

# 浙江大学实验报告

- **课程名称**：Linux程序设计
- **实验类型**：设计型
- **实验项目**：实验2-shell程序设计
- **学生姓名**：黄彦玮
- **专业**：混合班
- **学号**：3180102067
- **电子邮件**：[3180102067@zju.edu.cn](mailto:3180102067@zju.edu.cn)
- **手机**：15852758315
- **实验日期**：2020年07月15日

## 浙江大学实验报告

一、实验环境

二、实验内容和结果及分析

Exp2-1

实验结果说明

完整代码

Exp2-2

实验结果说明

完整代码

Exp2-3

实验结果说明

完整代码

Exp2-4

需求描述

设计文档

- 1、设计思想
- 2、设计亮点
- 3、功能模块
- 4、实现方法
- 5、改进与讨论

实验结果

- 1、主程序及管理员模式
- 2、教师模式
- 3、学生模式

完整代码

main.sh  
admin.sh  
teacher.sh  
student.sh

Exp2-5

设计文档

- 1、设计思想
- 2、设计亮点
- 3、功能模块
- 4、实现方法
- 5、讨论与改进

用户手册

- 1、简介
- 2、版本特性
- 3、安装方式
- 4、功能说明
- 5、指令说明
- 6、信号
- 7、附录：Linux Shell 相关概念解释

实验结果

完整代码

- Makefile
- main.h
- main.c
- controller.h
- controller.c
- mycmd.h
- mycmd.c
- proc\_list.h
- proc\_list.c

实验心得

## 一、 实验环境

**内存**：4GB

**CPU**：Intel® Core™ i7-8750H CPU @ 2.20GHz

**处理器**：2

**硬盘**：40GB

**操作系统环境**：Ubuntu 64位

**Linux版本**：Ubuntu 20.04

## 二、 实验内容和结果及分析

### Exp2-1

( 10分 ) 编写shell 脚本，统计指定目录下的普通文件、子目录及可执行文件的数目，统计该目录下所有普通文件字节数总和，目录的路径名字由参数传入。

### 实验结果说明

```

hyw@ubuntu:~$ ls -l
total 428
drwx--x--x 2 hyw hyw 4096 Jul 14 23:03 Desktop
-rw----- 1 hyw hyw 0 Jul 15 00:16 error.log
-rw----- 1 hyw hyw 82 Jul 15 00:25 fdata
-rw----- 1 hyw hyw 17 Jul 15 00:11 foobar.path
-rw----- 1 hyw hyw 126 Jul 15 00:25 fout
-rwxrwxr-x 1 hyw hyw 16688 Jul 15 15:25 hello
-rw-rw-r-- 1 hyw hyw 60 Jul 15 15:25 hello.c
-rw----- 1 hyw hyw 368652 Jul 14 23:10 largeFile
-rw----- 1 hyw hyw 1960 Jul 14 23:10 mediumFile
-rw----- 1 hyw hyw 12 Jul 15 00:16 output.data
-rw----- 1 hyw hyw 964 Jul 14 23:10 smallFile
-rw----- 1 hyw hyw 12 Jul 15 00:15 student.records
drwx--x--x 5 hyw hyw 4096 Jul 14 23:19 temp
-rwxrwxrwx 1 hyw hyw 1569 Jul 15 16:25 test.sh
hyw@ubuntu:~$ ./test.sh ~
number of normal files: 12
number of directories: 2
number of executable files: 2
the total bytes of normal files: 390142

```

## 完整代码

```

#!/bin/bash
#程序名: test.sh (对应实验2第1题)
#作者: 黄彦玮 3180102067
#判断参数个数是否为1, 若不是则退出
if test $# -ne 1
then
    echo "Parameter should be only one!"
    exit 1
fi

#储存文件名
dirname="$1"
#定义整形变量, 分别表示普通文件、目录、可执行文件的个数、普通文件总字节数
typeset -i num_of_normal=0
typeset -i num_of_dir=0
typeset -i num_of_exec=0
typeset -i sum=0
#定义整形临时变量
typeset -i tmp=0

#判断输入的目录是否存在
if [ -d $dirname ]
then
    #用ls -l指令列出详细信息后, 选择-开头的普通文件, 用grep统计后用wc计算行数
    num_of_normal=$(ls -l $dirname | grep '^-' | wc -l)
    echo "number of normal files: ${num_of_normal}"
    #用ls -F指令, 文件名后加/号的是目录, 加*号的是可执行文件
    num_of_dir=$(ls -F $dirname | grep '/' | wc -l)
    echo "number of directories: ${num_of_dir}"
    num_of_exec=$(ls -F $dirname | grep '*' | wc -l)
    echo "number of executable files: ${num_of_exec}"

