OS lab5实验报告

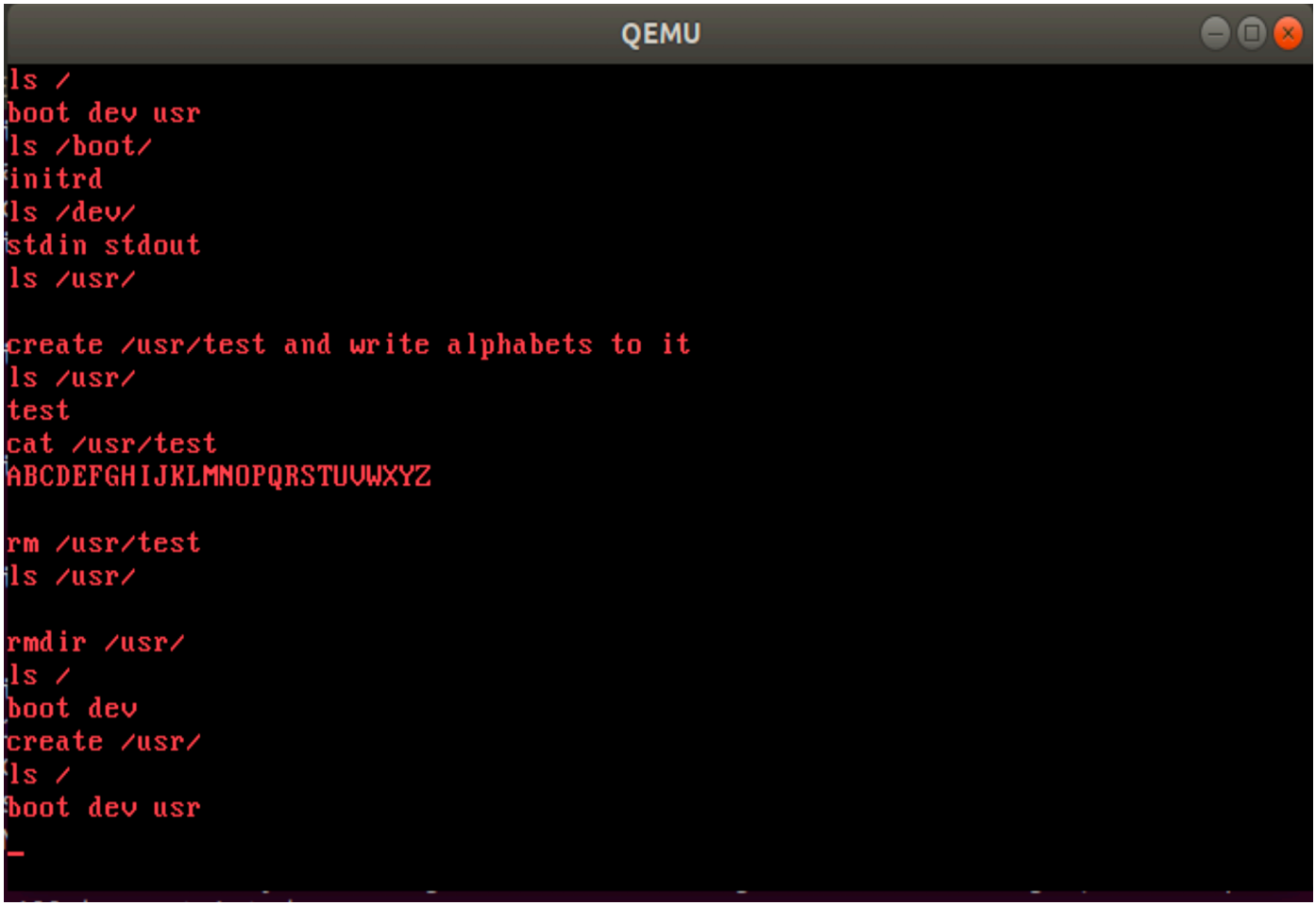
姓名	学号	邮箱	院系
张洋彬	191220169	1016466918@qq.com	计算机科学与技术系

一、实验目的

- 内核支持文件读写，实现 `open` ， `read` ,`write` ， `lseek` ， `close` ， `remove` 系统调用
- 实现 `ls` 和 `cat`

二、实验结果

完成了需完成的



三、实验过程

3.1 实现 `open`

按照助教给的to do，可以清楚的知道要完成什么，但有需要注意的几个点：

- 文件夹的取filename和文件取filename不一样，文件夹是以`/`结尾的，应该注意划分的时候的细节
- 同上，由于对str做出了改变，alloc完成后将str复原

```
void syscallOpen(struct StackFrame *sf) {
  ret = readInode(&sBlock, gDesc, &destInode, &destInodeOffset, str);
  if (file exist) {
     先检查是否是否是错误的情况：是文件夹没有set O_DIRECTORY，不是文件夹set了 O_DIRECTORY；
     检查dev和file数组，查看文件是否已经打开，如果已经打开返回-1；（dev返回下标）
     如果没有打开，则检查是否还有空位，如果有则加入file数组，如果没有则返回-1；
  }
  else { // try to create file
       创建文件或文件夹，创建成功则返回fd；
    }
    pcb[current].regs.eax = -1; // create success but no available file[]
    return;
  }
```

