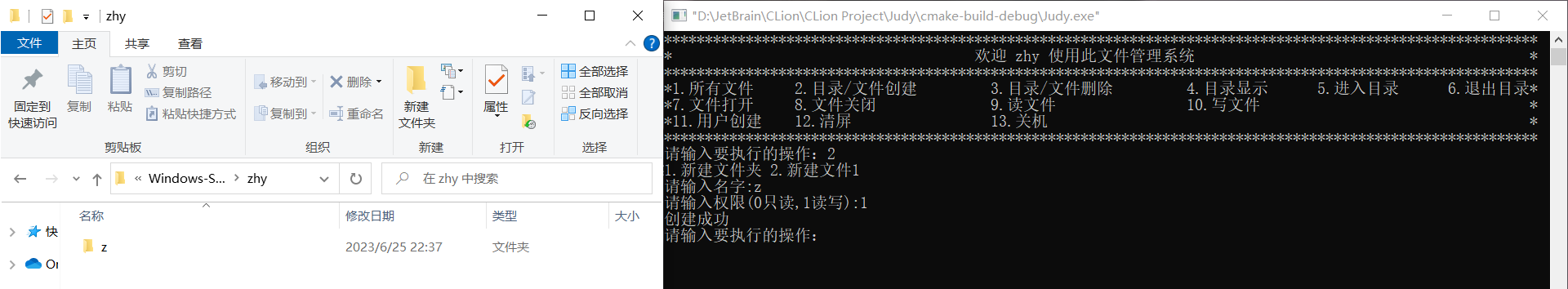
**①磁盘格式化后进入锁屏状态**

**②用户使用界面**

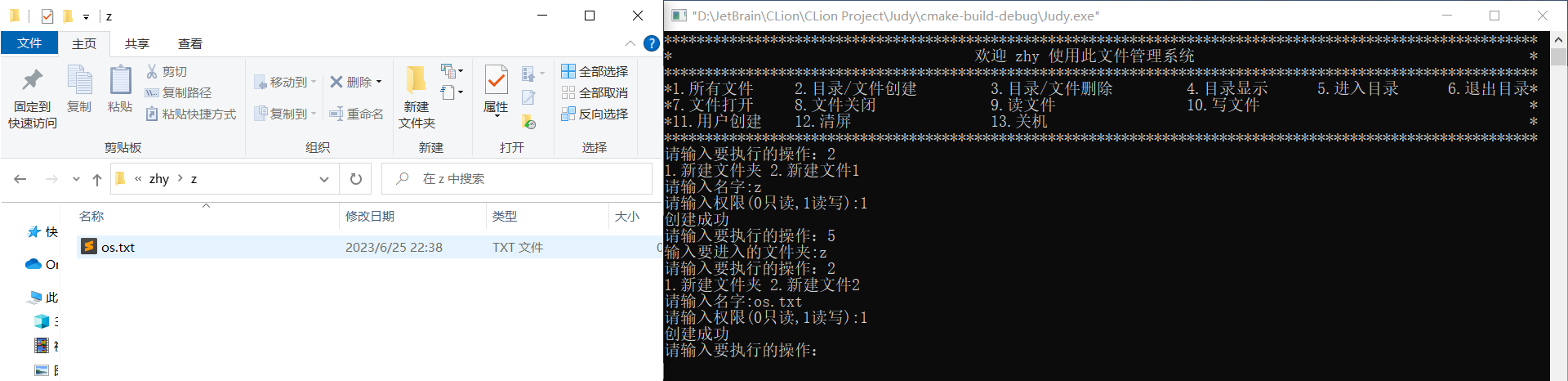




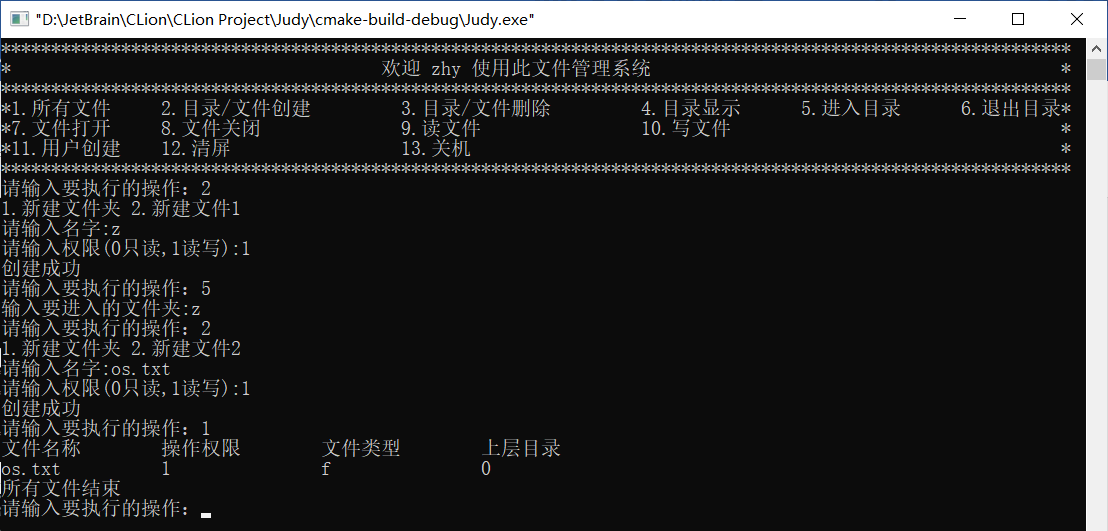
**③创建文件夹**



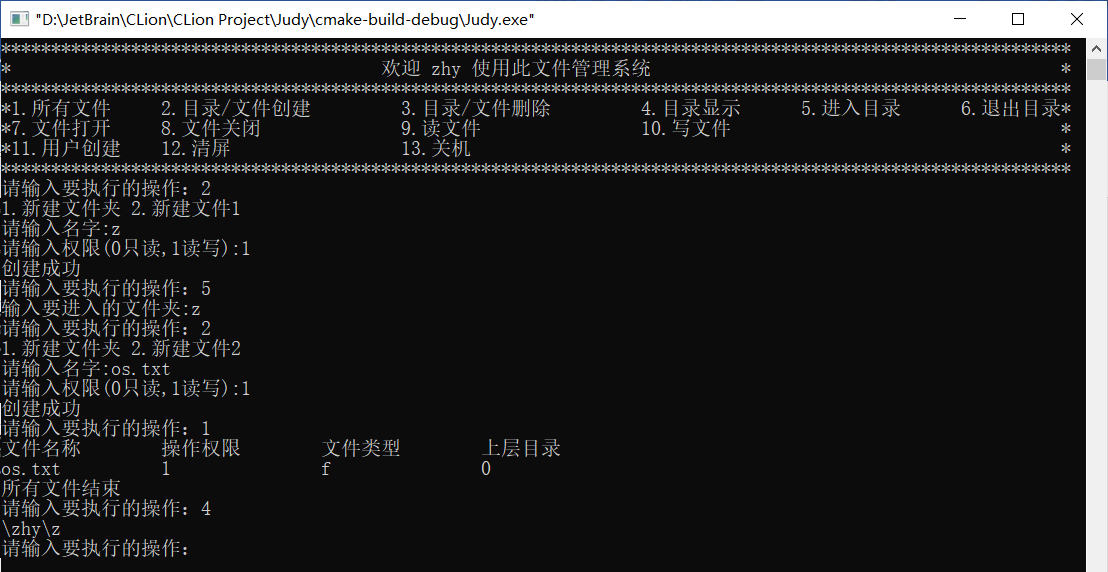
**④创建文件**



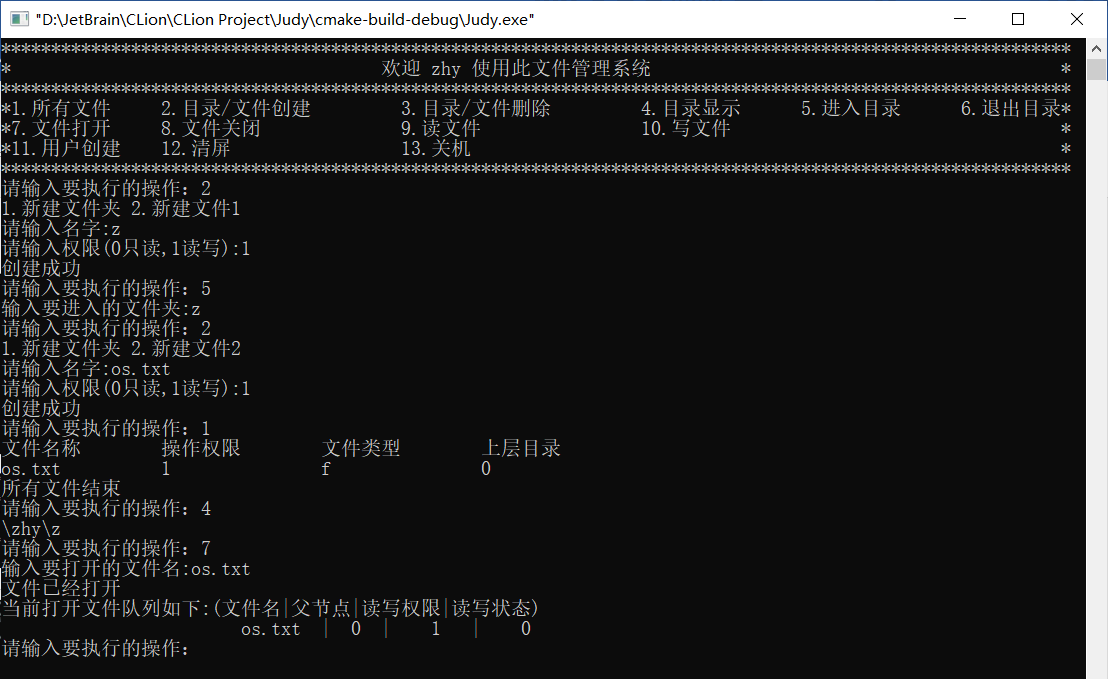
**⑤显示文件列表**

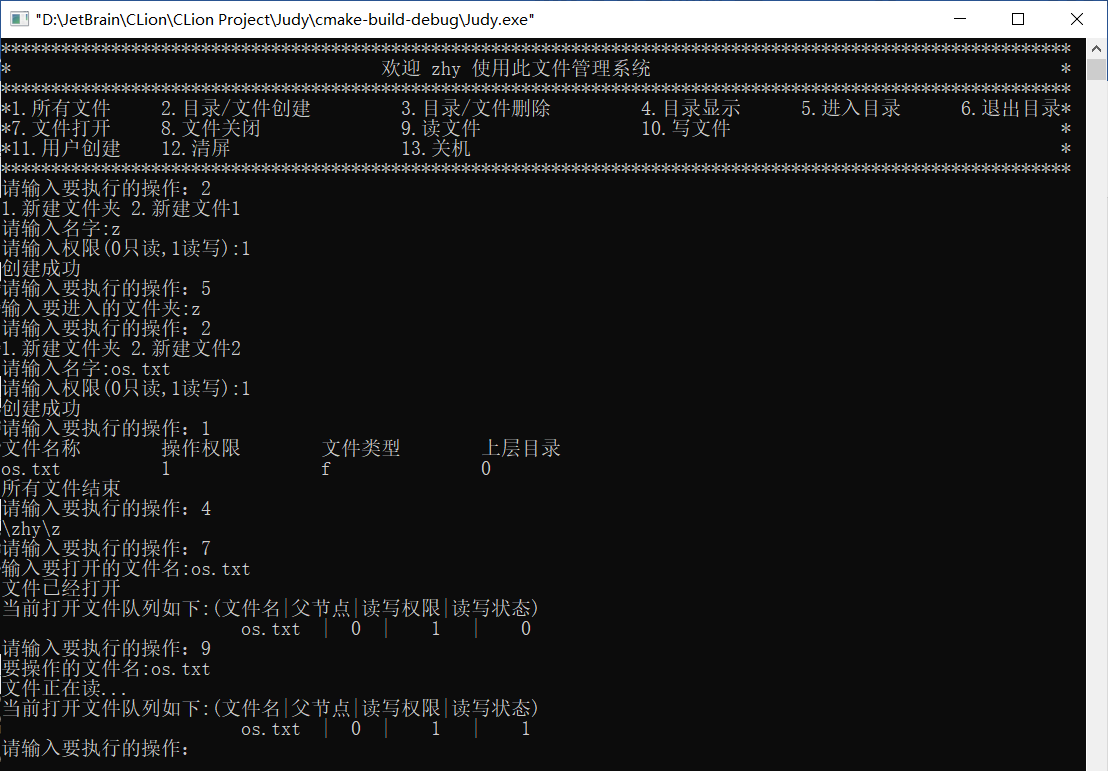


**⑥显示当前路径**

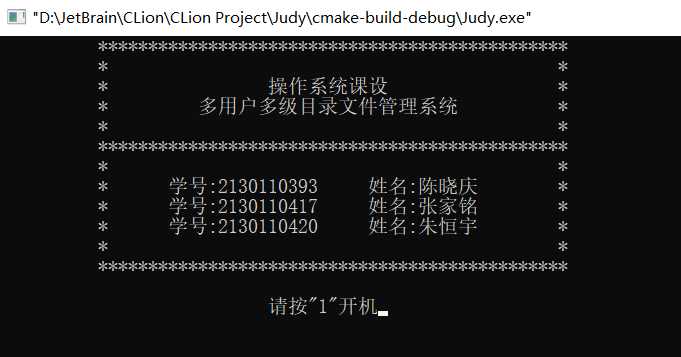