```

```
#枚举文件夹下的文件，并用ls -l显示详细信息
for f in $(ls $dirname)
do
    set -- $(ls -l $f)
    #根据第一个字段的第一个字符判断是否为普通文件
    if [[ $1="^-" ]]
    then
        #将大小转成整数后相加
        tmp=$5
        sum=$(($sum+$tmp))
    fi
done
echo "the total bytes of normal files: $sum"
#处理完成，正确退出
exit 0
else
    #若目录不存在，则退出
    echo "The directory doesn't exist."
    exit 1
fi
```

## Exp2-2

( 10分 ) 编写一个shell 脚本，输入一个字符串，忽略 ( 删除 ) 非字母后，检测该字符串是否为回文(palindrome)。对于一个字符串，如果从前向后读和从后向前读都是同一个字符串，则称之为回文串。例如，单词“mom”，“dad”和“noon”都是回文串。

### 实验结果说明

```
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
abcba
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
abcbe
False
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
aaaa
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
abc*ba
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
***aab
False
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
abcba abc
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
mom
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
dad
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
noon
True
hyw@ubuntu:~/s ./test2.sh
请输入一个字符串：
abc?bb
False
```

## 完整代码

```
#!/bin/bash
#程序名： test2.sh （对应实验2第2题）
#作者： 黄彦玮 3180102067

echo "请输入一个字符串："
#读取字符串
read str
#得到字符串长度
typeset -i len=${#str}
#循环所用变量
typeset -i i=0
#枚举每个字符
```

```

for (( i=0; i<len; i++ ))
do
    tmpchar=${str:$i:1}
    #若当前字符为空格则表示字符串结束，直接退出循环
    if [[ $tmpchar = ' ' ]]
    then
        break
    #特判掉星号和问号的情况，因为这两个字符在正则表达式中可以代替字符
    elif [[ $tmpchar = '*' ]] || [[ $tmpchar = '?' ]]
    then
        :
    elif [[ ${tmpchar}==[a-zA-Z] ]]
    then
        #是英文字母，加入过滤后的字符串
        newstr="$newstr$tmpchar"
    fi
done

#获取过滤后的字符串长度
len=${#newstr}
#如果过滤后是空串，直接退出
if (( $len == 0 ))
then
    echo "True"
    exit 0
fi

for (( i=0; i<len; i++ ))
do
    tmp1=${newstr:$i:1}
    tmp2=${newstr:($len-$i-1):1}
    #从两头开始遍历每个字符比较是否一致
    if [[ $tmp1 != $tmp2 ]]
    then
        #不一致直接返回
        echo "False"
        exit 0
    fi
done

#比较完成，字符串是回文串
echo "True"
exit 0

