

文件系统实验

小组信息

小组成员

姓名	学号	学院
臧祝利	202011998088	人工智能学院
陈金利	202011998145	人工智能学院
孙泽林	202011998122	人工智能学院
赵建锟	202011998073	人工智能学院

成员分工

臧祝利：

- 增加文件模拟磁盘功能
- 增加新建用户功能
- 增加命令提示符前路径功能
- “程序错误” 部分+Bug①⑤
- 程序总流程图+adduser+exit

陈金利：

- 增加password修改密码功能
- 解决Bug③⑥
- 流程图mkdir+mkfile+write & read

孙泽林：

- 增加查看用户信息功能
- 解决Bug②⑦
- 流程图password+del

赵建锟：

- 增加help功能
- 解决Bug④⑧
- 流程图dir+cd+help

实验内容

实验环境

IDE：Visual Studio 2022 Current

C++语言标准：C++14

绘图工具：BoardMix

程序错误

由于使用的是Visual Studio 2022，有的函数已经被废弃，因此在对源代码进行运行时会出现问题，主要内容如下：

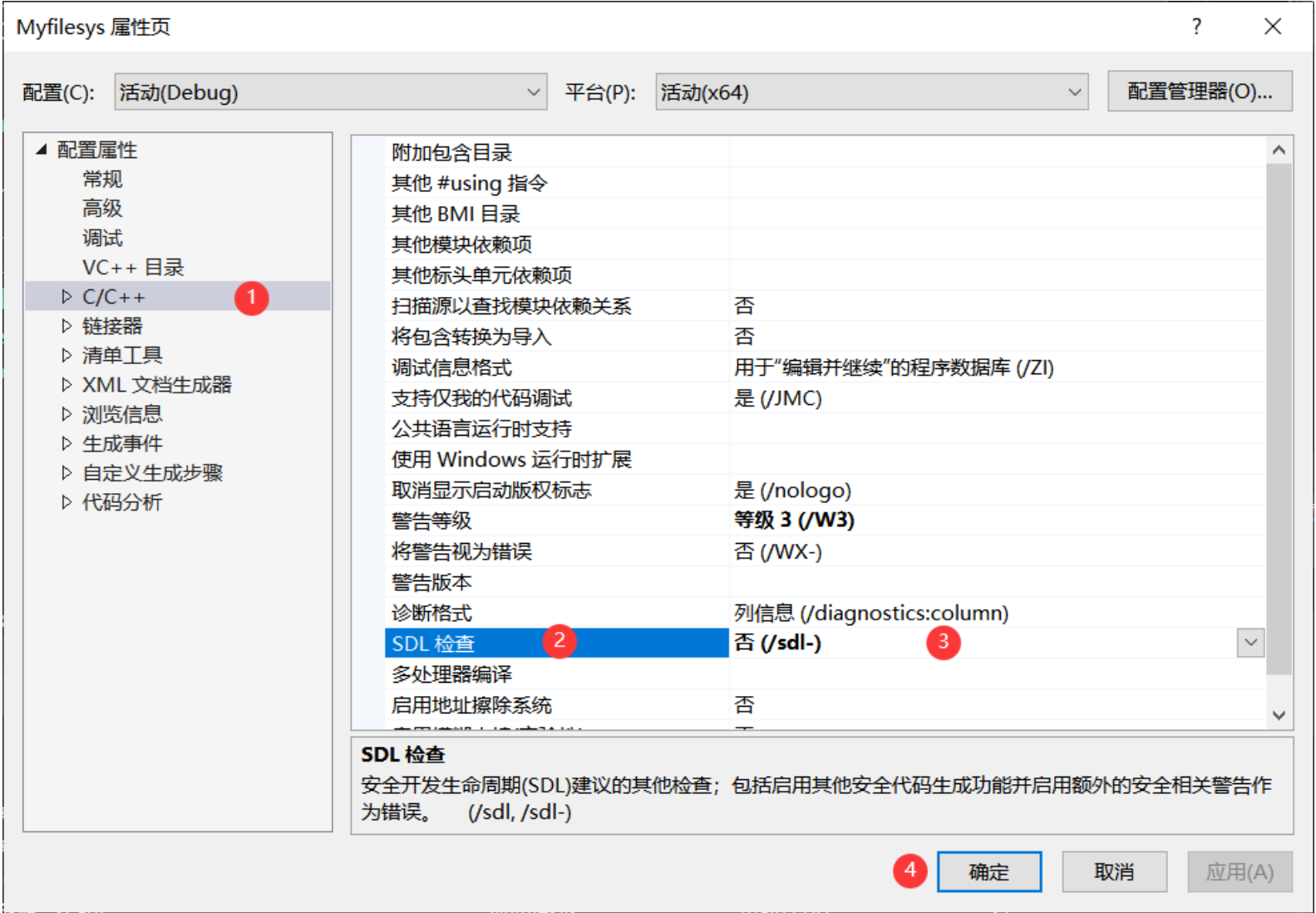
① `gets()` 函数：显示“未定义标识符”；

原因：`gets()` 函数在C++14中被删除，因此无法使用

解决方案：改用 `gets_s()` 函数

② `scanf()` 函数：`scanf`的返回值被忽略；

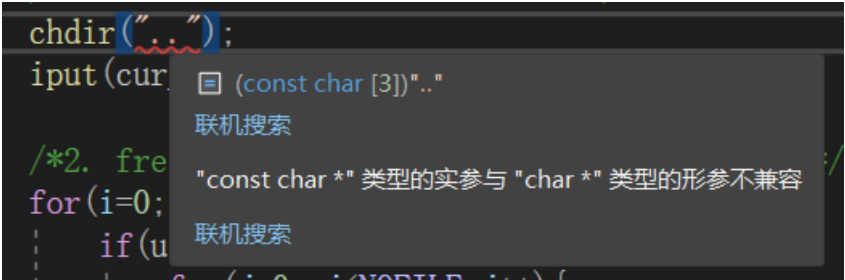
解决方案：更改项目属性。将“SDL检查选项改为[否]”；



也可以都改用 `scanf_s()` 函数或者在代码开头添加

```
#define _CRT_SECURE_NO_WARNINGS
```

③参数不兼容报错；



把 `halt.cpp` 中路径代码更改为:

```
char s[] = "..";
char* ss = s;
chdir(s);
```

Bug修复

①程序运行开始就越界崩溃



报错原因：内存越界；

在 `format.cpp` 中

```
for (i=5; i<PWDNUM; i++) {
    passwd[i].p_uid = 0;
    passwd[i].p_gid = 0;
    strcpy(passwd[i].password, "          "); // PWDSIZ " "
}

memcpy(pwd, passwd, 32*sizeof(struct passwd));
memcpy(disk+DATASTART+BLOCKSIZ*2, passwd, BLOCKSIZ);
iput(inode);
```

解决方案：减少 `strcpy()` 函数中的第二个参数，即减少一个空格，避免其缓冲区溢出，即可解决问题；

修改后，不再出现问题：

```
for (i=5; i<PWDNUM; i++) {
    passwd[i].p_uid = 0;
    passwd[i].p_gid = 0;
    strcpy(passwd[i].password, "        "); // PWDSIZ " "
}

memcpy(pwd, passwd, 32*sizeof(struct passwd));
memcpy(disk+DATASTART+BLOCKSIZ*2, passwd, BLOCKSIZ);
iput(inode);
```

②创建一个与目录同名的文件时，系统崩溃。

如图，登录后建立目录 `dir1`，再建立文件时报错，报错内容为“`dir1 is not a file`”，崩溃；

```
Welcome to mini filesystem!
Login:2116
Password:dddd
> > mkdir dir1
> dir

CURRENT DIRECTORY :..
当前共有4个文件/目录
..          xxxxxxxxx i_ino->1      <dir>
.           xxxxxxxxx i_ino->1      <dir>
etc         xxxxxxxxx i_ino->2      <dir>
dir1        xxxxxxxxx i_ino->4      <dir>
> mkfile dir1
存在同名目录!
dir1 is not a file!!!
```

报错原因：经过 `Debug` 可知，使用 `mkfile` 命令会调用 `creat()` 函数，而 `creat()` 函数的返回值为-1,由语句

```
return open(user_id,filename,WRITE);
```

返回-1，在 `shell.cpp` 中，由变量 `fd` 接收结果

```
fd = creat(user_id,tstr,mode);
if(fd == -1){
    printf("创建文件失败! \n");
    break;
}
```

但是定义时将 `fd` 设置为 `unsigned int` 类型，无法接收负数，因此出现错误；

解决方案：

如果判断其不是文件类型，可以直接 `return -1`，而不是通过 `open()` 函数返回，即直接添加一句`return -1;`

```
if(!(inode->di_mode&DIFILE)){ //如果不是文件
    printf("存在同名目录! \n");
    return -1;
}
```

同时将 `fd` 更改为 `int` 类型；

运行结果 如下:

```
> > mkdir dir
> mkfile dir
存在同名目录!
创建文件失败!
> _
```

③重复创建文件目录时提示错误;

重复创建同一个文件时出错, 错误如下:

```
> mkdir a
> mkdir a
a是一个文件!
> mkfile a
存在同名目录!
a不是一个文件!
创建文件失败!
> mkfile b
> mkfile b
> mkdir b
目录b已存在!
```

错误原因:

dir.cpp 中 mkdir() 函数中的 iget() 函数传参错误, 应当传入的参数是目录项对应内存结点i的目录号, 即 dir.direct[dirid].d_ino

```
dirid= namei(dirname);
if (dirid != -1){
    inode = iget(dirid);
    if (inode->di_mode & DIDIR)
        printf("目录%s已存在! \n", dirname); //xiao
    else
        printf("%s是一个文件! \n", dirname);
    iput(inode);
    return;
}
```

解决方案:

更改 iget 的参数;

```
dirid= namei(dirname);
if (dirid != -1){
    inode = iget(dir.direct[dirid].d_ino);
    if (inode->di_mode & DIDIR)
        printf("目录%s已存在! \n", dirname); //xiao
    else
        printf("%s是一个文件! 目录创建失败! \n", dirname);
    iput(inode);
    return;
}
```