3.2 实现`write`

完成了`syscallWrite`和`syscallWriteFile`

在`syscallWrite`中调用`syscallWriteFile`(fd合法的情况下)。

write为Linux提供的系统原语，其用于向fd索引的FCB中的文件读写偏移量处开始，向文件中写入从buffer开始的内存中的size个字节，并返回成功写入的字节数，若文件支持seek操作，则同时修改该FCB中的文件读写偏移量

3.3 实现`read`

完成了`syscallRead`和`syscallReadFile`

在`syscallRead`中调用`syscallReadFile`(fd合法的情况下)。

read为Linux提供的系统原语，其用于从fd索引的FCB中的文件读写偏移量处开始，从文件中读取size个字节至从buffer开始的内存中，并返回成功读取的字节数，若文件支持seek操作，则同时修改该FCB中的文件读写偏移量。

3.4 实现`lseek`

1. 如果 whence 是 SEEK_SET，文件偏移量将被设置为 offset。
2. 如果 whence 是 SEEK_CUR，文件偏移量将被设置为 cfo 加上 offset， offset 可以为正也可以为负。
3. 如果 whence 是 SEEK_END，文件偏移量将被设置为文件长度加上 offset，offset 可以为正也可以为负。

代码如下

```
switch(sf->ebx) { // whence
    case SEEK_SET:
      file[sf->ecx - MAX_DEV_NUM].offset = offset;// TODO: if SEEK_SET
      break;
    case SEEK_CUR:
      file[sf->ecx - MAX_DEV_NUM].offset += offset;// TODO: if SEEK_CUR
      break;
    case SEEK_END:
      file[sf->ecx - MAX_DEV_NUM].offset = inode.size + offset;// TODO: if SEEK_END
      break;
    default:
      break;
  }
```

3.5 实现`close`

close为Linux提供的系统原语，其用于关闭由fd索引的FCB。（对应的fd全设成0）

代码如下：

```
int i = (int)sf->ecx;
  if (i < MAX_DEV_NUM || i >= MAX_DEV_NUM + MAX_FILE_NUM) {
    // TODO: dev, can not be closed, or out of range
    pcb[current].regs.eax = -1;
    return;
  }
  if (file[i - MAX_DEV_NUM].state == 0) {
    // TODO: not in use
    pcb[current].regs.eax = -1;
    return;
  }
  file[i - MAX_DEV_NUM].state = 0;
  file[i - MAX_DEV_NUM].inodeOffset = 0;
  file[i - MAX_DEV_NUM].offset = 0;
  file[i - MAX_DEV_NUM].flags = 0;
  pcb[current].regs.eax = 0;
  return;
```

3.6 实现`remove`

remove为C标准库的函数，其用于删除path指定的文件。和open类似，注意删除文件夹和文件的区别，调用`freeInode`函数。

3.7 实现`ls`和`cat`

要写的内容就是to do的内容，也就是读fd里的信息。同时也注意文件夹和文件的区别。

ls：

```
fd = open(destFilePath, O_READ | O_DIRECTORY);
  if (fd == -1)
    return -1;
  ret = read(fd, buffer, 512 * 2);
  while (ret != 0) {
    dirEntry = (DirEntry *)buffer;
    for (i = 0; i < (512 * 2) / sizeof(DirEntry); i ++) {
      if (dirEntry[i].inode != 0)
        printf("%s ", dirEntry[i].name);
    }
    ret = read(fd, buffer, 512 * 2);// TODO: Complete 'ls'.
  }
```

Cat:

```
fd = open(destFilePath, O_READ);
  if (fd == -1)
    return -1;
  ret = read(fd, buffer, 512 * 2);
  while (ret != 0) {
    for( int i=0;i<512*2;i++){
      if(buffer[i]==0){
        break;
        ret=0;
      }
      printf("%c",buffer[i]);
    }
    ret = read(fd, buffer, 512 * 2);
  }
```

四、感想与心得

这次实验感觉比较复杂，需要对文件系统有深刻的理解才能实现。读框架代码读了很久才知道了怎么去下笔，感谢yxz同学为我解答了很多疑惑，在完成了这次实验后对文件系统的理解也加深了。总体上还是基于前面四次实验，也是对前四次实验的复习。结束了就好好复习理论课了！