上海大学

SHANGHAI UNIVERSITY

操作系统（二）实验四报告

组　　号	第 9 组
姓　　名	蔡卓悦
学　　号	18120482
实 验 序 号	实验四
日　　期	2020 年 1 月 7 日

一、 实验目的与要求

1 实验目的

随着社会信息量的极大增长，要求计算机处理的信息与日俱增，涉及到社会生活的各个方面。因此，文件 管理是操作系统的一个极为重要的组成部分。学生应独立地用高级语言编写和调试一个简单的文件系统，模拟 文件管理的工作过程。从而对各种文件操作命令的实质内容和执行过程有比较深入的了解，掌握它们的实施方 法，加深理解课堂上讲授过的知识。

2 实验要求

⑴实际一个 n 个用户的文件系统，每个用户最多可保存 m 个文件。

⑵限制用户在一次运行中只能打开 1 个文件。

⑶系统应能检查命令的正确性，出错要能显示出错原因。

⑷对文件必须设置保护措施，如只能执行，允许读、允许写等。在每次打开文件时，根据本次打开的要求，再次设置保护级别，即可有二级保护。

⑸对文件的操作至少应有下述几条命令：

create 建立文件

delete 删除文件

open 打开文件

close 关闭文件

read 读文件

write 写文件

二、实验环境

本实验操作系统为 macOS 操作系统，使用电脑为 MacBook Pro，本实验的 IDE 是苹果官方的 Xcode 软件。

三、实验内容及其设计与实现

1、实验题目

(1)设计一个 10 个用户的文件系统，每个用户最多可保存 10 个文件，一次运行中用户可打开 5 个文件。

(2)程序采用二级文件目录，即设置了主文件目录（MFD）和用户文件目录（UFD）。前者应包含文件主（即用户）及他们的目录区指针；后者应给出每个文件主占有的文件目录，即文件名，保护码，文件长度 以及他们存放的位置等。另外为打开文件设置了运行文件目录（AFD），在文件打开时应填入打开文件号，本次打开保护码和读写指针等。

