总体设计

Bitmap 部分

bitmap分成两种

- inode bitmap
- datablock bitmap

inode bitmap

前者有2^{15}个，大小为4KB,是1个datablock

datablock bitmap

而共有256MB空间，为2^{28} 大小

而每一个datablock都是2^{12} 大小

所以一空需要2^{16} 个标记为

为64K bit，8KB 占用2个datablock

空间安排

从第一块是inode

从第二三块，是datablock node

iNode 部分

```
typedef struct

{
    mode_t  st_mode; //文件对应的模式    4
    nlink_t st_nlink;//文件的链接数        8
    uid_t   st_uid;  //文件所有者         4
    gid_t   st_gid;  //文件所有者的组     4
    off_t   st_size; //文件字节数         8
    time_t  atime;//被访问的时间         8
    time_t  mtime;//被修改的时间         8
    time_t  ctime;//状态改变时间         8
    int direct_pointer[DerictPointer];
    int indirect_pointer[InDerictPointer];
    int id;
    char make_full[12];

}iNode;
```

特别鸣谢徐晨同学为我提供了对其到128字节的思路，使得后续操作简单很多

由于总体要支持32768个文件，所以要有2^{15} 个iNode

经过设计和补全，每一个inode的大小都有128 = 2^7 字节大小。所以对于inode部分共有2^{22} 也就是4MB的大小

我们不妨规定，/这个最初的目录在所有文件中排第一个

inode是从第25个块开始的

每一个块中有4KB，也就是会有32个inode

所以inode部分为从第4块到第1027块

Data Block 部分

从1027到最终

目录文件的存储

设计数据结构

```
typedef struct  DictioryContent {
    char file_name[24];
    char file_extent[4];
    int inode_number;
};
```