修改 creat.cpp , 增加文件存在语句, 并根据是否覆盖选择输出文件是否创建失败;

```
dirid = namei(filename);
if (dirid != -1){//如果存在同名文件/目录
    inode = iget(dir.direct[dirid].d_ino);
    if(!(inode->di_mode&DIFILE)){//如果不是文件
        printf("存在同名目录! \n");
        return -1;
    }
    else {
        printf("文件%s已存在! \n", filename); //新加
        printf("是否选择覆盖此文件? [Y/N]");
        char ans[2];
        scanf("%s", ans);
        if (ans[0] == 'N') return -1; //不覆盖, 则创建失败!
    }
}
```

运行结果 如下:

```
> > mkdir a
> mkdir a
目录a已存在!
> mkfile b
> mkfile b
文件b已存在!
是否选择覆盖此文件? [Y/N]N
创建文件失败!
> > mkdir b
b是一个文件! 目录创建失败!
```

④**Dir**命令，显示当前目录一直是“..”，请修改为正确的当前路径；

错误原因：

dir.cpp 中的 _dir() 函数中输出如下:

```
printf("\n CURRENT DIRECTORY :%s\n",dir.direct[0].d_name);
printf("当前共有%d个文件/目录\n",dir.size);
```

`dir.direct[0].d_name` 一直为存储的是上一级目录`..`，因此输出一直没有改变：

解决方案：

在创建目录时把当前目录变为正确的路径即可;

```
strcpy(buf[0].d_name, ".."); //子目录的上一层目录 当前目录
buf[0].d_ino = cur_path_inode->i_ino;
memcpy(buf[1].d_name, dirname, DIRSIZ); //把当前目录存在d_name中
buf[1].d_ino = inode->i_ino; //子目录的本目录 子目录
```

同时更改输出

```
printf("\n CURRENT DIRECTORY :%s\n",dir.direct[1].d_name);
```

运行结果如下：

```
> > mkdir a
> dir

CURRENT DIRECTORY :.
当前共有4个文件/目录
..                xxxxxxxxxx i_ino->1          <dir>
.                 xxxxxxxxxx i_ino->1          <dir>
etc               xxxxxxxxxx i_ino->2          <dir>
a                 xxxxxxxxxx i_ino->4          <dir>
> cd a
> mkdir b
> dir

CURRENT DIRECTORY :a
当前共有3个文件/目录
..                xxxxxxxxxx i_ino->1          <dir>
a                 xxxxxxxxxx i_ino->4          <dir>
b                 xxxxxxxxxx i_ino->5          <dir>
> cd b
> dir

CURRENT DIRECTORY :b
当前共有2个文件/目录
..                xxxxxxxxxx i_ino->4          <dir>
b                 xxxxxxxxxx i_ino->5          <dir>
>
```

⑤ 在根目录下面创建子目录**a**，在**a**中创建文件**b**，并写入大于一个块(**512字节**)的内容，之后返回根目录，查看目录内容的时候会出现错误。

```
> mkdir a
> cd a
> mkfile b
> write b 2000
2000 字节已经被写入文件 b.
> dir

当前目录 :. 其inode编号: 4
当前共有3个文件/目录
..          xxxxxxxx i_ino->1          <dir>
.           xxxxxxxx i_ino->4          <dir>
b           xxxxxxxx i_ino->5          2000 block chain: 4 5 6 7
> cd ..
> dir

当前目录 :电电电电电电电电电电电电电电电电电电电电电电电电 其inode编号: -842150451
当前共有4个文件/目录
电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电电
```

错误原因：

在写入时，会调用 `write()` 函数，`write()` 函数在写入的时候传入的地址出现错误；错误地址为文件开始地址加上指针所在块号与块的乘积，但是正确的地址是文件开始地址加上 **即将写入的块号** 与块大小的乘积；

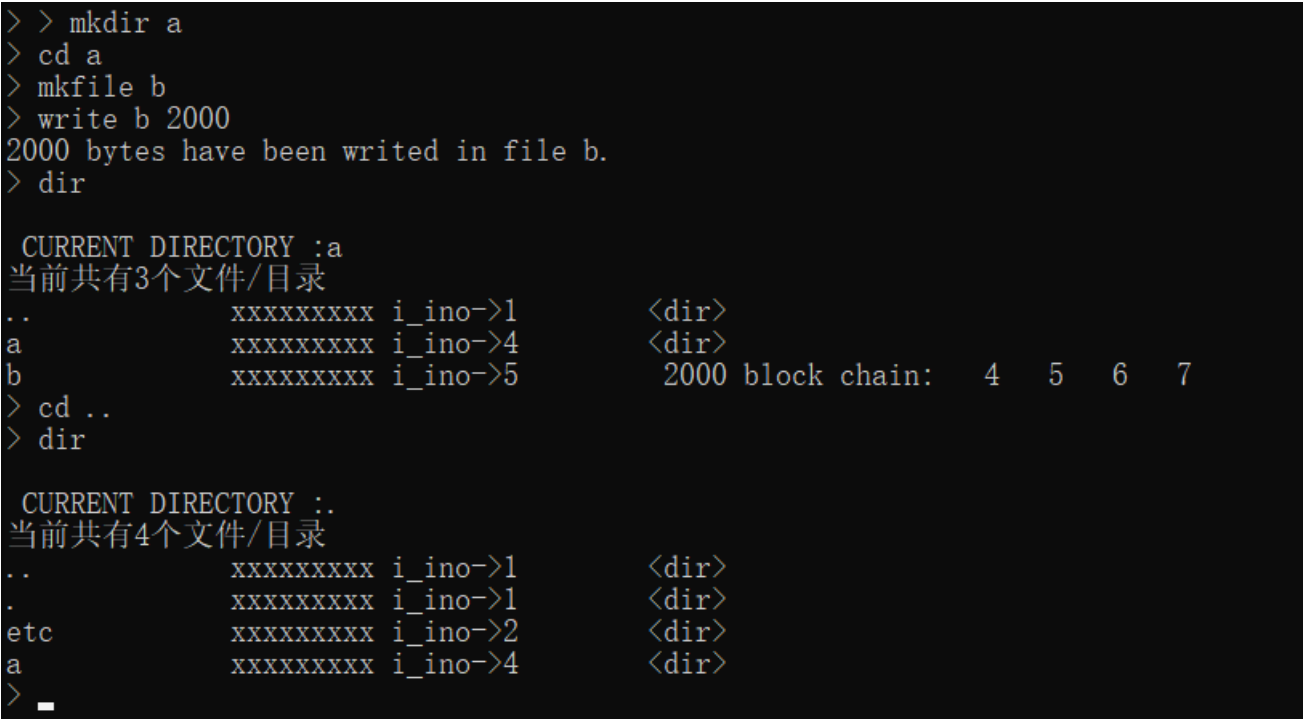
解决方案：

更改代码，由

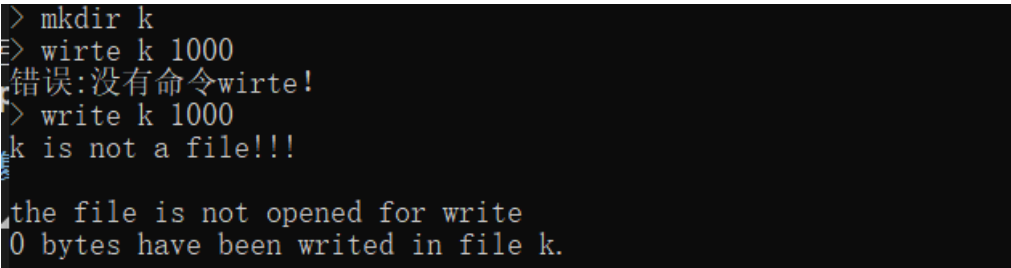
```
memcpy(disk+DATASTART+block*BLOCKSIZ, temp_buf, block_off);
```

更改为

```
memcpy(disk+DATASTART+inode->di_addr[block+i+1]*BLOCKSIZ, temp_buf, block_off);
```



⑥Read或Write一个不存在的文件时，程序会崩溃



错误原因：

使用 `read` 和 `write` 指令时，都会调用 `open()` 函数，`open()` 当中会使用 `namei()` 函数寻找编号，由于文件不存在，因此编号返回-1，但是其定义的 `dinodeid` 为 `unsigned` 类型，因此语句

```
if (dinodeid == 0){
    printf("\nfile does not existed!!!\n");
    return -1;
}
```

理论上不会执行，因此不会输出 `file does not existed!!!` 并return -1;

返回-1之后，应该在 `shell.cpp` 中增加文件不存在的分支结构，以结束此命令；

解决方案：

更改 `dinodeid` 为 `int` 类型，并修改条件为 `if (dinodeid == -1)`；

在 `shell.cpp` 中增加文件不存在时候的判断：

```
fd = open(user_id, tstr, char(mode));
if (fd == -1) {
    printf("unable to write!\n");
    break;
}
```



```
fd = open(user_id,tstr,READ);
if (fd == -1) {
    printf("unable to read\n");
    break;
}
```

运行结果 如下：

```
> > write k 100
file does not existed!!!
unable to write!
> read kk 20
file does not existed!!!
unable to read
> _
```

⑦磁盘回收后再分配会出错；

错误原因：

执行 del 命令时，调用 delete() 函数，并使用 iput() 函数回收内存块；

iput() 函数回收内存块的时候存在两处错误：

- 如果删除的文件大小并非512的整数倍，最后的一个块没有被回收
- 删除的顺序有错误，与栈不符合；

```
else
{
    /* 删除磁盘i结点和文件对应的物理块 */
    block_num = pinode->di_size/BLOCKSIZ; 1
    for (i=0; i<block_num; i++) 2
        bfree(pinode->di_addr[i]);
    ifree(pinode->i_ino);
}
```

解决方案：

- 增加剩余物理块判断
- 倒序删除块

```
else
{
    /* 删除磁盘i结点和文件对应的物理块 */
    block_num = pinode->di_size/BLOCKSIZ;
    if (pinode->di_size % BLOCKSIZ) { //剩余块判断
        block_num++;
    }
    for (i=block_num-1; i>=0; i--) //逆序
        bfree(pinode->di_addr[i]);
    ifree(pinode->i_ino);
}
```