(3)为了便于实现，对文件的读写作了简化，在执行读写命令时，只修改读写指针，并不进行实际文件的读写操作。

2、实验总体思路

(1)因系统小，文件目录的检索使用了简单的线性搜索，而没有采用 Hash 等有效算法。

(2)文件保护简单实用了三位保护码，对应于允许读、允许写和运行执行，如下所示：

1 1 1

允许写 允许读 允许执行

如对应位为 0，则不允许。

(3)程序中使用的主要数据结构如下：

①主文件目录和用户文件目录

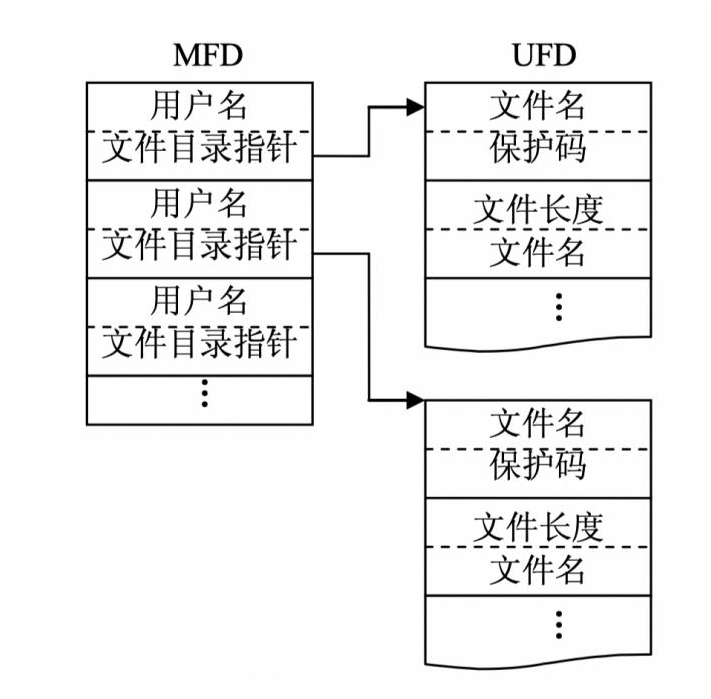

图 1 主文件目录 MFD 和用户文件目录 UFD

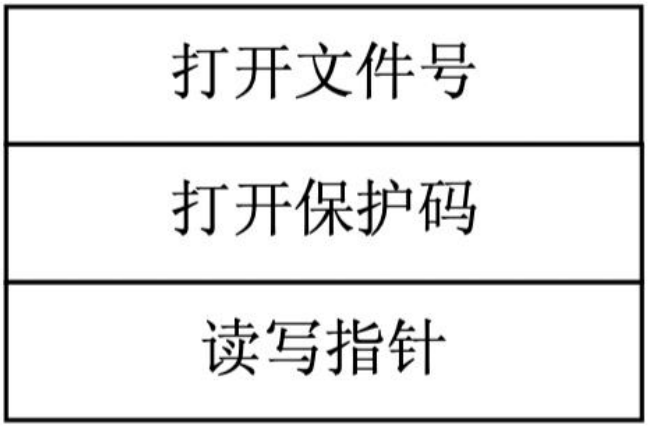

图 2 打开文件目录

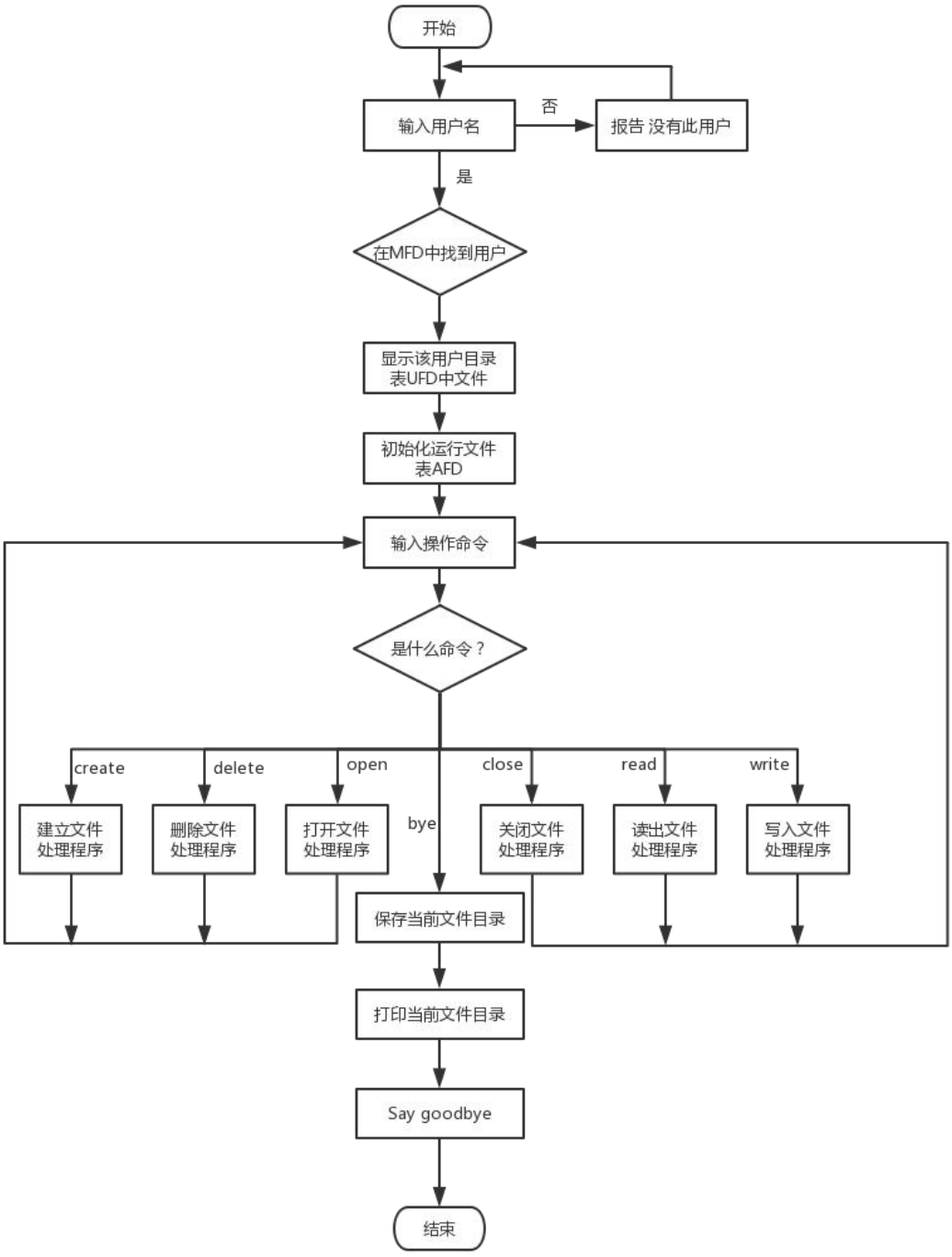
开始
输入用户名
否
报告 没有此用户
是
在MFD中找到用户
显示该用户目录
表UFD中文件
初始化运行文件
表AFD
输入操作命令
是什么命令？
create
delete
open
close
read
write
bye
建立文件
处理程序
删除文件
处理程序
打开文件
处理程序
关闭文件
处理程序
读出文件
处理程序
写入文件
处理程序
保存当前文件目录
打印当前文件目录
Say goodbye
结束