如此一来得到每一个文件占用32个字节，这样一个块4KB中有128个文件可以存

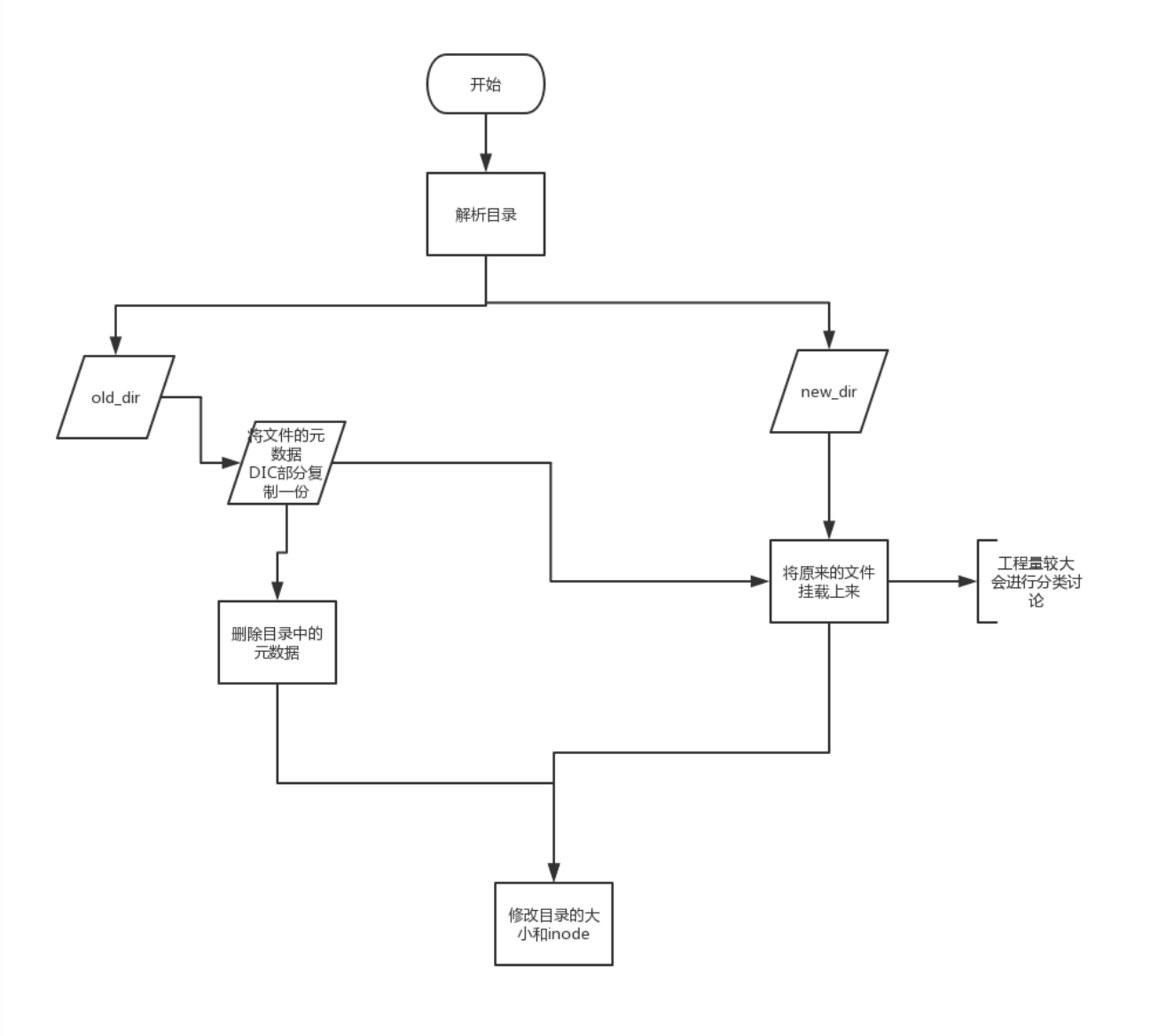
文件的储存结构

从上到下依次进行，有着大小标记。

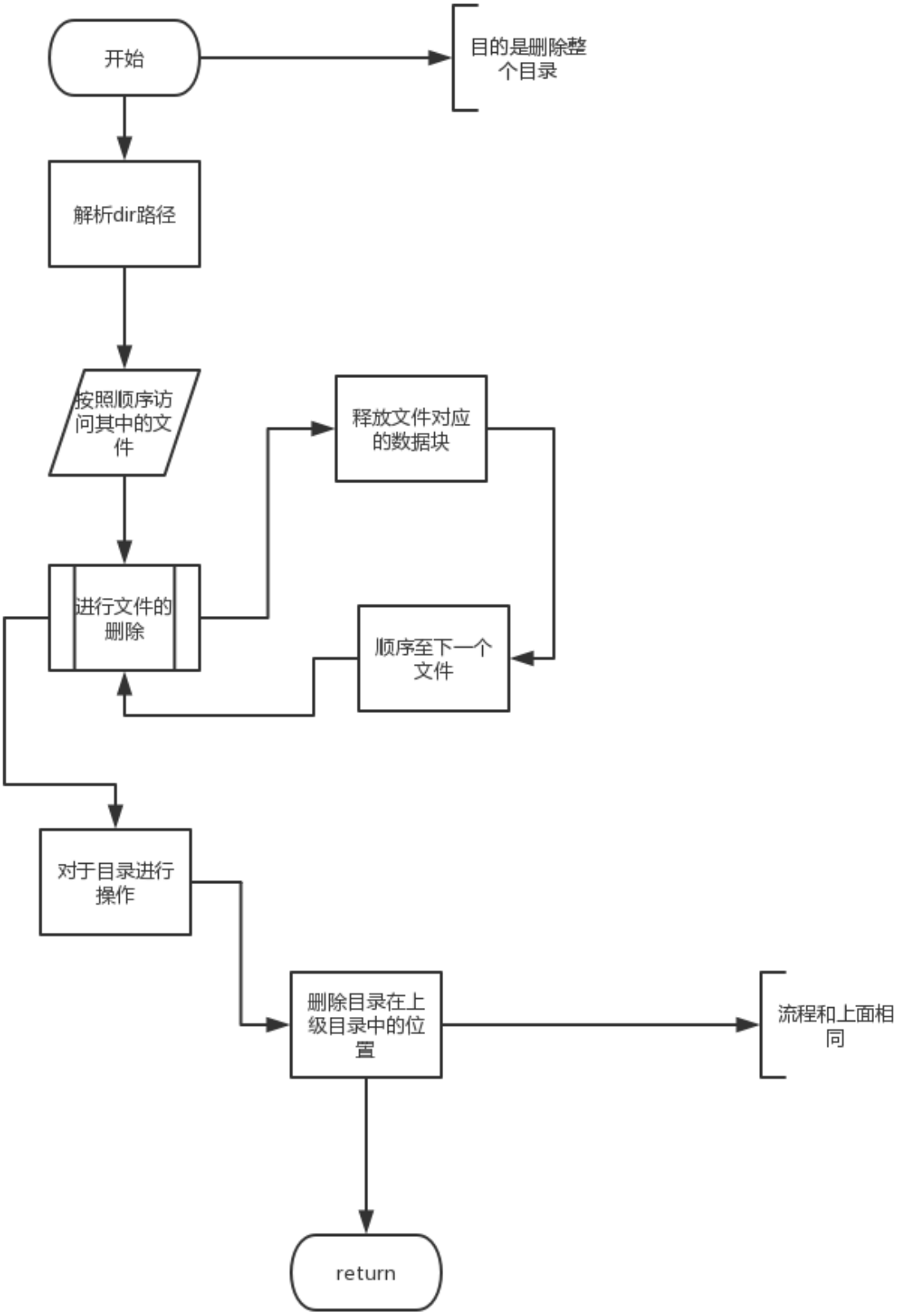
具体实现

部分函数的流程图

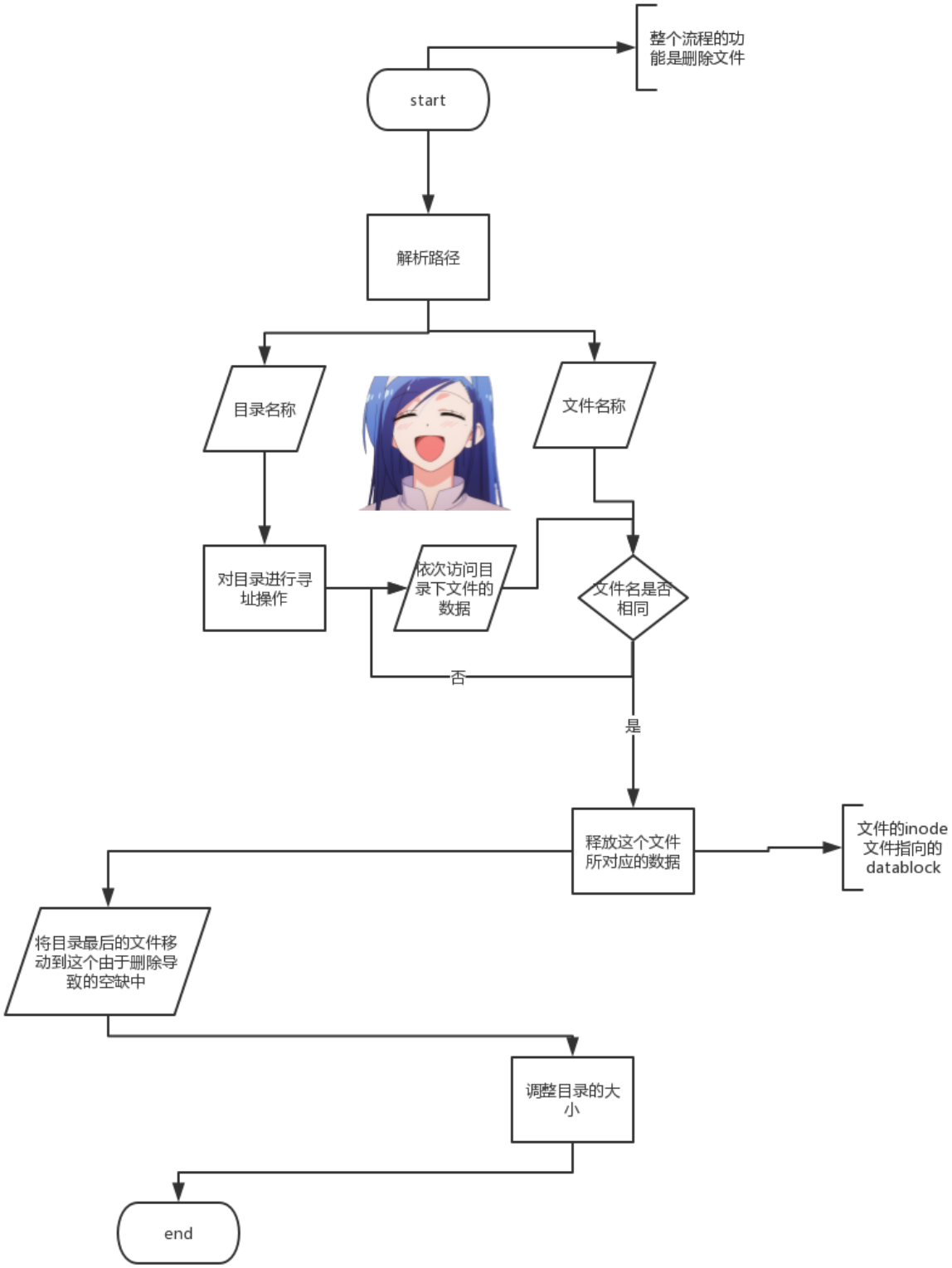
mv



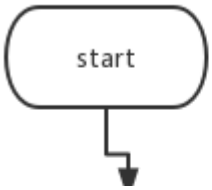
rename

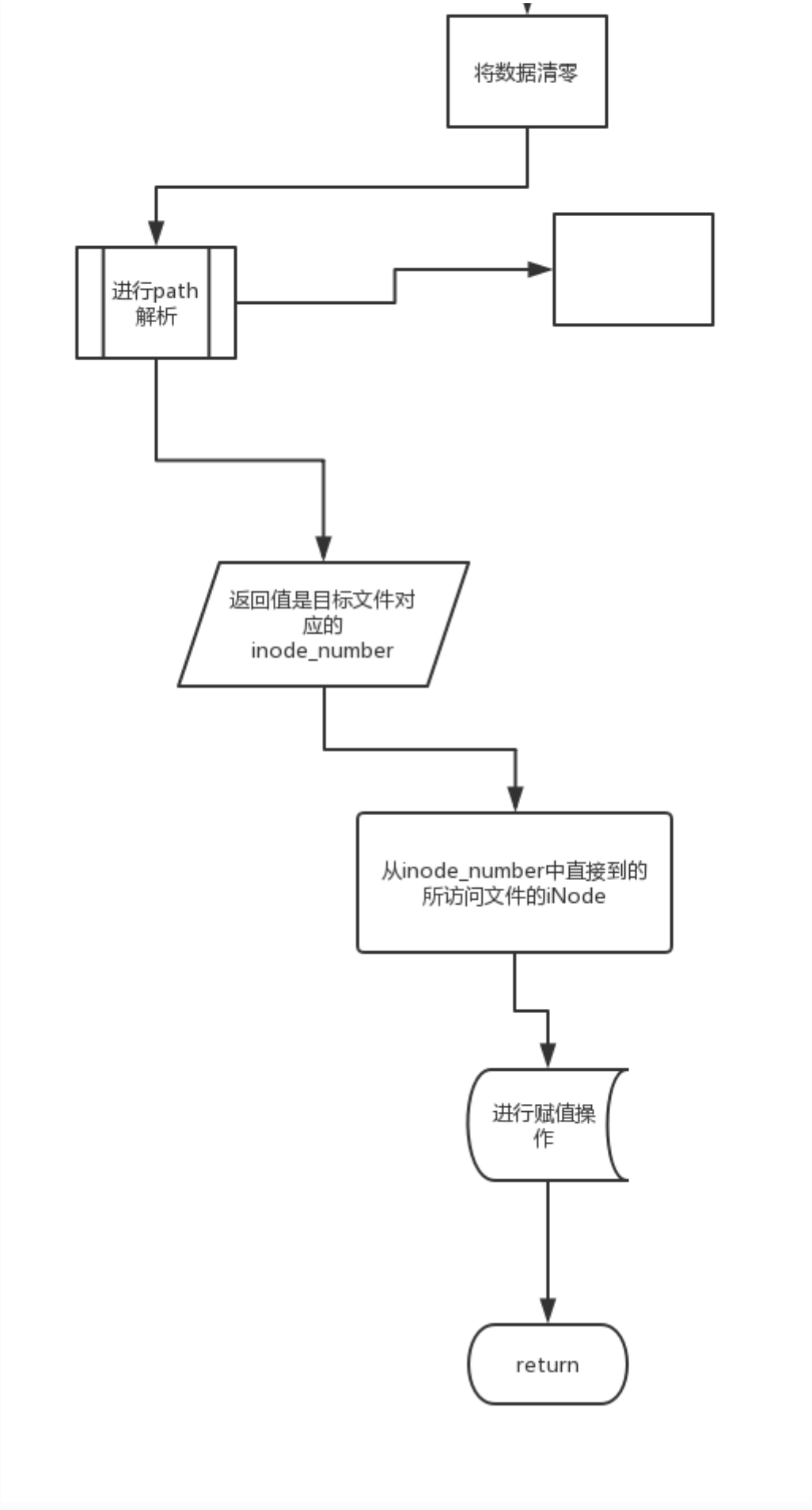

开始
目的是删除整个目录
解析dir路径
按照顺序访问其中的文件
释放文件对应的数据块
进行文件的删除
顺序至下一个文件
对于目录进行操作
删除目录在上级目录中的位置
流程和上面相同
return

rm


整个流程的功能是删除文件
start
解析路径
目录名称
文件名称
对目录进行寻址操作
依次访问目录下文件的数据
文件名是否相同