**⑦打开文件**

**⑧读文件**



**⑨关机**



# 六. 总结

此次模拟文件管理系统的设计，基本上全部满足了设计任务的要求，此外还添加了一些其他功能，存储区可以正常的分配与回收，使整个系统更加具有实用性。在实验过程中，我们遇到了一些错误和异常情况，比如文件打开失败、文件路径显示错误等。通过解决这些问题，我们积累了宝贵的经验，学会了如何处理错误和异常情况，保障文件系统的稳定性和安全性。虽然这个只是简单的模拟文件系统，但对我们来说是很大的收获，凝集辛勤的汗水。从理论到实践，更好的联系实际，从而更好的巩固书本上的知识，多接触些这样有意义、实际性的操作课程，这样才能培养我们深度学习操作系统的兴趣，而不是仅仅局限于课本上的知识。

在此次设计中，我们全面了解了文件系统的各个组成部分，包括磁盘空间管理、目录结构、索引节点等概念。并且在实践对这些概念有了更深刻的理解，从而更好地掌握了如何设计和实现一个简单的文件系统。还掌握了创建、读取、写入、删除文件以及管理目录和路径解析等基本操作。这些操作的实践加深了我们对文件系统的使用和管理的认识，提高了操作技能。

通过此次课程设计，培养了我们的问题解决、编程和调试能力。在未来的学习中，我们会进一步研究和探索更复杂的文件系统，如分布式文件系统、日志文件系统等，拓宽自己的知识广度和深度。同时，保持对新技术和发展的关注，并进行持续的学习和实践，在实践中积累丰富的知识，打下坚实的基础，从而在未来的学习和工作中取得更好的成果。

# 参考文献

1. 费祥林,骆斌.操作系统教程（第5版）[M]. 北京：高等教育出版社.2014.2：297-327
2. 孟庆昌.操作系统[M]. 北京：电子工业出版社.2004.
3. 汤晓丹等.现代操作系统[M]. 北京：电子工业出版社.2008.