图 3 文件系统整体框图

四、实验结果

1.新建用户并登陆

如图：创建了 caizhuoyue 用户并登陆 caizhuoyue 用户。

```
Pleae input command:  adduser
 Please input the username: caizhuoyue
 Created!
Pleae input command:  login
 Please input the username: caizhuoyue
 Login successfully
```

图 4 新建用户并登陆

2.查看文件

由于是首次进入系统，系统中并没有文件。

```
Pleae input command:  ls
 Nothing!
```

图 5 查看文件

3.创建文件操作并设置保护码

如下图, 创建了 file1 和 file2 文件。为了后续验证操作, 将它们的保护码分别设置为 111 和 100.

```
Pleae input command:  create
 Please input the file's name: file1
 The new file is created!
 Enter the open mode:    111
Pleae input command:  create
 Please input the file's name: file2
 The new file is created!
 Enter the open mode:    100
Pleae input command:  ls
 filename   mode    length
 file1      111   0
 file2      100   0
```

图 6 创建文件 file1 和 file2

同时，同一用户的文件中不能有同名文件，为了检测同用户的新建文件重名检测，我在 caizhuoyue 用户下创建两个名为 file1 的文件。在第二次创建文件时系统提示“file1 already exists!”，文件创建失败，证明文件重名检测系统工作正常。

```
Pleae input command:  create
 Please input the file's name: file1
 The new file is created!
 Enter the open mode:    111
Pleae input command:  create
 Please input the file's name: file1
 The filename already exists!
```

图 7 不能创建同名文件

而在不同用户的文件中，文件可以重名。下图我创建了两个用户 u1 和 u2。为了验证是否可以在两个用户的目录下创建同名文件，我在这里分别登陆用户 u1 和 u2，并分别创建文件。

```
Pleae input command:  adduser
 Please input the username: u1
 Created!
Pleae input command:  adduser
 Please input the username: u2
 Created!
Pleae input command:  login
 Please input the username: u1
 Login successfully
Pleae input command:  create
 Please input the file's name: file1
 The new file is created!
 Enter the open mode:    111
Pleae input command:  ls
 filename   mode    length
 file1      111   0
Pleae input command:  create
 Please input the file's name: file1
 The filename already exists!
Pleae input command:  logout
 Bye!
Pleae input command:  login
 Please input the username: u2
 Login successfully
Pleae input command:  create
 Please input the file's name: file1
 The new file is created!
```

图 8 可以在不同用户下创建同名文件

4.打开文件操作

从下图可以知道，不能重复打开同一个文件。

```
Pleae input command:  open
 Pleae input the file's name: file1
 The file is opened.It's open num is 0.
Pleae input command:  open
 Pleae input the file's name: file1
 The file has opened!
Pleae input command:  read
 Please input the file's open num: 0
 The file length is 0
Pleae input command:  write
 Please input the file's open num: 0
 Plese input the length: 100
Pleae input command:  read
 Please input the file's open num: 0
 The file length is 100
```

