操作系统lab5【海纳百川-文件系统】

201220199 肖丹妮

- 3个exercise
- 1个challenge
- 2个task

Exercises

exercise1

1. 文件名是字符串，使用文件名作为文件标识，占用存储空间大并且操作速度慢；文件描述符本身是整数，使用文件描述符，相比文件名操作速度更快。
2. 文件描述符本身直接与FCB相关联，可以直接查询到文件的FCB，但是文件名难以做到这一点。

exercise2

因为通过exec执行新程序时，并没有减少或者增加进程的数目，新程序的工作目录将会继承原来程序所在的目录。当进程结束、执行exit时, 会对文件描述符表中所有有效的表项分别执行close，保证文件的正常关闭。

exercise3

1. which ls是输出ls程序所在文件的绝对路径，因为ls文件的绝对路径是/usr/bin/ls，所以输出结果是“/usr/bin/ls”;
2. ls命令本身是打印当前工作目录下的文件列表，在/home/yxz这个目录执行ls时，当前工作目录是/home/yxz/，因此输出的是/home/yxz/路径下的文件列表。

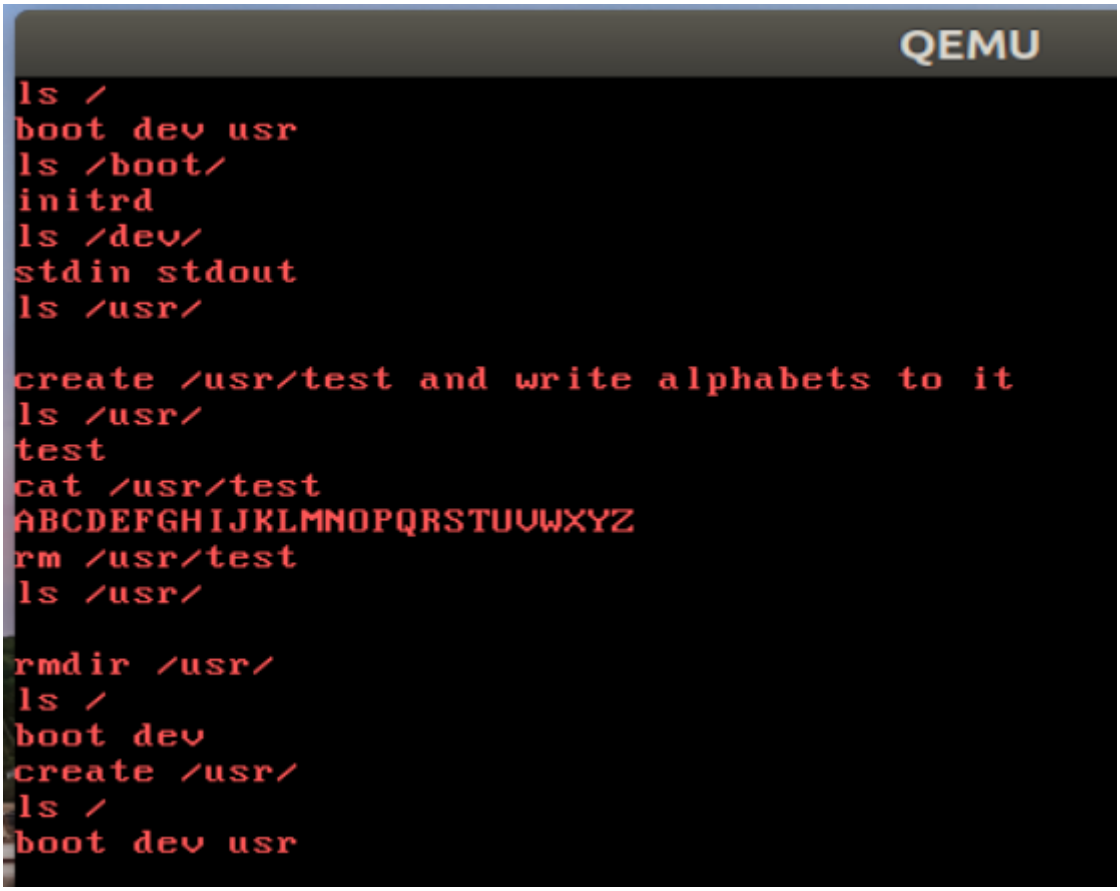
Challenge

challenge1

```
oslab@oslab-VirtualBox:~/draft$ gcc challenge.c -o challenge
oslab@oslab-VirtualBox:~/draft$ ./challenge
total 76
drwxrwxr-x  3 oslab oslab 4096 6月    7 01:00 .
drwxr-xr-x 22 oslab oslab 4096 6月    4 22:52 ..
-rw-rw-r--  1 oslab oslab  584 6月    7 00:59 a.txt
-rwxrwxr-x  1 oslab oslab 8744 6月    7 01:00 challenge
-rw-rw-r--  1 oslab oslab  849 6月    7 01:00 challenge.c
-rwxrwxr-x  1 oslab oslab 8432 4月   20 04:50 test
-rwxrwxr-x  1 oslab oslab 8432 4月   25 09:37 test1
-rw-rw-r--  1 oslab oslab  859 4月   25 09:37 test1.c
-rwxrwxr-x  1 oslab oslab 8736 6月    7 00:57 test2
-rw-rw-r--  1 oslab oslab  605 4月   20 04:50 test.c
drwxrwxr-x  2 oslab oslab 4096 6月    6 22:33 www
oslab@oslab-VirtualBox:~/draft$ ls -al
total 76
drwxrwxr-x  3 oslab oslab 4096 6月    7 01:00 .
drwxr-xr-x 22 oslab oslab 4096 6月    4 22:52 ..
-rw-rw-r--  1 oslab oslab  584 6月    7 01:00 a.txt
-rwxrwxr-x  1 oslab oslab 8744 6月    7 01:00 challenge
-rw-rw-r--  1 oslab oslab  849 6月    7 01:00 challenge.c
-rwxrwxr-x  1 oslab oslab 8432 4月   20 04:50 test
-rwxrwxr-x  1 oslab oslab 8432 4月   25 09:37 test1
-rw-rw-r--  1 oslab oslab  859 4月   25 09:37 test1.c
-rwxrwxr-x  1 oslab oslab 8736 6月    7 00:57 test2
-rw-rw-r--  1 oslab oslab  605 4月   20 04:50 test.c
drwxrwxr-x  2 oslab oslab 4096 6月    6 22:33 www
```

Tasks

结果如下



task1

- open
  - open1
    - 比较(sf->edx >> 3) % 2与destInode.type是否一致
  - open2
    - 遍历dev和file数组，如果dev[i].inodeOffset/file[i].inodeOffset == destInodeOffset == destInodeOffset，说明需要打开的是dev[i]/file[i]，若dev[i]/file[i].state==1，说明已打开，返回-1
  - open3
    - (sf->edx >> 2) % 2 == 0（O_CREATE没有被设置），返回-1
  - open4
    - stringChrR、stringCpy得到父目录fa_str，调用readInode得到fatherInode、fatherInodeOffset，目标文件名指针destfilename = str+size+1，调用allocInode，参数是REGULAR_TYPE。
