

# 操作系统实验五

## 文件系统

白晋斌

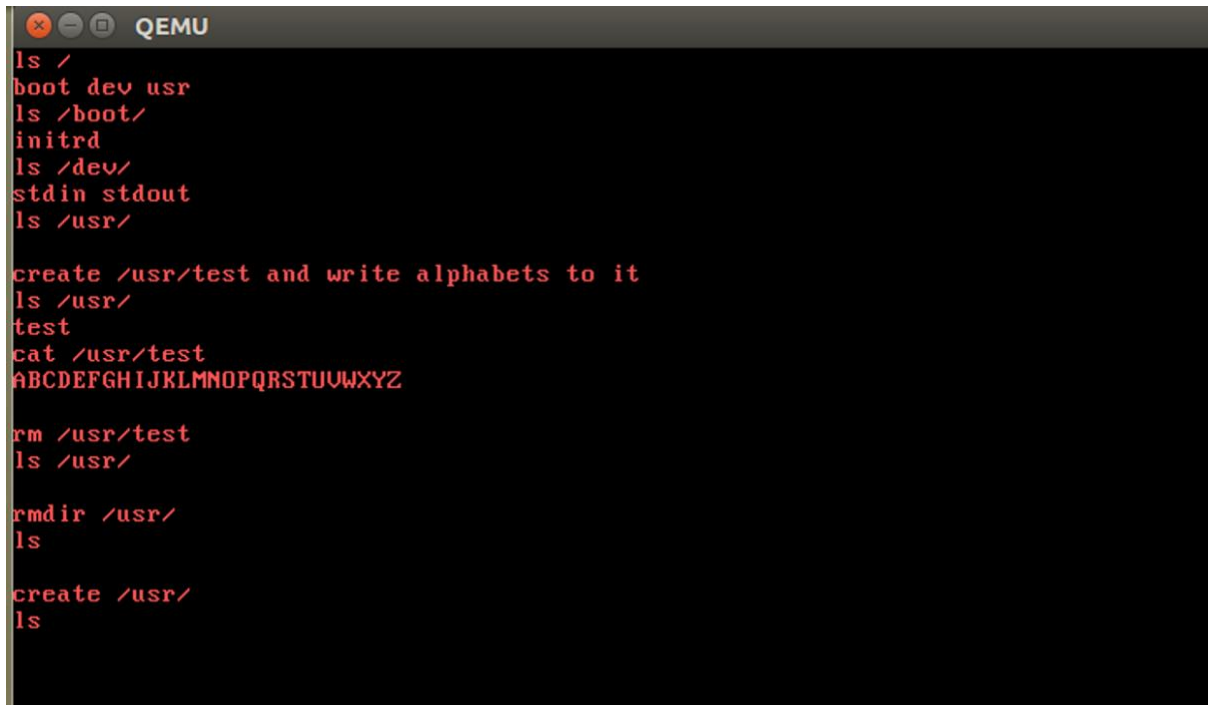
171860607

## 目录

一、实验结果.....	3
二、实验思路.....	3
1.step1 .....	3
2.step2 .....	4
3.step3 .....	4
4.step4 .....	4
5.step5 .....	4
6.step6 .....	4
7.step7 .....	5
8.step8 .....	5
9.step9 .....	5

## 一、实验结果

最终运行结果如图所示。



```
ls /
boot dev usr
ls /boot/
initrd
ls /dev/
stdin stdout
ls /usr/

create /usr/test and write alphabets to it
ls /usr/
test
cat /usr/test
ABCDEFGHIJKLMNOPQRSTUVWXYZ

rm /usr/test
ls /usr/

rmdir /usr/
ls

create /usr/
ls
```

主要的一处 bug 估计在 open 函数中，当 rmdir /usr/ 时，/消失。因时间问题，原因未知。

在 read 与 write 中出现的跨块问题均得到解决。但后续因为对 open 函数的修改，导致 cat 中 printf 一定概率输出异常（刷屏或者不输出，已排除 read 与 write 中出问题的可能性，并定位于 open 函数）。

综上，主要问题在 step2 的 open 函数。其他部分基本可行。

## 二、实验思路

### 1.step1

仿照前文被注释掉的函数即可。

```
format(driver,SECTOR_NUM,SECTORS_PER_BLOCK);
char destDirPath[NAME_LENGTH];
strcpy("/boot/", destDirPath, NAME_LENGTH - 1);
mkdir(driver,destDirPath);
strcpy("/dev/", destDirPath, NAME_LENGTH - 1);
mkdir(driver,destDirPath);
strcpy("/usr/", destDirPath, NAME_LENGTH - 1);
mkdir(driver,destDirPath);

strcpy("/dev/stdin", destFilePath, NAME_LENGTH - 1);
```

```
touch(driver, destFilePath);
strcpy("/dev/stdout", destFilePath, NAME_LENGTH - 1);
touch(driver, destFilePath);

strcpy("/boot/initrd", destFilePath, NAME_LENGTH - 1);
//touch(driver, destFilePath);
//strcpy("/boot/", destFilePath, NAME_LENGTH - 1);
strcpy("./uMain.elf", srcFilePath, NAME_LENGTH - 1);
cp(driver, srcFilePath, destFilePath);
```

## 2.step2

open 函数，至今仍有 bug。

基本按照助教的注释所写，逻辑如下

```
if file exit (ret == 0) {
    if INODE type != file open mode
        //这里主要考虑目录文件在打开时未加入目录标志，返回-1
    if the file refer to a device in use
        //对比 inodeOffset 实现，返回序号
    if the file refer to a file in use
        //同上，但引用数加一
    if the file refer to a file not in use
        //寻找 state 空闲的 file，填写相关信息，返回序号
}
else {
    if O_CREATE not set
        //返回-1
    if O_DIRECTORY not set
        //创建目录
    else
        //创建文件
```

## 3.step3

write 函数

基本思路是 diskread 读取 inode，readBlock 读取 buffer，更新 buffer，之后写入 inode，再写入 disk。注意的是写入的长度大于块剩下的长度，要调用 allocBlock 申请新块。

## 4.step4

read 函数

基本思路是 diskread 读取 inode，readBlock 读取 buffer，将 buffer 写入指定位置。

## 5.step5

lseek 函数

对文件是否打开、打开方式做安全性检查后，根据 whence 做简单处理即可。

## 6.step6

close 函数

安全性检查考虑各种意外情况后，对 inode 的 linkcount 减一即可。

#### 7.step7

remove 函数

如果是设备或者文件使用中，直接返回-1，否则根据文件名判断是目录还是文件，调用 freeInode 函数即可。

#### 8.step8

ls 函数

打开目录，循环读入 direntry，分别输出对应的 name。

#### 9.step9

cat 函数

打开文件，输出内容。