图 9 打开文件操作

5.修改保护码操作

在这里我们可以看出，在文件打开时不能修改保护码。在关闭文件后我们将 file1 的保护码修改为了 000.

```
Pleae input command:  ls
 filename   mode    length
 file1      111   100
 file2      100   0
Pleae input command:  change
 Please input the file's name: file1
 This file is occupied and the protectCode cannot be  change!
Pleae input command:  close
 Please input the file's open num: 0
Pleae input command:  change
 Please input the file's name: file1
 Enter the open mode:    000
It changed!
```

图 10 修改文件 file1 的保护码

在修改后，我试图去读、写、打开 file1，来验证修改后的保护码是否生效。如下图所示，既不能读写也不能打开文件 file1，证明 file1 的保护码被成功修改为了 000.

```
Pleae input command:  ls
 filename   mode    length
 file1      000   100
 file2      100   0
Pleae input command:  open
 Pleae input the file's name: file1
 The file is opened.It's open num is 0.
Pleae input command:  write
 Please input the file's open num: 0
 The file is unwritable!
Pleae input command:  read
 Please input the file's open num: 0
 The file is unreadable!
```

图 11 验证修改后的保护码是否为 000

7.删除文件操作

如下图可知，不能删除打开的文件。需要先关闭文件再删除。

```
Pleae input command:  delete
 Please input the file's name: file1
 This file is occupied and cannot be deleted
Pleae input command:  close
 Please input the file's open num: 0
Pleae input command:  delete
 Please input the file's name: file1
Deleted File!
Pleae input command:  ls
 filename   mode    length
 file2      100   0
```

图 12 删除文件 file1

8.删除用户操作

如下图，登陆的状态的用户不能被删除，要先退出登录再删除。

```
Pleae input command:  deleteuser
 Illegal operation! Please logout!
Pleae input command:  logout
 Bye!
Pleae input command:  deleteuser
 Please input the username: caizhuoyue
Deleted!
```

图 13 删除用户 caizhuoyue

六、收获与体会

本次实验四是方老师重点验收的第二个实验。是关于文件系统的和用户管理的。

在该实验中，对于我们自己设计的文件系统有多种多样的要求，比如说要管理用户可以存放的文件最大个数、每个用户每次可以打开的文件个数等等。我利用 c++语言编写了代码，完成了实验的基本要求。同时我还根据指导书的要求与指导，实现了对于文件的 6 个不同操作的 c++代码。

实验四不仅我对于操作系统中文件系统、用户管理模块的理解，也巩固了我的 c++语言的类和数据结构的知识。这个初步的系统虽然并不完美，但是对于我的操作系统学习意义重大。