运行结果 如下：

```

CURRENT DIRECTORY :test
当前共有7个文件/目录
..          xxxxxxxxxx i_ino->1      <dir>
test        xxxxxxxxxx i_ino->4      <dir>
a           xxxxxxxxxx i_ino->5      2000 block chain:  4   5   6   7
b           xxxxxxxxxx i_ino->6      3000 block chain:  8   9  10  11  12  13
c           xxxxxxxxxx i_ino->7      4000 block chain: 14  15  16  17  18  19  20  21
d           xxxxxxxxxx i_ino->8      5000 block chain: 22  23  24  25  26  27  28  29  30  31
e           xxxxxxxxxx i_ino->9      0 block chain:
> deil b
错误:没有命令deil!
> del b
> dir

CURRENT DIRECTORY :test
当前共有6个文件/目录
..          xxxxxxxxxx i_ino->1      <dir>
test        xxxxxxxxxx i_ino->4      <dir>
a           xxxxxxxxxx i_ino->5      2000 block chain:  4   5   6   7
c           xxxxxxxxxx i_ino->7      4000 block chain: 14  15  16  17  18  19  20  21
d           xxxxxxxxxx i_ino->8      5000 block chain: 22  23  24  25  26  27  28  29  30  31
e           xxxxxxxxxx i_ino->9      0 block chain:
> mkfile f
> write e 4000
4000 bytes have been written in file e.
> dir

CURRENT DIRECTORY :test
当前共有7个文件/目录
..          xxxxxxxxxx i_ino->1      <dir>
test        xxxxxxxxxx i_ino->4      <dir>
a           xxxxxxxxxx i_ino->5      2000 block chain:  4   5   6   7
f           xxxxxxxxxx i_ino->6      0 block chain:
c           xxxxxxxxxx i_ino->7      4000 block chain: 14  15  16  17  18  19  20  21
d           xxxxxxxxxx i_ino->8      5000 block chain: 22  23  24  25  26  27  28  29  30  31
e           xxxxxxxxxx i_ino->9      4000 block chain:  8   9  10  11  12  13  32  33
>

```

⑧其他错误

(1) 输入 `exit` , 报错:

```

> exit
no such a file
Good Bye. See You Next Time. Please turn off the switch

```

错误原因: 在 `main.cpp` 中, 使用的是 `logout(2118)` 函数, 但是用户名是2116, 因此会输出这句话;

解决方案: 更改为 `logout(username)`

(2) 一开始会重复输出">";

```

Login:2116
Password:dddd
> > dir

CURRENT DIRECTORY :.
当前共有3个文件/目录
..          xxxxxxxxxx i_ino->1      <dir>
.           xxxxxxxxxx i_ino->1      <dir>
etc         xxxxxxxxxx i_ino->2      <dir>
>

```

错误原因:

```

do{
    printf("> ");
    fflush(stdin);
    gets_s(str);
}while(shell(user_id, str));

```

解决方案: 将 `fflush(stdin)` 更改为 `rewind(stdin)` 即可;

```

Login:2116
Password:dddd
> mkdir a
> dir

```


新增加功能

文件模拟磁盘

在每次退出前，使用文件将磁盘信息保存下来，因此在 `halt()` 函数中增加向文件中写入磁盘的功能；同时为了区分用户，因此更改 `halt()` 函数，使得对于每一个 `username` 都会生成对应的 `username.txt` 存储；

在 `halt()` 函数的最后，使用如下代码：

```
/*5. say GOOD BYE to all the user*/
printf("\nGood Bye. See You Next Time. Please turn off the switch\n");
char txtname[100]; //文件名称
itoa(username, txtname, 10); //将数字转换成字符串
strcat(txtname, ".txt"); //生成文件后缀
char filename[100] = "user/"; //增加目录名
strcat(filename, txtname); //完整文件路径
FILE* fp;
fp = fopen(filename, "wb"); //打开文件
fwrite(disk, sizeof(char), (DINODEBLK + FILEBLK + 2) * BLOCKSIZ, fp); //写入磁盘
fclose(fp);
exit(0);
```

在 `main.cpp` 中，增加选项——>用户是进行上一次的操作还是重新操作；

```
do {
    printf("Please enter your choice!\n");
    printf("[0] New Filesys; [1] Last Filesys\n");
    scanf("%d", &op);
    if (op == 0) {
        format();
    }
    else if (op == 1) {
        fp = fopen(filename, "rb");
        memset(disk, 0x00, ((DINODEBLK + FILEBLK + 2) * BLOCKSIZ));
        fread(disk, sizeof(char), (DINODEBLK + FILEBLK + 2) * BLOCKSIZ, fp);
        memcpy(pwd, disk + DATASTART + BLOCKSIZ * 2, BLOCKSIZ);
        fclose(fp);
    }
    else {
        printf("Please enter the choice again!\n");
    }
} while (op != 0 && op != 1);
```

为了找到对应的文件，多加了一步，用于让用户确定自己本次操作的用户名称；

```
printf("Enter the username you will operator! ,make sure you won't change it when you log in\n");
scanf("%d", &username);
char txtname[100];
itoa(username, txtname, 10);
strcat(txtname, ".txt");
char filename[100] = "user/";
strcat(filename, txtname);
```

同时修改了 `install()` 函数的内容；

```
memcpy(&dir.direct[(BLOCKSIZ)/(DIRSIZ+4)*i],
disk + DATASTART + BLOCKSIZ * cur_path_inode->di_addr[i], (dir.size * (DIRSIZ + 4)) % BLOCKSIZ);
```

效果如下：

首次创建：此时为2116

```
debug x64 本地 Windows 调试器
Microsoft Visual Studio 调试控制台
Welcome to mini filesystem!
Enter the username you will operator! ,make sure you won't change it when you log in
2116
Please enter your choice!
[0] New Filesys; [1] Last Filesys
0
Login:2116
Password:dddd
> > mkdir a
> mkdir b
> mkdir c
> cd a
> mkdir d
> dir

CURRENT DIRECTORY :a
当前共有3个文件/目录
. .          xxxxxxxxx i_ino->1          <dir>
a            xxxxxxxxx i_ino->4          <dir>
d            xxxxxxxxx i_ino->7          <dir>
> cd ..
> dir

CURRENT DIRECTORY :.
当前共有6个文件/目录
. .          xxxxxxxxx i_ino->1          <dir>
.            xxxxxxxxx i_ino->1          <dir>
etc          xxxxxxxxx i_ino->2          <dir>
a            xxxxxxxxx i_ino->4          <dir>
b            xxxxxxxxx i_ino->5          <dir>
c            xxxxxxxxx i_ino->6          <dir>
> exit

Good Bye. See You Next Time. Please turn off the switch
```

二次调用:

```
Welcome to mini filesystem!
Enter the username you will operator! ,make sure you won't change it when you log in
2116
Please enter your choice!
[0] New Filesys; [1] Last Filesys
1
Login:2116
Password:dddd
> > dir

CURRENT DIRECTORY :.
当前共有6个文件/目录
. .          xxxxxxxxx i_ino->1          <dir>
.            xxxxxxxxx i_ino->1          <dir>
etc          xxxxxxxxx i_ino->2          <dir>
a            xxxxxxxxx i_ino->4          <dir>
b            xxxxxxxxx i_ino->5          <dir>
c            xxxxxxxxx i_ino->6          <dir>
> cd a
> dir

CURRENT DIRECTORY :a
当前共有3个文件/目录
. .          xxxxxxxxx i_ino->1          <dir>
a            xxxxxxxxx i_ino->4          <dir>
d            xxxxxxxxx i_ino->7          <dir>
> _
```

换用2117登录时, 如果选择1, 会进行以下判断, 判断是否可以取用上一次的结果;

```
else if (op == 1) {
    fp = fopen(filename, "rb");
    if (fp == NULL) {
        printf("此用户本次为第一次操作! 无法获取上一次操作\n");
        format();
        break;
    }
    memset(disk, 0x00, ((DINODEBLK + FILEBLK + 2) * BLOCKSIZ));
    fread(disk, sizeof(char), (DINODEBLK + FILEBLK + 2) * BLOCKSIZ, fp);
    memcpy(pwd, disk + DATASTART + BLOCKSIZ * 2, BLOCKSIZ);
    fclose(fp);
}
```

运行结果如下:

```
Welcome to mini filesystem!
Enter the username you will operator! ,make sure you won't change it when you log in
2117
Please enter your choice!
[0] New Filesys; [1] Last Filesys
1
此用户本次为第一次操作！无法获取上一次操作
Login:2117
Password:bbbb
> > dir

CURRENT DIRECTORY :.
当前共有3个文件/目录
..          xxxxxxxxx i_ino->1      <dir>
.           xxxxxxxxx i_ino->1      <dir>
etc         xxxxxxxxx i_ino->2      <dir>
>
```

新建用户

想法是使用文件来存储用户信息；

首先的五个用户已经存在，暂时不需要更改；需要文件用来

- 存储当前使用的用户个数
- 存储用户的id号、登录密码、所属用户组信息；

创建文件 `user.cpp` 和函数 `adduser()`：

```
void adduser(int username,char* pswd,int group) {
    if (username == 2116 || username == 2117 || username == 2118 || username == 2119 || username == 2120) {
        printf("Sorry ,the user existed\n");
        return;
    }
    usernumber = getusernumber(); //自己写的函数，返回当前的用户个数
    for (int i = 0;i < usernumber;i++) {
        if (username == pwd[i].p_uid) {
            printf("Sorry ,the user existed\n"); //判断
            return;
        }
    }
    if (usernumber == 32) {
        printf("用户数量已达上限(32)!\n");
        return;
    }
    char txtname[100]; //文件名称
    itoa(usernumber, txtname, 10); //将数字转换成字符串
    strcat(txtname, ".txt"); //生成文件后缀,例如5.txt,用于之后赋值pwd
    char filename[100] = "user_info/"; //增加目录名
    strcat(filename, txtname); //完整文件路径
    FILE* fp;
    fp = fopen(filename, "wb"); //打开文件
    fprintf(fp, "%d\n%s\n%d\n", username, pswd, group);
    fclose(fp);
    printf("用户创建成功!\n");
    usernumber++;
    storeusernumber(); //把更改后的信息存入
    return;
}
```