```

## Exp2-3

( 15分 ) 编写一个实现文件备份和同步的shell脚本程序dirstsync。程序的参数是两个需要备份同步的目录，如：

dirstsync ~\dir1 ~\dir2 # ~\dir1为源目录，~\dir2为目标目录

dirsync程序实现两个目录内的所有文件和子目录（递归所有的子目录）内容保持一致。程序基本功能如下。

1) 备份功能：目标目录将使用来自源目录的最新文件，新文件和新子目录进行升级，源目录将保持不变。dirsync程序能够实现增量备份。

2) 同步功能：两个方向上的旧文件都将被最新文件替换，新文件都将被双向复制。源目录被删除的文件和子目录，目标目录也要对应删除。

3) 其它功能自行添加设计。

提示：不能使用现有的备份或同步程序，如：/usr/bin/rsync

## 实验结果说明

我在实验的(1)(2)两个要求基础上，添加了以下功能：

- 在备份/同步时可以区分一般文件/全部文件（含隐藏文件）
- 将输入的路径名自动扩展为绝对路径，因此，在命令行中即使输入的是相对路径也可以正常运行

参数说明如下：

-a：备份/同步全部文件，包括隐藏文件，若不申明此参数，则默认普通文件

-c：备份功能（即要求中的(1)功能，若不加-s参数则默认为此功能）

-s：同步功能（即要求中的(2)功能）

功能说明及演示：

**1、基本使用：**如下图，主目录下已有一个目录test3a，现新建一个目录test3b，并用dirsync test3a test3b命令进行同步。如下图所示，实现了递归更新，且目录下所有文件和子目录中的文件都是一致的。

```
hyw@ubuntu:~$ ls -al test3a
total 16
drwxrwxr-x 3 hyw hyw 4096 Jul 17 17:30 .
drwxr-xr-x 9 hyw hyw 4096 Jul 17 17:31 ..
drwxrwxr-x 2 hyw hyw 4096 Jul 17 13:58 a
-rw-rw-r-- 1 hyw hyw 7 Jul 17 17:30 bbb
hyw@ubuntu:~$ mkdir test3b
hyw@ubuntu:~$ dirs sync test3a test3b
hyw@ubuntu:~$ ls -al test3b
total 16
drwxrwxr-x 3 hyw hyw 4096 Jul 17 17:31 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 17:31 ..
drwxrwxr-x 2 hyw hyw 4096 Jul 17 17:31 a
-rw-rw-r-- 1 hyw hyw 7 Jul 17 17:31 bbb
hyw@ubuntu:~$ cat test3a/bbb
123456
hyw@ubuntu:~$ cat test3b/bbb
123456
hyw@ubuntu:~$ ls -al test3a/a
total 12
drwxrwxr-x 2 hyw hyw 4096 Jul 17 13:58 .
drwxrwxr-x 3 hyw hyw 4096 Jul 17 17:30 ..
-rw-rw-r-- 1 hyw hyw 4 Jul 17 13:58 a
hyw@ubuntu:~$ ls -al test3b/a
total 12
drwxrwxr-x 2 hyw hyw 4096 Jul 17 17:31 .
drwxrwxr-x 3 hyw hyw 4096 Jul 17 17:31 ..
-rw-rw-r-- 1 hyw hyw 4 Jul 17 17:31 a
hyw@ubuntu:~$ cat test3a/a/a
123
hyw@ubuntu:~$ cat test3b/a/a
123
```

2、增量更新：修改test3a/bbb内容，再进行备份，如下图所示：

```
hyw@ubuntu:~$ cat > test3a/bbb
abcdef
hyw@ubuntu:~$ dirs sync test3a test3b
hyw@ubuntu:~$ ls -al test3b
total 16
drwxrwxr-x 3 hyw hyw 4096 Jul 17 17:33 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 17:31 ..
drwxrwxr-x 2 hyw hyw 4096 Jul 17 17:33 a
-rw-rw-r-- 1 hyw hyw 7 Jul 17 17:33 bbb
hyw@ubuntu:~$ cat test3b/bbb
abcdef
```

3、隐藏文件更新：使用-a参数，删除文件bbb并新建文件.ccc，如下图所示：

```
hyw@ubuntu:~$ ls -al test3a
total 16
drwxrwxr-x  3 hyw hyw 4096 Jul 17 19:43 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 19:41 ..
drwxrwxr-x  2 hyw hyw 4096 Jul 17 13:58 a
-rw-rw-r--  1 hyw hyw    7 Jul 17 19:43 bbb
hyw@ubuntu:~$ ls -al test3b
total 16
drwxrwxr-x  3 hyw hyw 4096 Jul 17 19:43 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 19:41 ..
drwxrwxr-x  2 hyw hyw 4096 Jul 17 19:43 a
-rw-rw-r--  1 hyw hyw    7 Jul 17 19:43 bbb
hyw@ubuntu:~$ rm test3a/bbb
hyw@ubuntu:~$ cat > test3a/.ccc
asdf
hyw@ubuntu:~$ dirsync -a test3a test3b
hyw@ubuntu:~$ ls -al test3a
total 16
drwxrwxr-x  3 hyw hyw 4096 Jul 17 19:44 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 19:41 ..
drwxrwxr-x  2 hyw hyw 4096 Jul 17 13:58 a
-rw-rw-r--  1 hyw hyw    5 Jul 17 19:44 .ccc
hyw@ubuntu:~$ ls -al test3b
total 16
drwxrwxr-x  3 hyw hyw 4096 Jul 17 19:44 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 19:41 ..
drwxrwxr-x  2 hyw hyw 4096 Jul 17 19:44 a
-rw-rw-r--  1 hyw hyw    5 Jul 17 19:44 .ccc
```