附录1: 文件系统实现代码

```
//
//  main.cpp
//  操作系统实验四
//
//  Created by 蔡卓悦 on 2021/1/15.
//

#include <iostream>
#include <stdio.h>
#include <string>
#include <cstring>

#define NUM 10
#define MAX_OPEN_NUM 5

using namespace std;


struct MFD_Entry{//主文件目录 目录项
    string UserName;                // 用户名
    struct UFD_Entry *fp;            // UFD 的指针——用于来模拟目录指针
};

struct UFD_Entry{//用户文件目录 目录项
    string FileName;                // 文件名
    int length;                        // 文件长度。用更改文件长度，模拟修改文件。
    bool protectCode[3];            // 文件保护码
    bool isWork;
};

class FileManage
{
private:
    MFD_Entry MFD[NUM];                        // MFD 数组，用于存储用户名
    UFD_Entry *UFD;                            // UFD 指针
    UFD_Entry *openFiles[MAX_OPEN_NUM];        // 打开文件后的，文件指针。（这里简化，直接用 UFD 来模
拟）
    int MFD_Length;                            // MFD 目录使用的数量
    int Openfile_Length;                    // 打开文件的数量
    bool isLogin;                            // 标记用户登陆状态
public:

    void init();                            // 初始化函数
    void help();                            // 帮助函数

    void adduser();                            // 添加用户函数
    void deleteuser();                        // 删除用户函数

    void login();                            // 用户登陆函数
```

```
    void logout();                              // 用户登出函数

    void createfile();                          // 新建文件函数
    void deletefile();                          // 删除文件函数

    void openfile();                           // 打开文件函数
    void closefile();                           // 关闭文件函数

    void readfile();                           // 读文件函数
    void writefile();                           // 写文件函数
    void changeprotectCode();                   // 修改保护码

    void lsfunc();                              // 查看文件

};


//指令字符串数组
char const *commands_str[] = {
    "login",
    "logout",
    "adduser",
    "deleteuser",
    "create",
    "delete",
    "open",
    "close",
    "read",
    "write",
    "ls",
    "change",
    "help"
};


//函数指针数组
void (FileManage::*commands_func[])() = {
    &FileManage::login,
    &FileManage::logout,
    &FileManage::adduser,
    &FileManage::deleteuser,
    &FileManage::createfile,
    &FileManage::deletefile,
    &FileManage::openfile,
    &FileManage::closefile,
    &FileManage::readfile,
    &FileManage::writefile,
    &FileManage::lsfunc,
    &FileManage::changeprotectCode,
    &FileManage::help};
```

```
void FileManage::init()
{
    MFD_Length = 0;
    Openfile_Length = 0;
    UFD = NULL;
    isLogin = false;
    for(int i = 0 ;i < NUM; i++)
    {
        MFD[i].UserName = "";
        MFD[i].fp = NULL;
    }

    for(int i = 0; i < MAX_OPEN_NUM; i++)
    {
        openFiles[i] = NULL;
    }
}

void FileManage::adduser()
{
    if(isLogin)
    {
        cout << " Illegal operation! Please log in!" << endl;
        return;
    }
    string name;//用户名
    cout << " Please input the username: ";
    cin >> name;
    // 用于判断是否重名的标志
    bool flag;
    // 用户没有到达10个用户的上限 即主目录表项数目没有到10
    if(MFD_Length >= 0 && MFD_Length <= 10)
    {
        // 遍历MFD 判断用户名是否重名
        for(int i = 0; i < NUM; i++)
        {
            flag = 0;
            if(MFD[i].UserName.compare(name) == 0)
            {
                cout << " The username already exists!" << endl;
                break;
            }
            flag = 1;
        }

        if(flag == 1) //没有重名
        {
            // 创建新的用户
            for(int i = 0; i < NUM; i++)
            {
                if(MFD[i].UserName == "")
                {
```

```
                    MFD[i].UserName = name;
                    // 创建，并初始化
                    UFD_Entry *temp;
                    temp = new UFD_Entry[10];
                    // 创建 11 个项 其中第一个项用于储存一些基本信息，不提供用户使用，属于系统层面。
                    for(int j = 0; j <= NUM; j++)
                    {
                        temp[j].FileName = "*********";
                        temp[j].length = 0;
                        temp[j].isWork = false;
                        for(int k = 0; k <3; k++)
                        {
                            temp[j].protectCode[k] = 0;
                        }
                    }

                    MFD[i].fp = temp;
                    cout << " Created!" << endl;
                    MFD_Length++;
                    break;
                }
            }
        }

    }
}

void FileManage::deleteuser()
{
    // 在用户登陆后删除用户操作，不合法，应该禁止
    if(isLogin)
    {
        cout << " Illegal operation! Please logout!" << endl;
        return;
    }
    string name;
    cout << " Please input the username: ";
    cin >> name;

    // 用户判断
    if(MFD_Length > 0 && MFD_Length <= 10)
    {
        // 遍历MFD 寻找指定用户
        for(int i = 0; i < NUM; i++)
        {
            if(MFD[i].UserName.compare(name) == 0)
            {
                MFD[i].UserName = "";
                UFD_Entry *temp = MFD[i].fp;

                delete[] temp;
                MFD[i].fp = NULL;
```

```
                MFD_Length--;

                cout << "Deleted!" << endl;