更改 `format.cpp` 中的 `format()` 函数，使之能从文件中读取所有用户信息；

```

usernumber = getusernumber();
for (i = 5; i < usernumber; i++) {
    int p_uid, p_gid, pswds;
    char pswd[100];

    char txtname[100]; //文件名称
    itoa(i, txtname, 10); //将数字转换成字符串
    strcat(txtname, ".txt"); //生成文件后缀, 例如5.txt, 用于之后赋值pwd
    char filename[100] = "user_info/"; //增加目录名
    strcat(filename, txtname); //完整文件路径
    FILE* fp;
    fp = fopen(filename, "r"); //打开文件
    fscanf(fp, "%d%d%d", &p_uid, &pswds, &p_gid);
    itoa(pswds, pswd, 10);
    passwd[i].p_uid = p_uid;
    passwd[i].p_gid = p_gid;
    strcpy(passwd[i].password, pswd);
}
for (i=usernumber; i<PWDNUM; i++){
    passwd[i].p_uid = 0;
}

```

shell.cpp 更改如下:

```

case 8: //adduser
    int id, group;
    char* ids, * groups;
    char* pswd;
    token = strtok(NULL, seps);
    ids = token;
    token = strtok(NULL, seps);
    pswd = token;
    token = strtok(NULL, seps);
    if (token == NULL) {
        printf("adduser 命令的正确格式为adduser username password group\n");
        break;
    }
    groups = token;
    id = atoi(ids);
    group = atoi(groups);
    //printf("%d %d\n", id, group);
    adduser(id, pswd, group);
    break;

```

创建新用户结果如下:

```

2116@filesys:/root> adduser 2121 1234 4
用户创建成功!
2116@filesys:/root> _

```

存储的文件信息如下:



再次登录时信息如下, 出现了 2121 用户;







```
2116@filesys:/root> who
id          pswd          group
2116        dddd          3
暂时输出一下全部成员的username :
2116
2117
2118
2119
2120
2121
2116@filesys:/root> _
```

可以直接用2121登录;

```
Login:2121
Password:1234
2121@filesys:/root>
```

修改密码

为了统一，删除 `format()` 中的初始化，改成将已有用户信息存储为文件；

 0.txt	2022/12/13 15:57	文本文档
 1.txt	2022/12/13 15:57	文本文档
 2.txt	2022/12/13 15:57	文本文档
 3.txt	2022/12/13 15:57	文本文档
 4.txt	2022/12/13 15:57	文本文档
 5.txt	2022/12/13 15:16	文本文档

加入 `modify_pswd()` 函数，根据用户名去寻找其对应的下标，然后将对应的文件进行修改；

```
void modify_pswd(int username, char* new_pswd) {
    usernumber = getusernumber();
    for (int i = 0;i < usernumber;i++) {
        if (username == pwd[i].p_uid) { //找到用户了,开始修改
            char pswd[100];

            char txtname[100]; //文件名称
            itoa(i, txtname, 10); //将数字转换成字符串
            strcat(txtname, ".txt"); //生成文件后缀,例如5.txt,用于之后赋值pwd
            char filename[100] = "user_info/"; //增加目录名
            strcat(filename, txtname); //完整文件路径
            FILE* fp;
            fp = fopen(filename, "wb"); //打开文件
            fprintf(fp, "%d\n%s\n%d\n", username, new_pswd, pwd[i].p_gid);
            fclose(fp);
            printf("修改成功!\n");
            return;
        }
    }
    printf("Not found the user!\n");
    return;
}
```

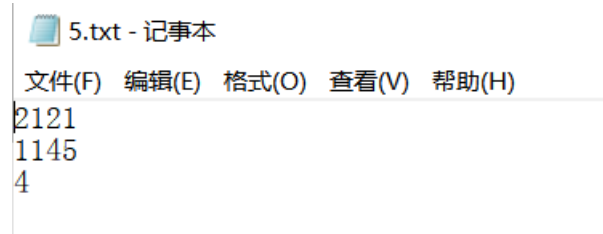
```
Login:2121
Password:1234
2121@filesys:/root> password 2121 1145
修改成功!
2121@filesys:/root>
```

重新使用此账号登录:

```
Login:2121
Password:1234

incorrect password
Login:2121
Password:1145
2121@filesys:/root>
```

查看对应的文本文档，发现也进行了改变：



5.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

2121
1145
4

查看当前用户信息

增加代码：

```
case 10:
    printf("id\t\tpswd\t\tgroup\n");
    printf("%d\t\t", pwd[user_id].p_uid);
    printf("%s\t\t", pwd[user_id].password);
    printf("%d\n", pwd[user_id].p_gid);
    break;
```

运行结果 如下：

```
Login:2116
Password:dddd
2116@filesys:/root> who
id          pswd          group
2116        dddd          3
2116@filesys:/root>
```

路径相关

在命令提示符前加入路径：

创建一个目录栈和栈指针，用于记录路径；

初始化：

```
point_address = 1;
for (int i = 0; i < 100; i++) {
    address_name[i] = (char*)malloc(sizeof(char) * 100);
}

strcpy(address_name[0], "root"); //初始目录命名为root
```

每次输出时，输出内容如下：

```
do{
    printf("%d@filesys:", user[user_id].u_uid);
    for (int i = 0; i < point_address; i++) {
        printf("/%s", address_name[i]);
    }
    printf("> ");
    rewind(stdin);
    gets_s(str);
}while(shell(user_id, str));
```

更改 dir.cpp 中的函数，将 _dir() 函数中的输出修改；


```
void _dir() {
    unsigned int di_mode;
    int i, j, k;           //xiao
    struct inode *temp_inode;

    //printf("\n CURRENT DIRECTORY :%s\n", dir.direct[1].d_name);
    printf("\n CURRENT DIRECTORY :%s\n", address_name[point_address-1]);
    printf("当前共有%d个文件/目录\n", dir.size);
    for (i=0; i<DIRNUM; i++) {
```

并修改 `chdir()` 函数的类型，由 `void` 修改到 `int`，以区分是否可以进入此目录；

```
    dirid = namei(dirname);
    if (dirid == -1) {
        printf("不存在目录%s! \n", dirname);
        return -1;
    }
    inode = iget(dir.direct[dirid].d_ino);
    if (!(inode->di_mode&DIDIR)) {
        printf("%s不是一个目录! \n");
        return -1;
    }
```

在 `shell.cpp` 中，更改如下：

```
case 3:
    token = strtok(NULL, seps);
    if(token == NULL) {
        printf("cd命令的正确格式为cd dirname, 请检查命令!\n");
        break;
    }
    if (strcmp(token, "..") && chdir(token) != -1) { //如果不是上级目录，且目录存在
        strcpy(address_name[point_address], token); //加入栈中
        point_address++;
    }
    else if (strcmp(token, "..") == 0) { //否则返回上一级目录
        point_address--;
    }
    break;
```

运行结果 如下：显示路径，路径不存在时，输出不存在目录，且可以通过 `cd ..` 返回上级目录并正确输出

```
2116@filesys:/root> mkdir a
2116@filesys:/root> cd a
2116@filesys:/root/a> cd ..
2116@filesys:/root> cd b
不存在目录b!
2116@filesys:/root> dir

CURRENT DIRECTORY :root
当前共有2个文件/目录
..            xxxxxxxxx i_ino->1      <dir>
a             xxxxxxxxx i_ino->4      <dir>
2116@filesys:/root> _
```

pwd命令

在 `shell.cpp` 中增加命令，增加的代码如下：

```
case 9: //pwd命令
    for (int i = 0; i < point_address - 1; i++) {
        printf("%s/", address_name[i]);
    }
    printf("%s\n", address_name[point_address - 1]);
    break;
```

运行结果 如下：

```
2116@filesys:/root> dir
CURRENT DIRECTORY :root
当前共有6个文件/目录
..          xxxxxxxxx i_ino->1      <dir>
.           xxxxxxxxx i_ino->1      <dir>
etc         xxxxxxxxx i_ino->2      <dir>
a           xxxxxxxxx i_ino->4      <dir>
b           xxxxxxxxx i_ino->5      <dir>
c           xxxxxxxxx i_ino->6      <dir>
2116@filesys:/root> cd a
2116@filesys:/root/a> pwd
root/a
2116@filesys:/root/a> mkdir aa
2116@filesys:/root/a> cd aa
2116@filesys:/root/a/aa> pwd
root/a/aa
2116@filesys:/root/a/aa>
```