**4、同步功能：**使用-s参数，实现两个文件夹的同步。对于可匹配的文件，进行双向更新（用最新的文件更新到两边），对于源目录中删除的文件，目标目录中也进行删除。如下图，分别在test3a中添加ddd文件，在test3b中修改.ccc文件，并且删除test3a/a目录下的a文件，用dirsinc -as test3a test3命令进行同步，可以看到test3b目录下多了ddd文件，和test3a/ddd一致；test3a/.ccc更新为与test3b/.ccc相同，这体现了双向更新；test3b/a目录下的a文件删除，体现了源目录下的子目录或子目录下的文件删除时，目标目录也进行删除。如下图所示。

```

hyw@ubuntu:~$ ls -al test3a
total 16
drwxrwxr-x 3 hyw hyw 4096 Jul 17 20:10 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 20:10 ..
drwxrwxr-x 2 hyw hyw 4096 Jul 17 20:10 a
-rw-rw-r-- 1 hyw hyw 4 Jul 17 20:08 .ccc
hyw@ubuntu:~$ ls -al test3b
total 16
drwxrwxr-x 3 hyw hyw 4096 Jul 17 20:10 .
drwxr-xr-x 10 hyw hyw 4096 Jul 17 20:10 ..
drwxrwxr-x 2 hyw hyw 4096 Jul 17 20:10 a
-rw-rw-r-- 1 hyw hyw 4 Jul 17 20:08 .ccc
hyw@ubuntu:~$ cat > test3a/ddd
132456
hyw@ubuntu:~$ cat > test3b/.ccc
uvw
hyw@ubuntu:~$ rm test3a/a/a
hyw@ubuntu:~$ ls -al test3b/a
total 12
drwxrwxr-x 2 hyw hyw 4096 Jul 17 20:10 .
drwxrwxr-x 3 hyw hyw 4096 Jul 17 20:10 ..
-rw-rw-r-- 1 hyw hyw 6 Jul 17 20:10 a
hyw@ubuntu:~$ dirsync -as test3a test3b
hyw@ubuntu:~$ ls -al test3b/a
total 8
drwxrwxr-x 2 hyw hyw 4096 Jul 17 20:11 .
drwxrwxr-x 3 hyw hyw 4096 Jul 17 20:11 ..
hyw@ubuntu:~$ cat test3a/ddd
132456
hyw@ubuntu:~$ cat test3b/ddd
132456
hyw@ubuntu:~$ cat test3b/.ccc
uvw
hyw@ubuntu:~$ cat test3a/.ccc
uvw

```

## 完整代码

```

#!/bin/bash
#程序名: test3.sh (对应实验2第3题)
#作者: 黄彦玮
#学号: 3180102067

mode="-c"
#判断参数个数
if test $# -eq 3  #参数个数为3, 捕捉参数
then
    if [[ $1 =~ -* ]]
    then
        if [[ $1 =~ a+ ]]
        then
            displayall="-a"  #储存-a参数, 后续指令中会用到
        fi
        if [[ $1 =~ s+ ]]
        then
            mode="-s"  #储存-s参数, 后续指令中会用到
        fi
    fi
fi