否
是
释放这个文件所对应的数据
文件的inode文件指向的datablock
将目录最后的文件移动到这个由于删除导致的空缺中
调整目录的大小
end

get attr


start


将数据清零
进行path
解析
返回值是目标文件对
应的
inode_number
从inode_number中直接到的
所访问文件的iNode
进行赋值操
作
return

调试过程

touch指令

touch (有bug)（已经此修复）

建立第二个目录之后，第二个目录中进行touch会出现错误

可能是mkdir的错误

或者是mknod的错误

可能是目录挂载的时候出错了

或者是建立文件写入的时候没有*申请*声明初始的datablock

几个尝试

- 先创建了很多文件，再进入第一个文件夹，这个文件夹的中进行touch会出错
- 先创建了很多文件，再进入第一个文件夹，这个文件夹的中进行mkdir会出错
- 创建了两个文件夹，进入第二个之后创建文件夹和文件都会出错
- 创建第一个文件夹，再这一层中可以随意操作，但是再进一层之后则会出错

发现问题

可以申请空间，也可以写进去

但是之后找不到

之后读进去也可以找到，可能是

understanding file的问题？？

最终定位到了understading the path

最终解决方案

原本准备要对于understanding path 进行重构，

将understanding path阅读完成之后，发现是在解析路径的时候出现了一个参数的错误

这个参数的错误导致所有的搜寻都是在根目录上完成的

rm指令

调试 rm(已经完成)

发现问题

rm在根目录下可以进行正常的删除

现在已经确定可以正常的读取比较了

错误在于将最后一个移动到被删除的位置的那一步操作，也就是line 1231

已经测试

在跟目录下进行删除空文件没有问题

```
start to getatr
        understanding the file'/2'length is 2
        the inode number we get  in get_arr is 1
 Getattr is called:/2
        understanding the file'/2'length is 2
        understanding the file''length is 0
Unlink is callded:/2
Opendir is called:/
```

```
start to getatr
        understanding the file'/2'length is 2
        the inode number we get  in get_arr is 1
 Getattr is called:/2
        understanding the file'/2'length is 2
        understanding the file''length is 0
Unlink is callded:/2
Opendir is called:/
```

但是会出现删除错误的情况??????????????????????????????????