help

输入help会显示所有的指令以及用途；

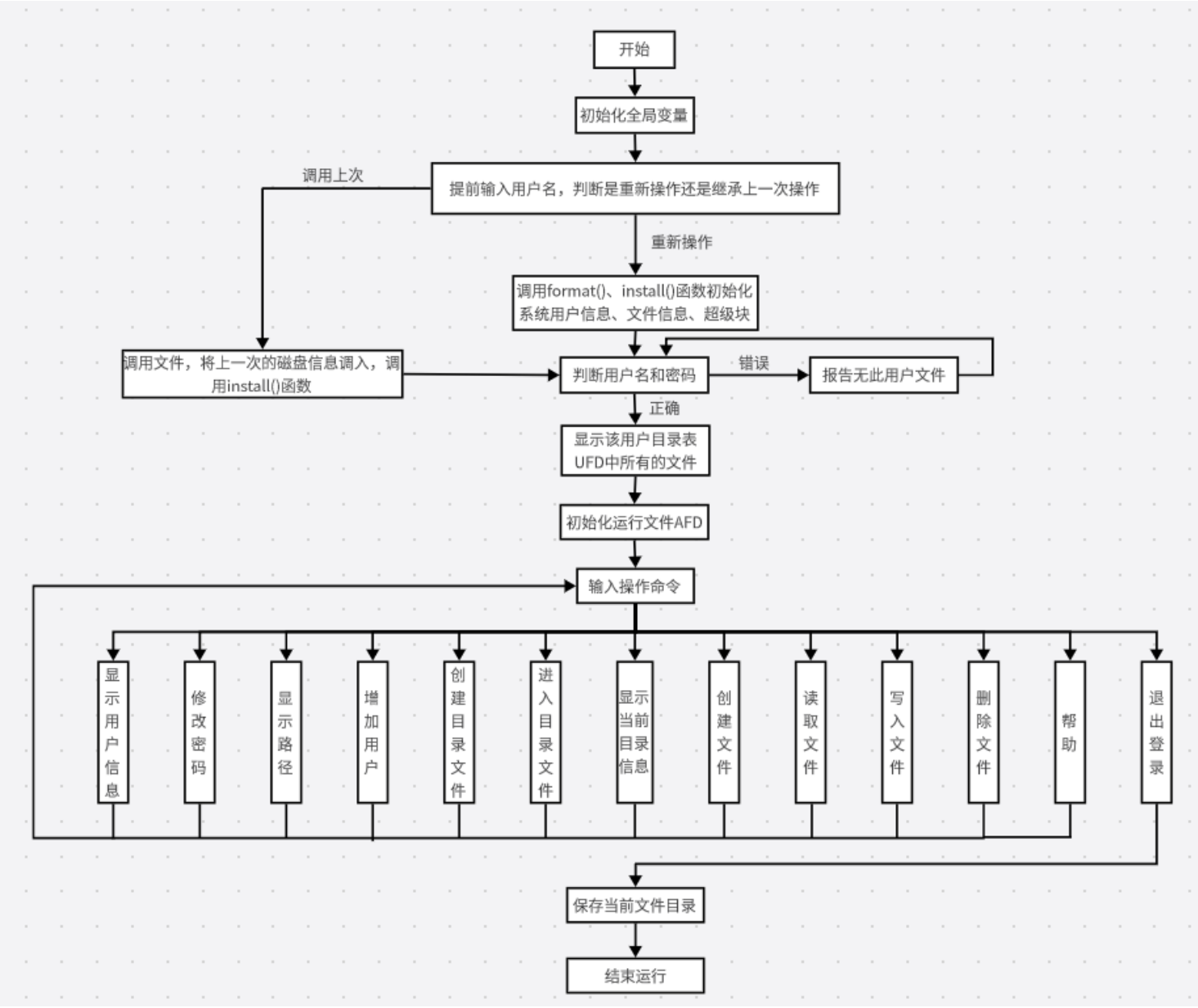
在help后键入具体指令会单独说明其用途；

```
case 12:
    token = strtok(NULL, seps);
    if (token == NULL) {
        printf("有关某个命令的详细信息，请键入 help 命令名\n");
        printf("exit\t\t退出程序并且将当前用户的磁盘存储到文件中\n");
        printf("dir\t\t显示当前目录\n");
        printf("mkdir\t\t创建新目录\n");
        printf("cd\t\t切换目录\n");
        printf("mkfile\t\t创建新文件\n");
        printf("del\t\t删除文件\n");
        printf("write\t\t写入文件\n");
        printf("read\t\t读取文件\n");
        printf("adduser\t\t加入新用户\n");
        printf("pwd\t\t显示当前路径\n");
        printf("who\t\t显示当前用户信息\n");
        printf("password\t\t修改用户密码\n");
        break;
    }
    helporder(token);
    break;
```

```
2121@filesys:/root> help
有关某个命令的详细信息，请键入 help 命令名
exit          退出程序并且将当前用户的磁盘存储到文件中
dir           显示当前目录
mkdir         创建新目录
cd            切换目录
mkfile        创建新文件
del           删除文件
write         写入文件
read          读取文件
adduser       加入新用户
pwd           显示当前路径
who           显示当前用户信息
password      修改用户密码
2121@filesys:/root> help adduser
增加新用户
adduser [username] [password] [group_id]
2121@filesys:/root> _
```