```

```

        fi
        dir1=$2
        dir2=$3          #储存两个目录的名字
    else
        echo "$1: Wrong Parameter!"
        exit 1
    fi
elif test $# -eq 2    #参数个数为2， 默认不带参数
then
    dir1=$1
    dir2=$2          #储存两个目录的名字
else
    echo "Wrong number of parameters!"
    exit 1
fi

#CheckDir(): 通过扫描环境变量判断目录是否存在，并将目录扩展为绝对路径
CheckDir_ret=""
CheckDir() {
    dir_name=$1
    CheckDir_ret=""
    #已经是绝对路径，则直接返回
    if [ -d ${dir_name} ]
    then
        CheckDir_ret=${dir_name}
        return 1
    fi
    #扫描环境变量
    array=(${PATH/\:/ })
    #顺序遍历环境变量进行检查
    for Env_var in ${array[@]}
    do
        if [ -d "${Env_var}/${dir_name}" ]
        then
            CheckDir_ret="${Env_var}/${dir_name}"
            return 1
        fi
    done
    return 0
}

#调用函数，将两个目录变为绝对路径
CheckDir $dir1
if test $? -eq 0
then
    echo "$dir1: Directory cannot be found!"
    exit 1
fi
dir1=${CheckDir_ret}
CheckDir $dir2
if test $? -eq 0
then
    echo "$dir2: Directory cannot be found!"
    exit 1
fi

```

```

dir2=${CheckDir_ret}

#定义整形变量flag用于后续循环
typeset -i flag=0

#以下为用数组模拟堆栈， stacklen表示堆栈长度
typeset -i stacklen=0
#入栈
push() {
    stack[$stacklen]=$1
    stacklen=$((stacklen+1))
}

#出栈， 分别将栈顶元素赋值给dir1和dir2
popdir1() {
    dir1=${stack[$((stacklen-1))]}
    stacklen=$((stacklen-1))
}

popdir2() {
    dir2=${stack[$((stacklen-1))]}
    stacklen=$((stacklen-1))
}

#最主要的处理函数， 用于将dir1同步到dir2
myRsyncCopy() {
    dir1=$1
    dir2=$2
    #预处理第二个目录下的文件和子目录
    for f in $(ls $displayall $dir2)
    do
        #. 和.. 两个文件夹特殊不需要移动， 需要特判
        if [[ $f == '.' ]] || [[ $f == '..' ]]
        then
            continue
        fi
        #将第二个目录下的所有文件和子目录重命名备用
        mv "$dir2/$f" "$dir2/TEMP_FILE_$f"
    done
    #枚举第一个文件夹中的文件， 搜索第二个文件夹中与之相匹配的文件
    for f in $(ls $displayall $dir1)
    do
        if [[ $f == '.' ]] || [[ $f == '..' ]]
        then
            continue
        fi
        #flag用于表示是否找到了与之匹配的文件
        flag=0
        #对于子目录， changedir表示与之匹配的子目录
        changedir=""
        #枚举第二个目录下的文件
        for g in $(ls $dir2)
        do
            #已经匹配过的文件就不考虑了
            if [[ $g =~ ^TEMP_FILE_ ]]
            then
                :
            fi
        done
        if [[ $flag -eq 0 ]]
        then
            #将第一个目录下的文件移动到第二个目录下
            mv "$dir1/$f" "$dir2/$f"
            #将第一个目录下的子目录移动到第二个目录下
            if [[ -d "$dir1/$f" ]]
            then
                mv -r "$dir1/$f" "$dir2/$f"
            fi
        fi
        flag=1
    done
}

```

```

        else
            continue
        fi

        if test -d "$dir1/$f" && test -d "$dir2/$g"      #对于子目录, 找到匹配的
子目录
        then
            #比较两个子目录下的目录名是否相同(不递归)
            diff "$dir1/$f" "$dir2/$g" 1>/dev/null
            if test $? -eq 0      #匹配
            then
                flag=1
                changedir=$g
                break
            fi
        elif test -f "$dir1/$f" && test -f "$dir2/$g"      #对于文件, 找到匹配的
文件
        then
            #比较两个文件是否相同
            diff -q "$dir1/$f" "$dir2/$g" 1>/dev/null
            if test $? -eq 0      #匹配
            then
                flag=1
                mv "$dir2/$g" "$dir2/$f"
                break
            fi
        #两个文件类型不同, 不匹配
        else
            continue
        fi
    done

    if test -d "$dir1/$f"
    then
        if test -n "$changedir"
        then
            #若子目录匹配, 则修改文件名表示已经匹配
            mv "$dir2/$changedir" "$dir2/$f"
            #递归调用函数, 需要先将两个文件夹名字存入堆栈, 调用完成后再取
出
            push "$dir1"
            push "$dir2"
            myRsyncCopy "$dir1/$f" "$dir2/$f"
            popdir2
            popdir1
        else
            #若无匹配, 直接拷贝
            scp -r "$dir1/$f" "$dir2/$f"
        fi
    elif test $flag -eq 0      #对于普通文件, 若无匹配, 直接拷贝
    then
        scp "$dir1/$f" "$dir2/$f"
    fi
done