  - open5
    - 若要建立的是目录文件，可能目录字符串最后是'/'，需忽略掉再利用stringChrR、stringCpy得到真正的父目录fa_str，allocInode参数是DIRECTORY_TYPE。
- write
  - write1
    - FCBindex < 0 || FCBindex >= MAX_FILE_NUM，文件超出范围
    - file[FCBindex].state == 0，文件没有打开
  - writefile1

```
    int ret = 0;
    if (quotient+j == inode.blockCount)
        ret = allocBlock(&sBlock, gDesc, &inode, file[sf->ecx -
MAX_DEV_NUM].inodeOffset);
    if(ret==-1){sf->eax=-1;return;}
    ret = readBlock(&sBlock, &inode, quotient+j, buffer);
    if(ret==-1){sf->eax=-1;return;}

    MemCpy(str+stroff, (uint8_t*)buffer+remainder, BLOCK_SIZE-remainder);
    stroff+=BLOCK_SIZE-remainder;
    ret = writeBlock(&sBlock, &inode, quotient+j, buffer);
    if(ret==-1){sf->eax=-1;return;}
    j++;

    while(stroff < sz){
        MemCpy(str+stroff, (uint8_t*)buffer, BLOCK_SIZE);
        stroff+=BLOCK_SIZE;
        if (quotient+j == inode.blockCount)
            ret = allocBlock(&sBlock, gDesc, &inode, file[sf->ecx -
MAX_DEV_NUM].inodeOffset);
        if(ret==-1){sf->eax=-1;return;}
        ret = writeBlock(&sBlock, &inode, quotient+j, buffer);
        if(ret==-1){sf->eax=-1;return;}
        j++;
    }
```

  - writefile2

```
    inode.size=sz+file[sf->ecx-MAX_DEV_NUM].offset;
    diskWrite(&inode, sizeof(Inode), 1, file[sf->ecx-
MAX_DEV_NUM].inodeOffset);
```

- read
  - 与write类似
- lseek
  - lseek1
    - ofs = offset;
  - lseek2
    - ofs = file[FCBindex].offset + offset;
  - lseek3
    - ofs = inode.size + offset;
- close

```
    int FCBindex = i - MAX_DEV_NUM;
    file[FCBindex].state = 0;
    file[FCBindex].flags = 0;
    file[FCBindex].inodeOffset = 0;
    file[FCBindex].offset = 0;
```

- remove
  - remove1
    - 文件本身不应该被使用
  - remove2
    - stringChrR、stringCpy得到父目录fa_str，调用readInode得到fatherInode、fatherInodeOffset，目标文件名指针destfilename = str+size+1，调用freeInode，参数是REGULAR_TYPE。
  - remove3
    - 若要建立的是目录文件，可能目录字符串最后是'/'，需忽略掉再利用stringChrR、stringCpy得到真正的父目录fa_str，freeInode参数是DIRECTORY_TYPE。

task2

- ls

从文件中读目录项到buffer，依次输出buffer中所有文件名，继续从文件中读目录项到buffer、输出，直至读完。

```
        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);
```

- cat

从文件中读内容到buffer，将buffer中内容打印到标准输出，继续从文件中读内容到buffer、输出，直至读完。

```
        write(STD_OUT, buffer, ret);
        ret = read(fd, buffer, 512 * 2);
```