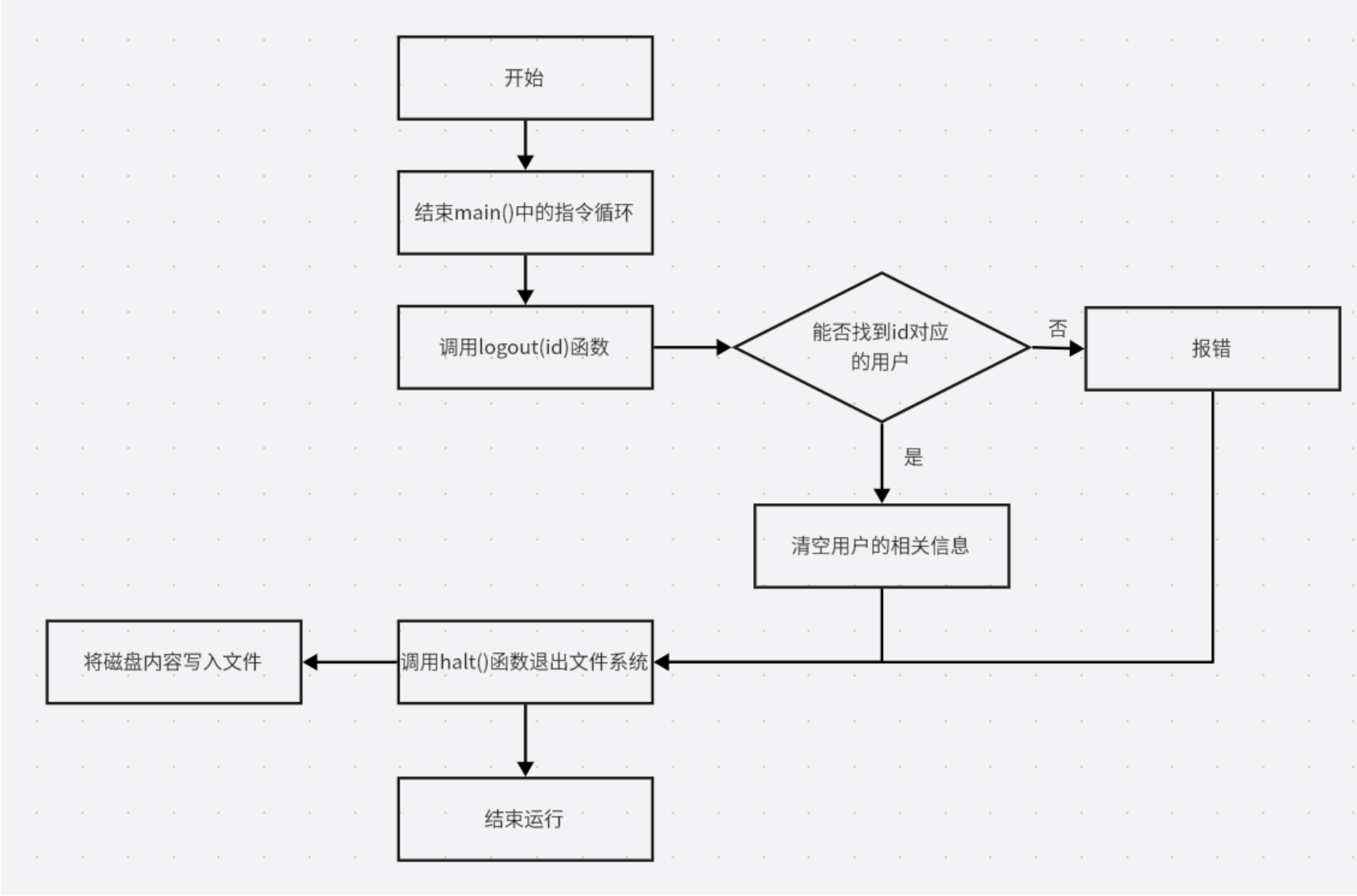
流程图

程序总流程图

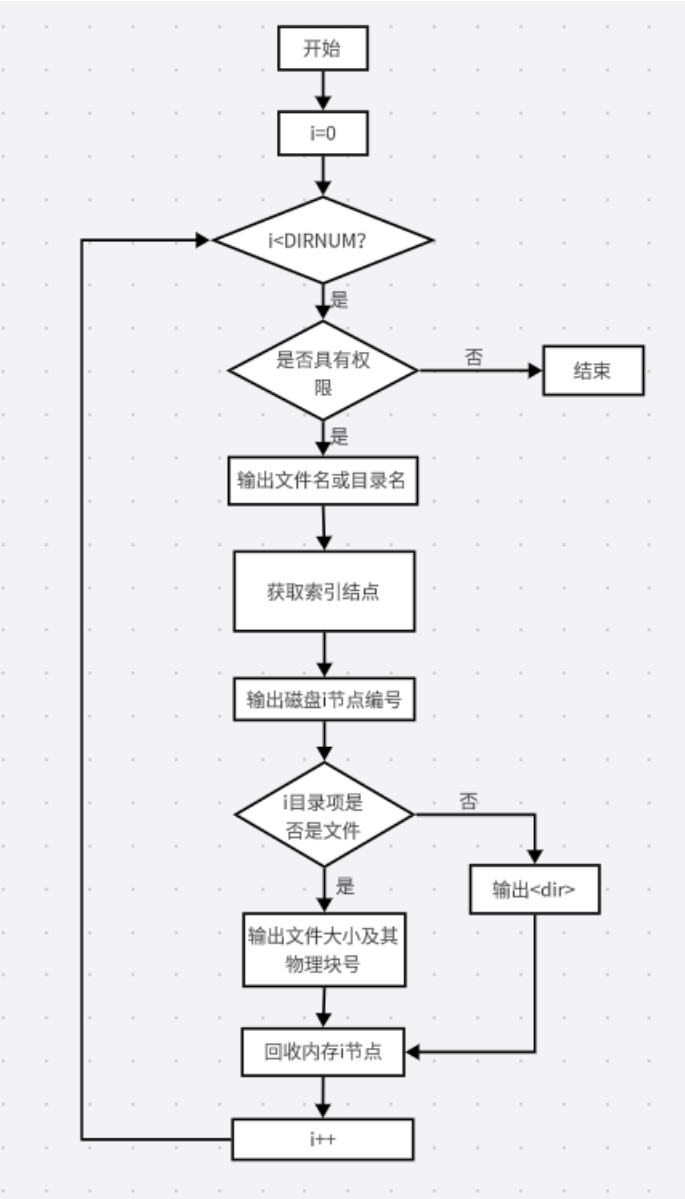


各模块流程图

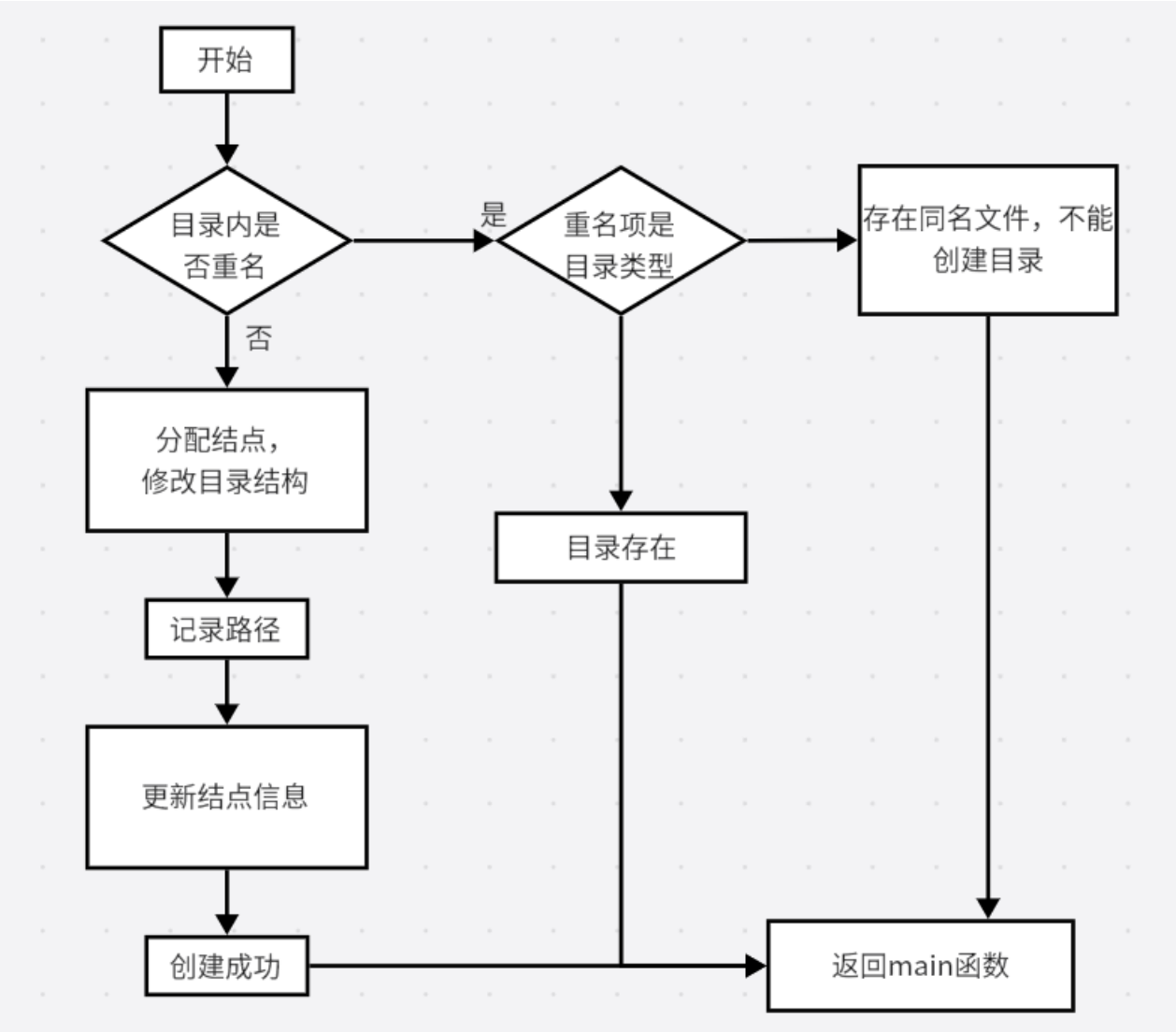
exit



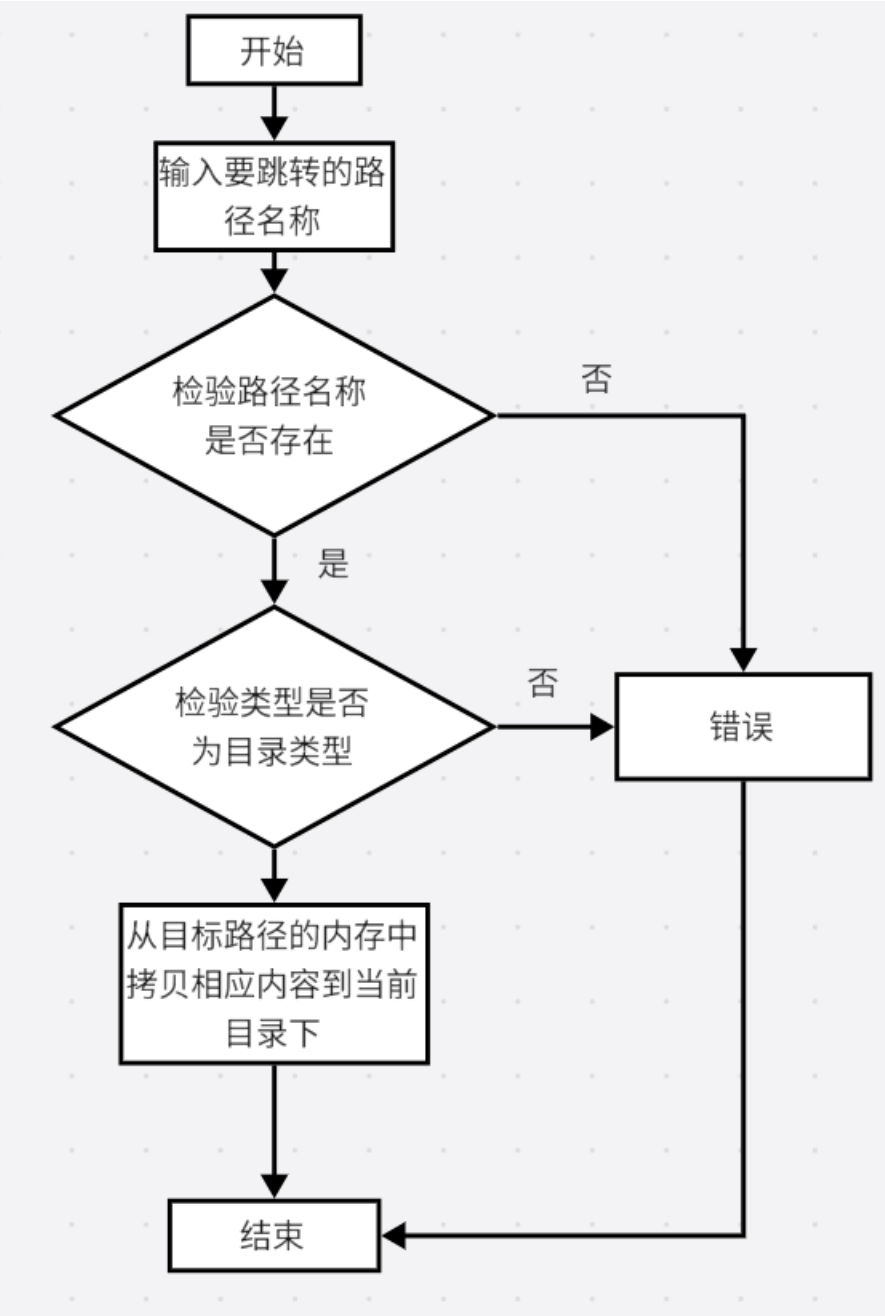
dir



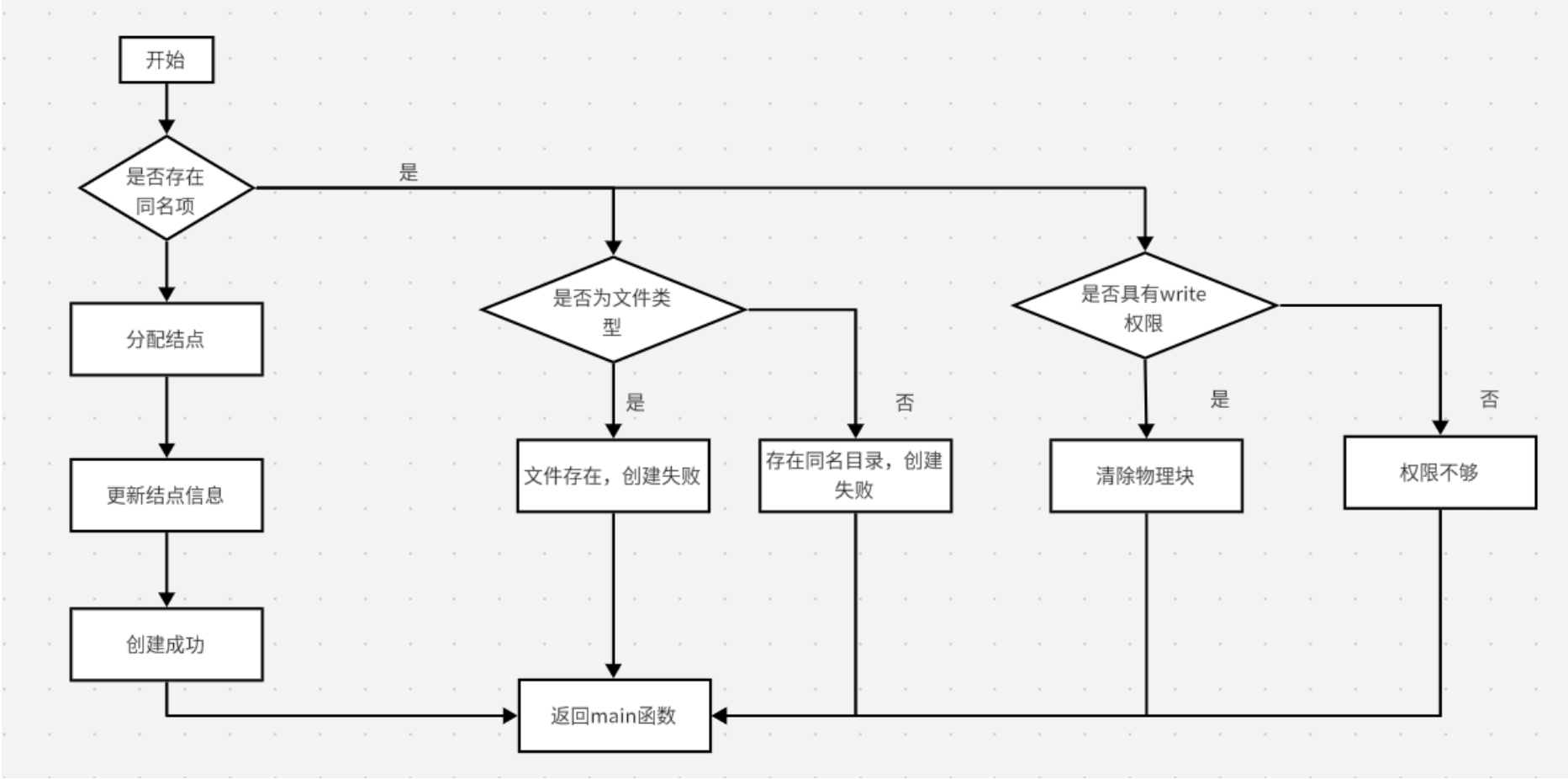