```

```

#删除第二个目录下未匹配的临时文件
for f in $(ls $dir2)
do
    if [[ $f =~ ^TEMP_FILE_ ]]
    then
        rm -r "$dir2/$f"
    fi
done
}

myRsyncSync() {
    dir1=$1
    dir2=$2
    #枚举第一个文件夹中的文件，搜索第二个文件夹中与之相匹配的文件
    for f in $(ls $displayall $dir1)
    do
        if [[ $f == '.' ]] || [[ $f == '..' ]]
        then
            continue
        fi
        #flag用于表示是否找到了与之匹配的文件
        flag=0
        #枚举第二个目录下的文件
        for g in $(ls $displayall $dir2)
        do
            if [[ $f != $g ]]
            then
                continue
            fi

            if test -d "$dir1/$f" && test -d "$dir2/$g"      #对于子目录，找到匹配的
子目录
            then
                flag=1
                break
            elif test -f "$dir1/$f" && test -f "$dir2/$g"    #对于文件，找到匹配的文
件
            then
                flag=1
                #比较文件新旧，并双向同步
                if test "$dir1/$f" -nt "$dir2/$g"
                then
                    scp "$dir1/$f" "$dir2/$f"
                elif test "$dir1/$f" -ot "$dir2/$g"
                then
                    scp "$dir2/$f" "$dir1/$f"
                fi
                #修改匹配完成的文件的文件名，用于区分
                mv "$dir1/$f" "$dir1/TEMP_FILE_$f"
                mv "$dir2/$g" "$dir2/TEMP_FILE_$g"
                break
            fi
        done
        if test -d "$dir1/$f"

```

```

        then
            if test $flag -eq 1
            then
                #若子目录匹配，则修改文件名表示已经匹配
                mv "$dir1/$f" "$dir1/TEMP_FILE_$f"
                mv "$dir2/$f" "$dir2/TEMP_FILE_$f"
                #递归调用函数，需要先将两个文件夹名字存入堆栈，调用完成后再取出
            done
        fi
    done

    #删除第二个目录下未匹配的文件
    for f in $(ls $displayall $dir2)
    do
        if [ $f == . ] || [ $f == .. ]
        then
            continue
        fi
        if [[ $f =~ ^TEMP_FILE_ ]]
        then
            :
        else
            rm "$dir2/$f"
        fi
    done

    #将dir1中未匹配的复制到dir2
    for f in $(ls $displayall $dir1)
    do
        if [ $f == . ] || [ $f == .. ]
        then
            continue
        fi
        if [[ $f =~ ^TEMP_FILE_ ]]
        then
            :
        elif test -d "$dir1/$f"
        then
            scp -r "$dir1/$f" "$dir2/$f"
        else
            scp "$dir1/$f" "$dir2/$f"
        fi
    done

    #将dir1中的已匹配文件的文件名改回原名
    for f in $(ls $displayall $dir1)
    do
        if [[ $f =~ TEMPFILE_ ]]
        then

```

```

        mv "$dir1/$f" "$dir1/${f:10}"
    fi
done

#将dir2中的已匹配文件的文件名改回原名
for f in $(ls $displayall $dir2)
do
    if [[ $f =~ TEMP_FILE_ ]]
    then
        mv "$dir2/$f" "$dir2/${f:10}"
    fi
done
}

if [[ $mode == -c ]]
then
    myRsyncCopy $dir1 $dir2
else
    myRsyncSync $dir1 $dir2
fi

```

## Exp2-4

( 25分 ) 用bash编写程序 , 实现一个简单的作业管理系统。可以使用图形和数据库软件包来实现 , 也可以用文件形式实现数据的存储。系统至少具备以下的基本功能 :