程序迷惑行为

```
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$ ls
1  2  3  4
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$ rm 2
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$ ls
1  2  3
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$
```

```
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$ ls
1  2  3  4
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$ rm 2
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$ ls
1  2  3
2017202010@VM-0-17-ubuntu:~/version12/fslab-handout/mnt$
```

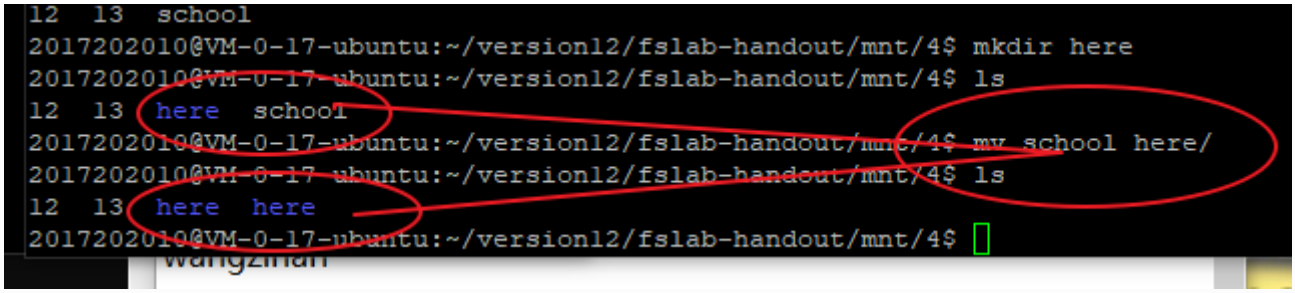
这部分功能已经完成

在非根目录下删除文件

在非根目录下删除非空的文件

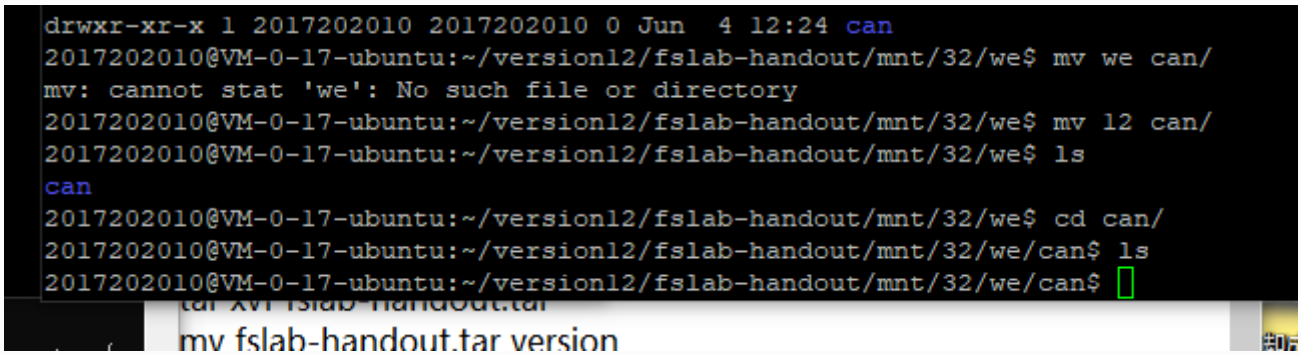
在跟目录下删除非空的文件

mv指令



进行move操作的时候没有把原来的文件写到新的路径中，反而是把目标目录reloadl了一次，十分迷惑

上述问题已经解决



目前的问题式在更深层进行mv操作无法在新路径中挂载

一层二层好像都可以，但是三层好像不行

~~调不动了，放弃治疗~~

心得体会

工程量大，要做好版本管理

即使最后自己的文件系统也不足以应对各种特殊情况，还有许多需要哦考虑

~~对自己的工程能力有了自信（不）~~

~~最后块干不动的时候对于以后的职业选择产生了怀疑（指正）~~

函数注释在大项目中十分重要

学会封装可以减少写作和调试的成本

~~以后的大工程可能双人写会更有效率一点~~