mkdir



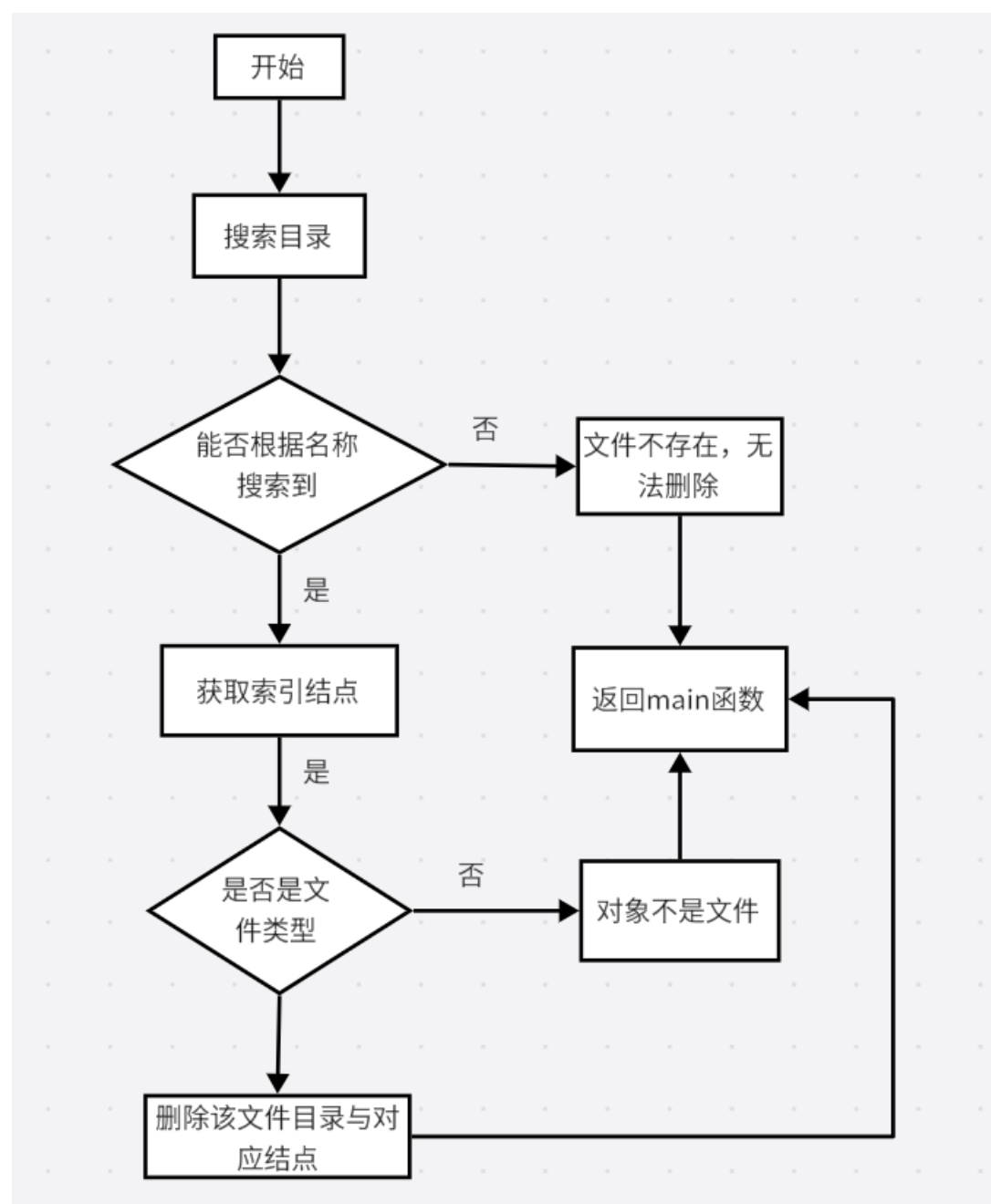
cd



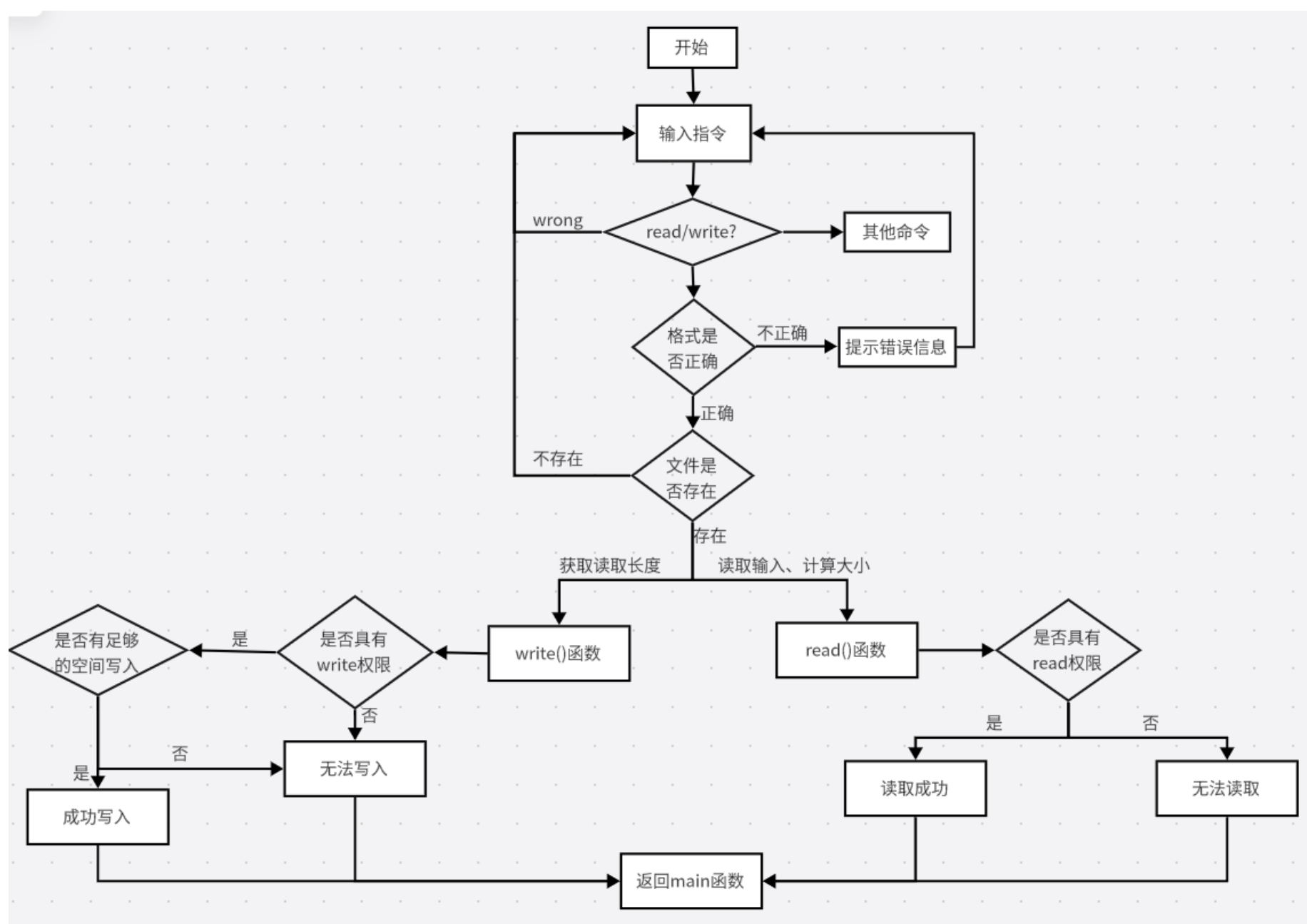
mkfile



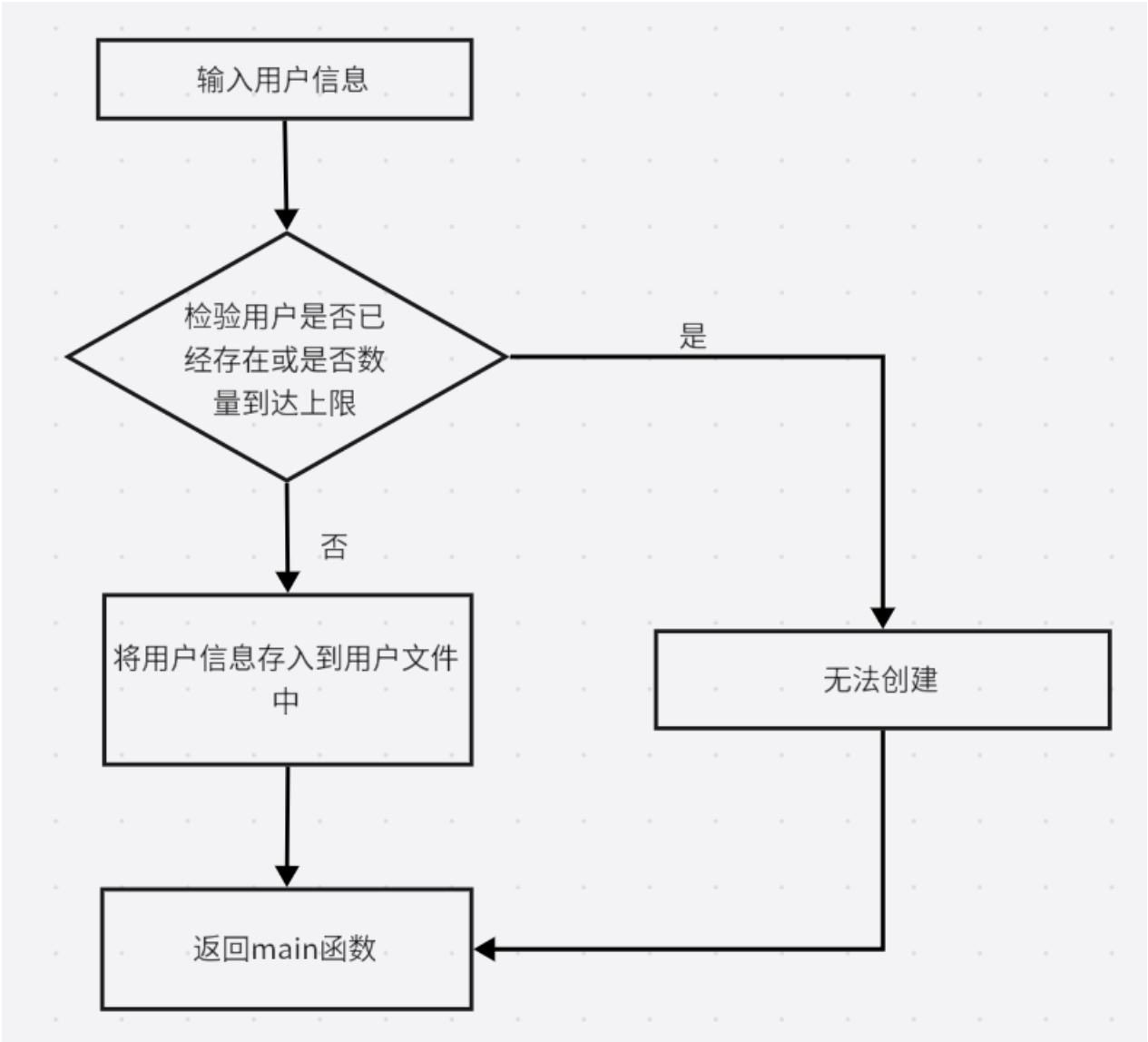
del



write & read



adduser



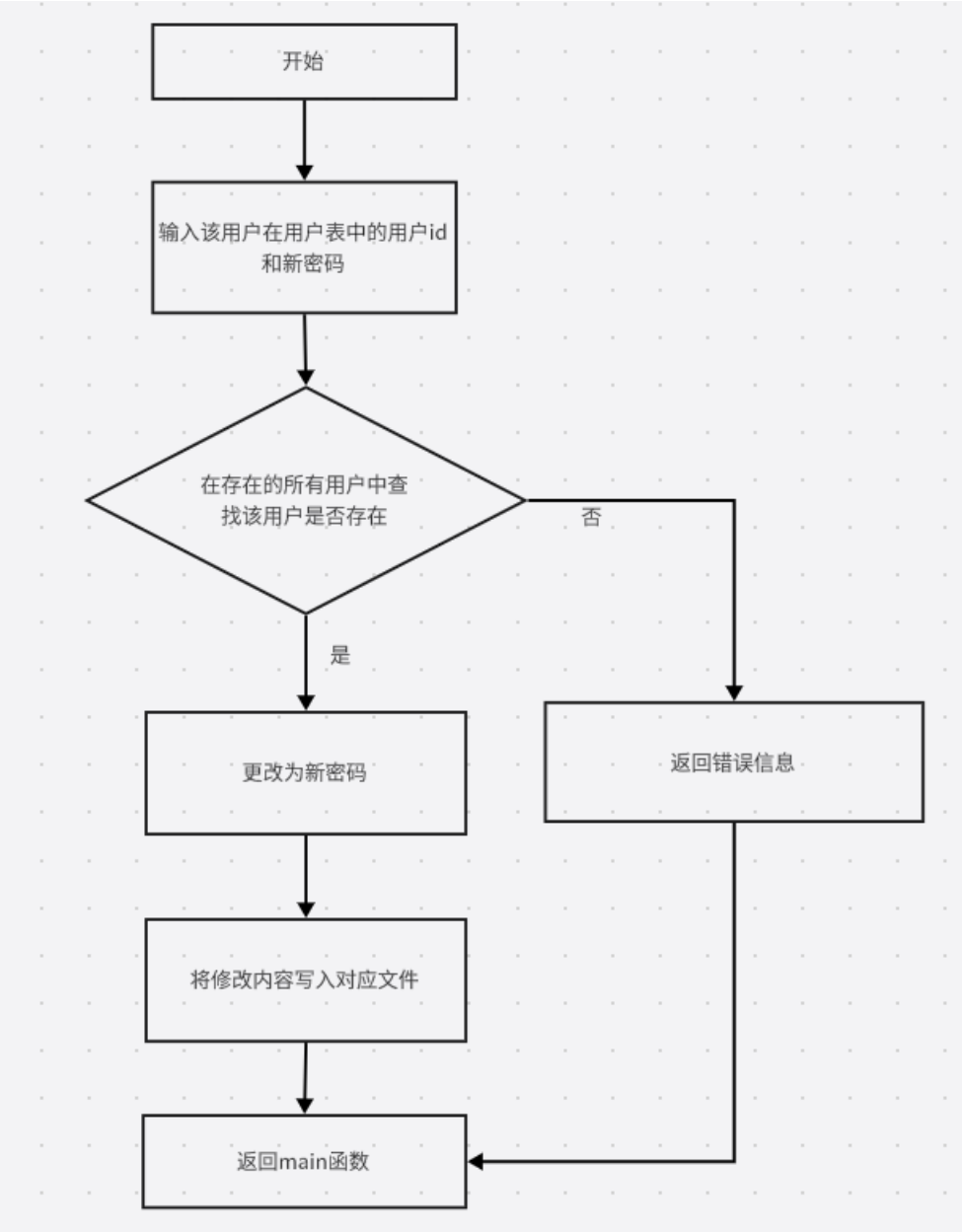
pwd

直接输出即可，无流程图