                break;
            }
        }
    }
}

void FileManage::login()
{
    if(isLogin)//重复登录
    {
        cout << " Illegal operation! You already logged in!" << endl;
        return;
    }

    string inBuf, temp;
    temp = " Unsuccessfully!  The usernamse not exist! ";
    cout << " Please input the username: ";
    cin >> inBuf;

    // 在 MFD 中查询用户
    for(int i = 0; i < NUM; i++)
    {
        if(MFD[i].UserName.compare(inBuf) == 0)
        {
            // 获取用户目录的目录指针
            UFD = MFD[i].fp;
            isLogin = true;
            temp = "successfully";
            break;
        }
    }

    cout << " Login " << temp << endl;;
}

void FileManage::logout()
{
    if(!isLogin)//没有登陆
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }
    UFD = NULL;
    isLogin = false;
    cout << " Bye!" << endl;
}
```

```
void FileManage::createfile()//创建文件
{

    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }

    string inBuf;
    bool flag;
    cout << " Please input the file's name: " ;
    cin >> inBuf;

    if(UFD[0].length >= 0 && UFD[0].length <=10)
    {
        // 判断文件是否重名
        for(int i = 1; i <= NUM; i++)
        {
            flag = 0;
            if(UFD[i].FileName.compare(inBuf) == 0)
            {
                cout << " The filename already exists!" << endl;
                break;
            }
            flag = 1;
        }

        if(flag == 1)
        {
            for(int i = 1; i <= NUM; i++)
            {
                // 找到空闲区域，创建文件
                if(UFD[i].FileName.compare("*********") == 0)
                {
                    // 辨别系统保留字符串
                    if(inBuf == "*********")
                    {
                        cout << "Please change the filename!" << endl;
                        return;
                    }
                    UFD[i].FileName = inBuf;
                    cout << " The new file is created!" << endl;

                    cout << " Enter the open mode:    ";
                    cin >> inBuf;
                    for(int j =0; j <3;j++)
                    {
                        if(inBuf[j] == '0')
                        {
                            UFD[i].protectCode[j] = false;
                        }
```

```
                        else {
                            UFD[i].protectCode[j] = true;
                        }
                    }
                    UFD[0].length++;
                    break;
                }
            }

        }

    }
    else {
        cout << "No free space!" << endl;
    }

}


void FileManage::deletefile()
{
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }
    string inBuf, temp;
    temp = " The file not found!";
    cout << " Please input the file's name: " ;
    cin >> inBuf;

    // 用户判断
    if(MFD_Length > 0 && MFD_Length <= 10)
    {
        // 遍历，寻找指定文件
        for(int i = 1; i <= NUM; i++)
        {
            if(UFD[i].FileName.compare(inBuf) == 0)
            {
                // 检查文件是否占用
                if(UFD[i].isWork == true)
                {
                    cout << " This file is occupied and cannot be deleted " << endl;
                    return;
                }

                UFD[i].FileName = "*********";
                UFD[i].length = 0;
                UFD[0].length--;

                temp = "Deleted File! ";
                break;
```

```
            }
        }
        cout << temp << endl;
    }
    else {
        cout << "error!" << endl;
    }
}

void FileManage::openfile()
{
    // 用户登陆后，才能进行操作。
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }
    string inBuf;
    bool isFind = false;
    cout << " Pleae input the file's name: ";
    cin >> inBuf;

    if(UFD[0].length > 0 && UFD[0].length <= 10)
    {
        // 遍历，寻找指定文件
        for(int i = 1; i <= NUM; i++)
        {
            if(UFD[i].FileName.compare(inBuf) == 0)
            {
                if(UFD[i].isWork == true)
                {
                    cout << " The file has opened!"<< endl;
                    return;
                }
                // 判断打开文件是否已经到达上限
                if(Openfile_Length>=0 && Openfile_Length < MAX_OPEN_NUM)
                {
                    for(int j = 0; j < MAX_OPEN_NUM; j++)
                    {
                        // 寻找空文件指针
                        if(openFiles[j] == NULL)
                        {
                            UFD[i].isWork = true;
                            // 获取文件指针
                            openFiles[j] = &UFD[i];
                            cout << " The file is opened.It's open num is " << j << "." <<
endl;

                            isFind = true;
                            break;
                        }
                    }
```

```
                    if(isFind == true)
                    {
                        break;
                    }
                }
                else {
                    cout <<  " openFiles error!" << endl;
                }

            }