系统中根据不同的权限分为三类用户 : 管理员、教师、学生 , 简要说明如下 :

- 管理员 :
  - 创建、修改、删除、显示 ( list ) 教师帐号 ; 教师帐户包括教师工号、教师姓名 , 教师用户以教师工号登录。
  - 创建、修改、删除课程 ; 绑定 ( 包括添加、删除 ) 课程与教师用户。课程名称以简单的中文或英文命名。
- 教师 :
  - 对某门课程 , 创建或导入、修改、删除学生帐户 , 根据学号查找学生帐号 ; 学生帐号的基本信息包括学号和姓名 , 学生使用学号登录。
  - 发布课程信息。包括新建、编辑、删除、显示 ( list ) 课程信息等功能。
  - 布置作业或实验。包括新建、编辑、删除、显示 ( list ) 作业或实验等功能。
  - 查找、打印所有学生的完成作业情况。
- 学生 :
  - 在教师添加学生账户后 , 学生就可以登录系统 , 并完成作业和实验。
  - 基本功能 : 新建、编辑作业或实验功能 ; 查询作业或实验完成情况。

## 需求描述

我们需要实现一个功能完整、界面友好的作业管理系统。简单而言 , 该系统需要具有如下特性 :

- 简洁、清晰的菜单指引
- 简单的用户系统
- 用户根据其不同身份使用不同功能
- 储存各类用户、课程、作业等信息，并能实现信息的增删查改

具体而言，该系统功能上的需求如下：

- 执行脚本后，输出详尽而美观的菜单提示，引导用户执行不同的操作，包括功能模块、返回上级、退出等选项。除最顶层菜单外，每一层都需要有返回上级的菜单选项。用户在菜单引导下进行登录操作，并根据不同的身份（管理员、教师、学生）进入不同的操作模式。
- 当用户身份为管理员时，用户可以管理用户信息和课程信息，或是修改自己的密码。其中，用户信息管理包括创建、修改、删除、显示教师帐号（包括工号、姓名和密码，每一项是否修改均可选，下同），其中显示教师账号支持按工号、姓名查找或是显示全部账号，按工号或姓名查找时，还支持模糊查找（即正则表达式匹配查找）；课程信息管理包括创建、修改、删除、查询课程，其中查询课程需要支持按课程号、课程名查找或显示全部课程，按课程号或课程名查找时，同样还需要支持模糊查找（即正则表达式匹配查找）；绑定、解绑课程与教师用户。
- 当用户身份为教师时，用户可以管理学生、课程、作业或实验信息，或是修改自己的密码。其中，学生管理包括创建、修改、删除、从文件导入学生账户（包括学生账号、密码、姓名），并支持根据学生账号查找学生账户；课程管理包括创建、修改、删除、显示全部课程信息（可以是一长段字符串，储存在文件中，每个课程可以有多条课程信息）；作业或实验信息管理包括创建、修改、删除、显示全部作业（包括起始时间、截止时间），能够打印所有选修该门课学生的完成情况，并能够查看指定学生的作业。

该系统数据上的需求如下：

- 账户：包括账号、密码、姓名、身份（管理员、教师、学生）。其中初始状态时可以有一个超级管理员super，用于添加其他管理员。
- 课程：包括课程号、课程名。
- 课程信息：课程下可以有课程信息，可以是一长段字符串，储存在文件中，每个课程可以有多条课程信息。
- 课程作业：包括起始时间、截止时间、文件路径。
- 授课信息、选课信息、作业提交信息等其他信息自行设计。

## 设计文档

### 1、设计思想

本实验的需求非常多，涉及到的数据类型也很多，因此设计的核心落在数据库设计上，即设计合适的数据库储存形式及关系来实现所有的需求。鉴于shell语言是一种脚本语言，拥有较为强大的文本处理能力，在设计思想上也采取了“万物皆是文件”的思想，数据库中的所有表以及表之间的关系全部用文件存储。对于课程信息、课程作业等文本量较大的属性，同样可以全部分开储存在单个文件中，并用一个catalog文件记录整体信息。