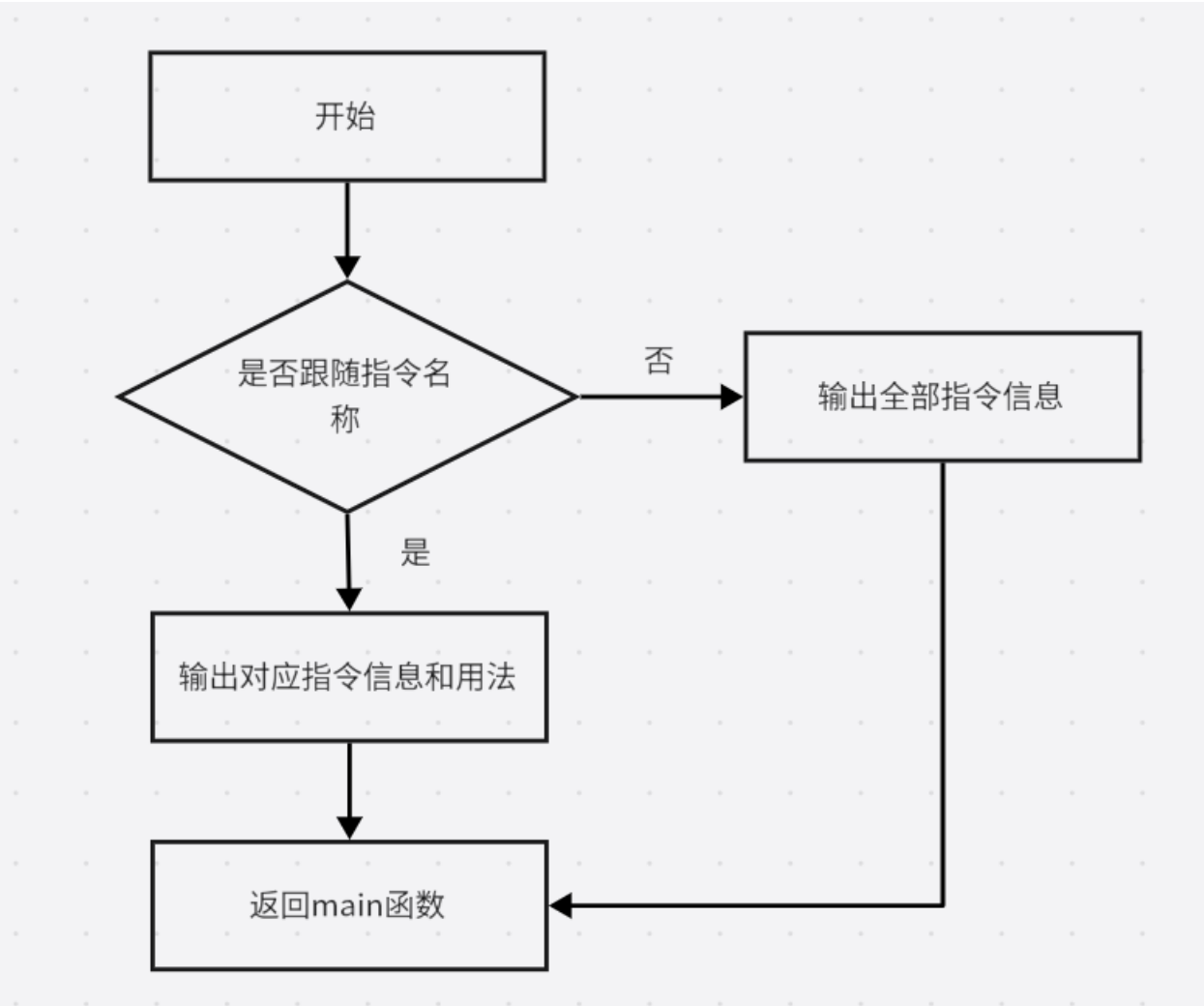
who

直接输出即可，无流程图；

password



help



运行结果

见报告各部分

实验总结

通过此次实验，大家对文件系统的运行过程有了更加深刻的理解；

虽然老师已经给出了一部分代码，但是整体而言难度还是有的，尤其是在涉及文件读写方面时，不太熟练导致花费了较多时间；

整体而言将老师建议文档中的全部部分完成了，但是肯定还是有些bug没有被发现，还有一些功能可以实现，比如删除用户、目录等；