        }
        if(isFind == false)
        {
            cout << " The file not found!" << endl;
        }
    }
    else
    {
        cout << " error!" << endl;
    }
}


void FileManage::closefile()
{
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }
    int inBuf;
    cout << " Please input the file's open num: ";
    cin >> inBuf;

    // open num 判断有效性, 避免指针误操作
    if(inBuf>=0 && inBuf < 5)
    {
        // 解除文件占用
        openFiles[inBuf]->isWork = false;
        openFiles[inBuf] = NULL;
    }
}


void FileManage::readfile()
{
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }
```

```
    int inBuf;
    cout << " Please input the file's open num: ";
    cin >> inBuf;

    // open num 判断有效性，避免指针误操作
    if(inBuf>=0 && inBuf < 5)
    {
        if(openFiles[inBuf] != NULL)
        {
            if(openFiles[inBuf]->protectCode[1] == 1)
            {
                cout << " The file length is " << openFiles[inBuf]->length << endl;
            }
            else
            {
                cout << " The file is unreadable!" << endl;
            }
        }
        else
        {
            cout << " File pointer is invalid!" << endl;
        }
    }
}

void FileManage::writefile()
{
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }

    int inBuf1, inBuf2;
    cout << " Please input the file's open num: ";
    cin >> inBuf1;

    // open num 判断有效性，避免指针误操作
    if(inBuf1>=0 && inBuf1 < 5)
    {
        if(openFiles[inBuf1] != NULL)
        {
            if(openFiles[inBuf1]->protectCode[0] == 1)
            {
                cout << " Plese input the length: ";
                cin >> inBuf2;
                if(inBuf2 >= 0)
                {
                    openFiles[inBuf1]->length = inBuf2;
                }
                else
                {
```

```
                    cout << " Input error!" << endl;
                }
            }
            else
            {
                cout << " The file is unwritable!" << endl;
            }
        }
        else
        {
            cout << " File pointer is invalid!" << endl;
        }
    }
}

void FileManage::lsfunc()
{
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }

    if(UFD[0].length == 0)
    {
        cout << " Nothing!" << endl;
        return;
    }
    else
    {
        cout << " filename\tmode\tlength" << endl;
    }
    for(int i = 0; i < NUM; i++)
    {
        if(UFD[i].FileName.compare("*********") != 0)
        {
            cout << " " <<UFD[i].FileName << "\t\t" << UFD[i].protectCode[0] <<
UFD[i].protectCode[1] << UFD[i].protectCode[2] << "\t  " << UFD[i].length << endl;
        }
    }
}


void FileManage::changeprotectCode()
{
    if(!isLogin)
    {
        cout << " Illegal operation! Please login!" << endl;
        return;
    }

    string inBuf;
```

```
    bool isfind=false;
    cout << " Please input the file's name: ";
    cin >> inBuf;

    // 用户判断
    if(MFD_Length > 0 && MFD_Length <= 10)
    {
        // 遍历，寻找指定文件
        for(int i = 1; i <= NUM; i++)
        {
            if(UFD[i].FileName.compare(inBuf) == 0)
            {
                isfind = true;

                // 判断更改的文件是否被占用
                if(UFD[i].isWork == true)
                {
                    cout << " This file is occupied and the protectCode cannot be  change!" <<
endl;

                    return;
                }

                cout << " Enter the open mode:    ";
                cin >> inBuf;
                for(int j =0; j <3;j++)
                {
                    if(inBuf[j] == '0')
                    {
                        UFD[i].protectCode[j] = false;
                    }
                    else {
                        UFD[i].protectCode[j] = true;
                    }
                }
                cout << "It changed!" << endl;
                break;
            }
        }
        if(isfind == false)
        {
            cout << "The file can't find!" << endl;
        }
    }
    else {
        cout << "error!" << endl;
    }
}

void FileManage::help()
{
    cout << " These shell commands are defined internally." << endl;
```

```
    cout << " create, delete, open, close, read, write, adduser, deleteuser, ls, change" <<
endl;
}

int main(int argc, char *argv[]) {
    FileManage filemanage;
    filemanage.init();
    cout << "Hello, you can input 'help' to find the command!" << endl;
    while(1)
    {
        string commandBuf;
        cout << "Pleae input command:  ";
        cin >> commandBuf;

        int len = (sizeof(commands_str) / sizeof(commands_str[0])); //求字符串数组长度
        bool isFind = false;

        for (int i = 0; i < len; i++)
        {
            if (!strcmp(commandBuf.data(), commands_str[i]))
            {
                (filemanage.*commands_func[i])();
                isFind = true;
                break;
            }
        }

        if (!isFind) // 未找到命令
        {
            cout << "command " <<commandBuf << " is not found!" << endl;
        }

    }

    return 0;
}
```