# 附录

#include <stdio.h>

#include <string.h>

#include <process.h>

#include <malloc.h>

#include <stdlib.h>

#include <conio.h>

#include <iostream>

#include <filesystem>

#include <fstream>

#define getb(type) (type\*)malloc(sizeof(type))

using namespace std;

namespace fs = std::filesystem;

int usernowpride;//全局变量，当前用户类型

char usernowname[10];//全局变量，当前用户名

int nowlevel; //全局变量，当前目录层次

char usernowpath[200] = "\\\0";

//用户类型定义

struct user {

    char name[10];//用户名

    int pride;//用户权限，1为管理员，0为普通用户

    char pass[10];//用户密码

};

//定义空白区项

struct freeblock {

    int number;

    struct freeblock \*next;

};

struct freeblock \*fblock = NULL;//全局变量，系统空闲区链

//定义文件打开项

struct fileb {

    int parent;//所在父节点

    char name[15];//文件名

    int pride;//读写权限，0只读，1读写

    int rex;//读写状态，0为没有，1读2写

    struct fileb \*next;

};

struct fileb \*flink = NULL;//全局变量，系统打开文件链

//定义文件索引项

struct fileindex {

    char name[15];

    int number;

    int parent;

    char kind;

    struct fileindex \*next;

};

struct fileindex \*findex = NULL;//全局变量，文件索引链

//定义目录表项

struct directory {

    char name[25];//目录或者文件名称

    int parent;//上层目录

    int pride;//文件操作权限，0只读，1读写

    int empty;//是否是空闲块，0为空闲块，1为非空

    char kind;//类型，文件为f，目录为d

};

//用户创建

bool user\_create() {

    struct user newuser;

    char name[10];

    char pass[10];

    int pride;

    if (usernowpride != 1) {

        cout << "当前用户没有新建用户的权限\n";

        return false;

    }

    FILE \*fp;

    if ((fp = fopen("user", "ab+")) == NULL) {

        cout << "用户表打开失败\n";

        return false;

    } else {

        cout << "请输入用户名:";

        cin >> name;

        if (strcmp(name, "root") == 0) {

            printf("Error:此为超级管理员\n");

            return false;

        }

        rewind(fp);

        while (!feof(fp)) {

            fread(&newuser, sizeof(struct user), 1, fp);

            if (strcmp(newuser.name, name) == 0) {

                cout << "该用户名已经存在\n";

                fclose(fp);

                return false;

            }

        }

        cout << "请输入用户密码:";

        cin >> pass;

        cout << "请输入用户权限(0普通用户，1管理员):";

        cin >> pride;

        strcpy(newuser.name, name);

        strcpy(newuser.pass, pass);

        newuser.pride = pride;

//     FILE fpuser;//为新建用户建立用户目录文件

        if (!fopen(newuser.name, "ab+")) {

            cout << "用户创建失败\n";

            //如创建失败则把已经建立的用户目录删除

            char cmd[20] = "DEL ";

            strcat(cmd, newuser.name);

            system(cmd);

            fclose(fp);

            return false;

        }

        if (!fwrite(&newuser, sizeof(struct user), 1, fp)) {

            cout << "创建失败\n";

            fclose(fp);

            return false;

        } else {

            cout << "用户创建成功\n";

            cout << "请输入要执行的操作：";

            fclose(fp);

            return true;

        }

    }

}

//用户登陆

bool user\_login() {

    char name[10];

    char pass[10];

    cout << "\n\t\t\t    ○用户名:";

    cin >> name;

    cout << "\t\t\t    ○密  码:";

    cin >> pass;

    if ((strcmp("root", name) == 0) && (strcmp("123456", pass) == 0))//管理员

    {

        usernowpride = 1;

        strcpy(usernowname, "root");

        return true;

    }

    FILE \*fp = NULL;

    struct user actuser;

    if (!(fp = fopen("user", "ab+"))) {

        cout << "Error:用户表错误\n";

        return false;

    }

    rewind(fp);

    while (!feof(fp)) {

        fread(&actuser, sizeof(struct user), 1, fp);

        if ((strcmp(actuser.name, name) == 0) && (strcmp(actuser.pass, pass) == 0)) {

            usernowpride = actuser.pride;//记录当前用户的权限

            strcpy(usernowname, actuser.name);//记录当前用户的主路径

            nowlevel = -1;//记录当前目录层次

            fclose(fp);

            //设置路径

            if (strcmp(usernowpath, "\\") != 0)              //不是根目录就添加斜杠

            {

                strcat(usernowpath, "\\");

            }

            strcat(usernowpath, usernowname);

            return true;

        }

    }

    cout << "Error:用户名或密码无效，请核对后再输入\n";

    fclose(fp);

    return false;

}

//初始化空闲区链表以及文件索引链

void list\_init() {

    //清空各链表

    findex = NULL;

    fblock = NULL;

    int i = 0;

    struct directory dnow;

    FILE \*fp;

    if (!(fp = fopen(usernowname, "rb"))) {

        cout << "Error:打开用户目录失败\n";

        return;

    } else {

        int enp;

        int sp;

        fseek(fp, 0, 2);

        enp = ftell(fp);

        fseek(fp, 0, 0);

        sp = ftell(fp);

        if (sp == enp) return;

        while (!feof(fp)) {

            fread(&dnow, sizeof(struct directory), 1, fp);

            if (dnow.empty == 0) {

                //把空闲区连到空闲链表中

                struct freeblock \*fb = getb(struct freeblock);

                fb->number = i;

                fb->next = NULL;

                struct freeblock \*p = fblock;

                if (p == NULL) {

                    fblock = getb(struct freeblock);

                    fblock->next = fb;

                } else {

                    while (p->next != NULL) {

                        p = p->next;

                    }

                    p->next = fb;

                }

            } else {

                //建立索引表

                struct fileindex \*fi = (struct fileindex \*) malloc(sizeof(struct fileindex));

                strcpy(fi->name, dnow.name);

                fi->number = i;

                fi->kind = dnow.kind;

                fi->parent = dnow.parent;

                fi->next = NULL;

                struct fileindex \*pi = findex;

                if (pi == NULL) {

                    findex = getb(struct fileindex);

                    findex->next = fi;

                } else {

                    while (pi->next != NULL) {

                        pi = pi->next;

                    }

                    pi->next = fi;

                }

            }

            i++;

        }

        fclose(fp);

        return;

    }

}

//目录/文件创建

//输入参数为要建立的类型，f为文件，d为文件夹

void drec\_file\_create(char ch) {

    struct directory dnow;

    char name[15];

    char pride;//权限，0只读，1读写

    int i;//记录空闲区区号

    bool cancrd = true;//用于判断是否有重名文件

    cout << "请输入名字:";

    fflush(stdin);

    cin >> name;

    //判断是否已经存在相同名字的文件或者目录

    struct fileindex \*fi = findex;

    while (fi) {

        if ((strcmp(fi->name, name) == 0) && (fi->parent == nowlevel) && (fi->kind == ch)) {

            cout << "Error:此文件或者目录已经存在\n";

            cancrd = false;

            break;

        }

        fi = fi->next;

    }

    if (!cancrd) return;

    cout << "请输入权限(0只读,1读写):";

    cin >> pride;

    dnow.empty = 0;

    strcpy(dnow.name, name);

    dnow.parent = nowlevel;

    if (pride == '0') {

        dnow.pride = 0;

    } else {

        dnow.pride = 1;

    }

    dnow.kind = ch;

    dnow.empty = 1;

    FILE \*fp;

    if (!(fp = fopen(usernowname, "rb+"))) {

        cout << "错误";

        return;

    } else {

        std::string path = "C:\\" + string(usernowpath) + "\\" + string(name);

        if (dnow.kind == 'd')

            fs::create\_directories(path);

        else

            std::ofstream file(path);

        if (fblock == NULL || fblock->next == NULL)       //空闲区没有就直接在文件添加

        {

            fseek(fp, 0, 2);

            fwrite(&dnow, sizeof(struct directory), 1, fp);

            cout << "创建成功\n";

            fclose(fp);

            return;

        } else {

            struct freeblock \*p = fblock->next;//拿第一个空闲区来存放文件

            if (p != NULL) {

                //fblock=p->next;

                i = p->number;

                fblock->next = p->next;

            }

            free(p);

            fseek(fp, i \* sizeof(struct directory), 0);

            fwrite(&dnow, sizeof(struct directory), 1, fp);

            cout << "创建成功\n";

            fclose(fp);

            return;

        }

    }

}

//列表当前路径下的文件

void file\_list() {

    long ep;

    FILE \*fp;

    struct directory drenow;

    fp = fopen(usernowname, "rb");

    fseek(fp, 0, 2);

    ep = ftell(fp);

    fseek(fp, 0, 0);

    cout << "文件名称\t操作权限\t文件类型\t上层目录\n";

    while (ep != ftell(fp)) {

        fread(&drenow, sizeof(struct directory), 1, fp);

        if (drenow.parent == nowlevel && drenow.empty != 0) {

            cout << drenow.name << "\t\t";

            cout << drenow.pride << "\t\t";

            cout << drenow.kind << "\t\t";

            cout << drenow.parent << "\n";

        }

    }

    cout << "所有文件结束\n";

    fclose(fp);

}

//退出目录

bool direc\_back() {

    if (nowlevel == -1) {

        cout << "Error:当前已为用户根目录,无法再回退\n";

        return false;

    }

    char name[15];

    int i;

    i = nowlevel;

    struct directory drecnow;

    FILE \*fp;

    fp = fopen(usernowname, "rb");

    fseek(fp, i \* sizeof(struct directory), 0);

    fread(&drecnow, sizeof(struct directory), 1, fp);

    fclose(fp);

    strcpy(name, drecnow.name);

    nowlevel = drecnow.parent;

    char cc;         //检测'\0'位置

    int ccidx = 0;          //记录最好一个'\'位置

    int j = 0;

    cc = usernowpath[j];

    while (cc != '\0') {

        j++;

        cc = usernowpath[j];

        if (cc == '\\') {

            ccidx = j;

        }

    }

    if (ccidx != 0) {

        usernowpath[ccidx] = '\0';

    }

    return true;

}

//进入目录

bool direc\_enter(char chpath[15]) {

    struct directory drenow;

    char mypath[15];

    int i = 0;

    strcpy(mypath, chpath);

    if (strcmp(mypath, "..") == 0) {

        if (direc\_back()) {

            return true;

        } else {

            return false;

        }

    }

    FILE \*fp;

    fp = fopen(usernowname, "rb");

    while (!feof(fp)) {

        fread(&drenow, sizeof(struct directory), 1, fp);

        if ((strcmp(drenow.name, mypath) == 0) && (drenow.kind == 'd') && (drenow.parent == nowlevel)) {

            nowlevel = i;

            fclose(fp);

            if (strcmp(usernowpath, "\\") != 0)              //不是根目录就添加斜杠

            {

                strcat(usernowpath, "\\");

            }

            strcat(usernowpath, mypath);

            return true;

        }

        i++;

    }

    fclose(fp);

    return false;

}

//进入路径完整版

void intopath() {

    int tempnl = nowlevel;

    char tempunp[200];

    strcpy(tempunp, usernowpath);

    char wholepath[100];

    char name[15];

    cout << "输入要进入的文件夹:";

    cin >> wholepath;

    int i = 0;

    char cc = wholepath[i];

    while (cc != '\0') {

        int j = 0;

        name[j] = cc;

        while (cc != '\0' && cc != '\\') {

            name[j] = cc;

            j++;

            i++;

            cc = wholepath[i];

        }

        if (i != 0) {

            name[j] = '\0';

            if (!direc\_enter(name)) {

                cout << "Error:路径输入错误，请核对\n";

                //如果路径错误，则还原

                nowlevel = tempnl;

                strcpy(usernowpath, tempunp);

                return;

            }

        }

        if (cc == '\0') break;

        i++;

        cc = wholepath[i];

    }

}

//删除文件,递归删除文件夹和文件夹里面的内容

//删除文件

bool drec\_file\_remove(int p) {

    bool isfind = false;

    int i = 0;

    int temp = 0;

    struct directory drecnow;

    FILE \*fp;

    fp = fopen(usernowname, "rb+");

    long ep;

    fseek(fp, 0, 2);

    ep = ftell(fp);

    fseek(fp, 0, 0);

    while (ep != ftell(fp)) {

        fread(&drecnow, sizeof(struct directory), 1, fp);

        if (drecnow.parent == p) {

            isfind = true;

            temp = p;

            drec\_file\_remove(i);

        }

        i++;

    }

    if (!isfind) {

        drecnow.empty = 0;

        strcpy(drecnow.name, "    ");

        drecnow.parent = -1;

        fseek(fp, p \* sizeof(struct directory), 0);

        fwrite(&drecnow, sizeof(struct directory), 1, fp);

        fclose(fp);

        return true;

    } else {

        drec\_file\_remove(temp);

        fclose(fp);

        return false;

    }

}

void drec\_file\_del() {

    char name[15];

    cout << "输入要删除的目录/文件:";

    cin >> name;

    std::string path = "C:\\" + string(usernowpath) + "\\" + string(name);

    fs::remove\_all(path);

    fileindex \*fb = findex;//用于索引表

    fileindex \*fi;

    fileb \*fob = flink;//用于已打开文件链表

    if (fb == NULL || fb->next == NULL) {//搜索索引表

        cout << "Error:没有此目录/文件\n";

        return;

    }

    bool isit = true;

    while (fb != NULL && fb->next != NULL) {

        fi = fb;

        fb = fb->next;

        if ((strcmp(fb->name, name) == 0) && (fb->parent == nowlevel)) {

            isit = false;

            if (drec\_file\_remove(fb->number)) {

                fi->next = fb->next;

                free(fb);

                cout << "该目录/文件已被删除\n";

                return;

            }

        }

    }

    if (isit) {

        cout << "Error:没有此目录/文件\n";

    } else {

        cout << "该目录/文件已被删除\n";

    }

}

//打开文件

void file\_open() {

    long ep;

    int i = 0;

    FILE \*fp;

    struct directory drenow;

    char name[15];

    cout << "输入要打开的文件名:";

    cin >> name;

    struct fileb \*fb = flink;

    while (fb != NULL && fb->next != NULL) {

        fb = fb->next;

        if ((strcmp(fb->name, name) == 0) && (fb->parent == nowlevel)) {

            cout << "Error:该文件已经打开\n";

            return;

        }

    }

    fp = fopen(usernowname, "rb");                       //可用索引节点

    fseek(fp, 0, 2);

    ep = ftell(fp);

    fseek(fp, 0, 0);

    while (ep != ftell(fp)) {

        fread(&drenow, sizeof(struct directory), 1, fp);

        if ((strcmp(drenow.name, name) == 0) && (drenow.parent == nowlevel) && (drenow.kind == 'f')) {

            //添加到打开链表

            fileb \*pb = getb(struct fileb);

            strcpy(pb->name, name);

            pb->parent = nowlevel;

            pb->next = NULL;

            pb->rex = 0;

            pb->pride = drenow.pride;

            if (flink == NULL) {

                flink = getb(struct fileb);

                flink->next = pb;

            } else {

                pb->next = flink->next;

                flink->next = pb;

            }

            cout << "文件已经打开\n";

            fclose(fp);

            //这里是否要更新显示已打开链表

            return;

        }

        i++;

    }

    cout << "Error:当前目录下无此文件，请核对\n";

    fclose(fp);

}

//显示当前打开所有文件

void openfile\_queue() {

    if (flink == NULL || flink->next == NULL) {

        cout << "当前打开文件队列空\n";

        return;

    } else {

        cout << "当前打开文件队列如下:(文件名|父节点|读写权限|读写状态)\n";

        fileb \*fb = flink;

        while (fb != NULL && fb->next != NULL) {

            fb = fb->next;

            printf("\t\t\t%s  |  %d  |    %d   |    %d", fb->name, fb->parent, fb->pride, fb->rex);

        }

        cout << "\n";

    }

}

//关闭文件

void file\_close() {

    char name[15];

    cout << "输入要关闭的文件:";

    cin >> name;

    if (flink == NULL || flink->next == NULL) {

        cout << "Error:该文件没有打开\n";

        return;

    }

    fileb \*fb = flink;

    fileb \*ffb = NULL;

    while (fb != NULL && fb->next != NULL) {

        ffb = fb;

        fb = fb->next;

        if ((strcmp(fb->name, name) == 0) && (fb->parent == nowlevel)) {

            ffb->next = fb->next;

            free(fb);

            cout << "文件已经关闭\n";

            return;

        }

    }

    cout << "Error:当前路径下找不到你要关闭的文件，请确定文件名已经路径是否正确\n";

}

//读写文件，该文件必须先打开后才可以进行读写操作

//读写文件，输入1为读文件，输入2为写文件

void file\_readwrite(int ch) {

    char name[15];

    cout << "要操作的文件名:";

    cin >> name;

    if (flink == NULL || flink->next == NULL) {

        cout << "Error:该文件尚未打开\n";                         //可以调用文件打开函数，以后先修改

        return;

    }

    fileb \*fb = flink;

    while (fb != NULL && fb->next != NULL) {

        fb = fb->next;

        if ((strcmp(fb->name, name) == 0) && (fb->parent == nowlevel)) {

            //char ch;

            if (fb->rex != ch && fb->rex != 0) {

                if (ch == 2) {

                    if (fb->pride != 1) {

                        cout << "Error:该文件只读，没有写权限\n";

                        return;

                    } else {

                        cout << "当前文件正在读进程，是否终止读进程进行写?(y OR n):";

                    }

                } else {

                    cout << "当前文件正在写进程，是否终止写进程进行读?(y OR n):";

                }

                fflush(stdin);

                cin >> ch;

                if (ch == 'y' || ch == 'Y') {

                    if (ch == 2) {

                        fb->rex = 1;

                        cout << "文件正在读...\n";

                    } else {

                        fb->rex = 2;

                        cout << "文件正在写...\n";

                    }

                }

            } else {

                if (ch == 2 && fb->pride != 1) {

                    cout << "Error:该文件只读，没有写权限\n";

                    return;

                }

                fb->rex = ch;

                if (ch == 1) {

                    cout << "文件正在读...\n";

                } else {

                    cout << "文件正在写...\n";

                }

            }

            return;

        }

    }

}

//目录显示

void direct\_display() {

    cout << usernowpath << endl;

}

//注销

void exit() {

    strcpy(usernowname, "  ");

    nowlevel = -1;

    fblock = NULL;

    findex = NULL;

    strcpy(usernowpath, "\\");

    flink = NULL;

}

//修改用户类型

void user\_setpride() {

    char name[15];

    cout << "输入用户名字:";

    cin >> name;

    if ((strcmp("root", name) == 0))//超级用户

    {

        cout << "Error:超级管理员权限不可以更改\n";

        return;

    }

    FILE \*fp = NULL;

    struct user actuser;

    if (!(fp = fopen("user", "rb+"))) {

        cout << "Error:用户表错误\n";

        return;

    }

    rewind(fp);

    long np;

    while (!feof(fp)) {

        np = ftell(fp);

        fread(&actuser, sizeof(struct user), 1, fp);

        if ((strcmp(actuser.name, name) == 0)) {

            if (actuser.pride == 1) {

                int pp = 1;

                cout << "该用户是管理员\n输入0设置普通用户:";

                cin >> pp;

                if (pp == 0) {

                    actuser.pride = 0;

                }

            } else if (actuser.pride == 0) {

                int pp = 0;

                cout << "该用户是普通用户\n输入1设置为管理员:";

                cin >> pp;

                {

                    if (pp == 1) {

                        actuser.pride = 1;

                    }

                }

            }

            fseek(fp, np, 0);

            fwrite(&actuser, sizeof(struct user), 1, fp);

            fclose(fp);

            return;

        }

    }

}

void os() {

    printf("\t  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\t  \*　　　　　　　　　　　　　                   \* \n");

    printf("\t  \*     　　       操作系统课设  　     　  　  \*\n");

    printf("\t  \*         多用户多级目录文件管理系统  　  　  \*\n");

    printf("\t  \*　　　　　　　　　　　　　                   \* \n");

    printf("\t  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\t  \*　　　　　　　　　　　　　                   \* \n");

    printf("\t  \*　　　学号:2130110393     姓名:陈晓庆　　 　 \* \n");

    printf("\t  \*　　　学号:2130110417     姓名:张家铭　　 　 \* \n");

    printf("\t  \*　　　学号:2130110420     姓名:朱恒宇　　 　 \* \n");

    printf("\t  \*　　　　　　　　　　　　　　　　　　　　　　 \*\n");

    printf("\t  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    cout << "\n\t\t\t   请按\"1\"开机";

}

int main() {

    int choince;

    while (1) {

        os();

        bool enter;

        cin >> enter;

        while (enter) {

            system("cls");

            printf("\t  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

            printf("\t  \*　　　　　　　　　　　　　                   \* \n");

            printf("\t  \*                 用 户 登 录                 \*\n");

            printf("\t  \*　　　　　　　　　　　　　                   \* \n");

            printf("\t  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

            if (user\_login()) break;

        }

        if (strcmp(usernowname, "root") == 0) {

            system("cls");

            cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

            cout << "\*\t            欢迎超级管理员使用此文件管理系统               \t\*\n";

            cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

            cout << "\*                                                                       \*\n";

            cout << "\*\t1.新建用户\t\t2.修改用户\t\t3.退出\t\t\*\n";

            cout << "\*                                                                       \*\n";

            cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

            cout << "请输入要执行的操作：";

            bool lout = false;

            while (1) {

                fflush(stdin);

                scanf("%d", &choince);

                switch (choince) {

                    case 1:

                        user\_create();

                        break;

                    case 2:

                        user\_setpride();

                        break;

                    case 3:

                        char out;

                        printf("确定是否退出吗?(y OR n):");

                        fflush(stdin);

                        scanf("%c", &out);

                        if (out == 'y' || out == 'Y') {

                            lout = true;

                            exit();

                            system("cls");

                        }

                        break;

                    default:

                        printf("Error:错误命令\n");

                        break;

                }

                if (lout)

                    break;

            }

        } else {

            list\_init();

            system("cls");

            cout

                    << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

            cout << "\*\t\t\t\t      欢迎 " << usernowname << " 使用此文件管理系统         \t\t\t\t  \*\n";

            cout

                    << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

            cout << "\*1.所有文件\t2.目录/文件创建\t\t3.目录/文件删除\t\t4.目录显示\t5.进入目录\t6.退出目录\*\n";

            cout << "\*7.文件打开\t8.文件关闭\t\t9.读文件\t\t10.写文件\t\t\t\t  \*\n";

            cout << "\*11.用户创建\t12.清屏\t\t\t13.关机\t\t\t\t\t\t\t\t  \*\n";

            cout

                    << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

            cout << "请输入要执行的操作：";

            bool lout = false;

            while (1) {

                fflush(stdin);

                scanf("%d", &choince);

                switch (choince) {

                    case 1:

                        file\_list();

                        cout << "请输入要执行的操作：";

                        break;

                    case 2:

                        int cc;

                        printf("1.新建文件夹 2.新建文件");

                        scanf("%d", &cc);

                        if (cc == 1) {

                            drec\_file\_create('d');

                        }

                        if (cc == 2) {

                            drec\_file\_create('f');

                        }

                        list\_init();

                        cout << "请输入要执行的操作：";

                        break;

                    case 3:

                        drec\_file\_del();

                        list\_init();

                        cout << "请输入要执行的操作：";

                        break;

                    case 4:

                        direct\_display();

                        cout << "请输入要执行的操作：";

                        break;

                    case 5:

                        intopath();

                        cout << "请输入要执行的操作：";

                        break;

                    case 6:

                        direc\_back();

                        cout << "请输入要执行的操作：";

                        break;

                    case 7:

                        file\_open();

                        openfile\_queue();

                        cout << "请输入要执行的操作：";

                        break;

                    case 8:

                        file\_close();

                        openfile\_queue();

                        cout << "请输入要执行的操作：";

                        break;

                    case 9:

                        file\_readwrite(1);

                        openfile\_queue();

                        cout << "请输入要执行的操作：";

                        break;

                    case 10:

                        file\_readwrite(2);

                        openfile\_queue();

                        cout << "请输入要执行的操作：";

                        break;

                    case 11:

                        user\_create();

                        cout << "请输入要执行的操作：";

                        break;

                    case 12:

                        system("cls");

                        cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

                        cout << "\*\t\t\t\t      欢迎 " << usernowname << " 使用此文件管理系统         \t\t\t\t  \*\n";

                        cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

                        cout << "\*1.所有文件\t2.目录/文件创建\t\t3.目录/文件删除\t\t4.目录显示\t5.进入目录\t6.退出目录\*\n";

                        cout << "\*7.文件打开\t8.文件关闭\t\t9.读文件\t\t10.写文件\t\t\t\t  \*\n";

                        cout << "\*11.用户创建\t12.清屏\t\t\t13.关机\t\t\t\t\t\t\t\t  \*\n";

                        cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

                        cout << "请输入要执行的操作：";

                        break;

                    case 13:

                        char out;

                        printf("确定是否关机吗?(y OR n):");

                        fflush(stdin);

                        scanf("%c", &out);

                        if (out == 'y' || out == 'Y') {

                            lout = true;

                            exit();

                            system("cls");

                        }

                        break;

                    default:

                        printf("Error:错误命令\n");

                        cout << "请输入要执行的操作：";

                        break;

                }

                if (lout)

                    break;

            }

        }

    }

    printf("完成\n");

    return 0;

}