在代码实现时，主要采取了自顶向下层层调用的思想，整个程序的功能呈树状结构，因此菜单也就相应的有第一层、第二层、第三层等等。每进入下一层菜单相当于一次函数嵌套调用，返回上一级相当于退出函数返回上一层调用，这使得整个代码的结构十分清晰，实现时也较为方便。

在各部分功能的具体实现上，采用的思想是尽可能地利用Linux现有的指令进行处理，因为Linux的优势就在于强大的文本处理能力，采用现有指令进行文本处理不仅更为准确，而且大大减少了代码量，使得整个程序显得更为简洁。当然，对于较为复杂的文本处理，还是需要通过逐行读入文件内容来进行实现。

### 2、设计亮点

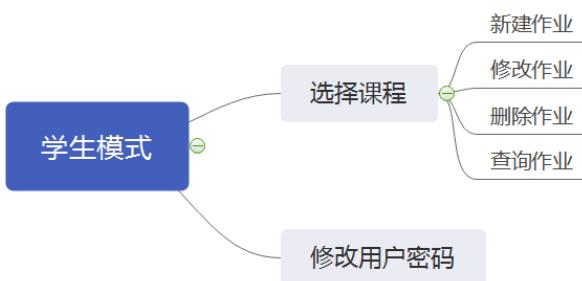
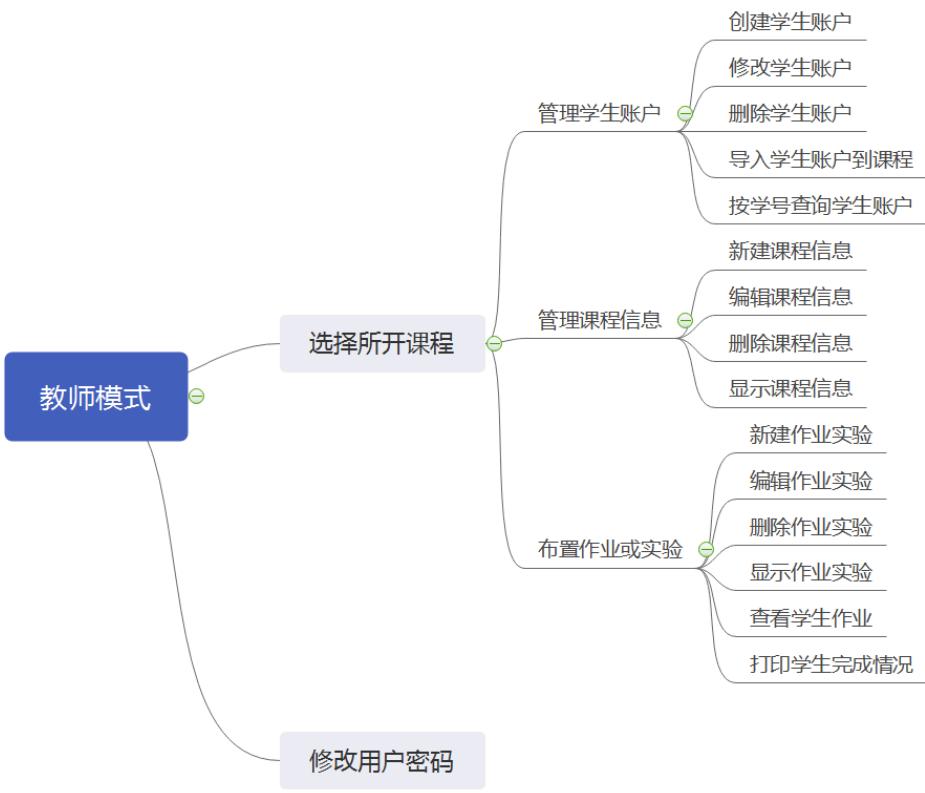
- 友好的用户界面与简洁、美观的菜单设计；
- **周全的特殊情况判断**（考虑了各种非法输入、非法操作等特殊情况，在设计时考虑的相当周全，具体可参考实验结果和代码部分）；
- **查询支持了按不同选项查找和模糊查找**（即正则表达式匹配）；
- **作业加入了起止时间**，对于不在开放时间内的作业禁止学生操作；
- 层层嵌套的设计模式增强了代码的结构性与可读性。

### 3、功能模块

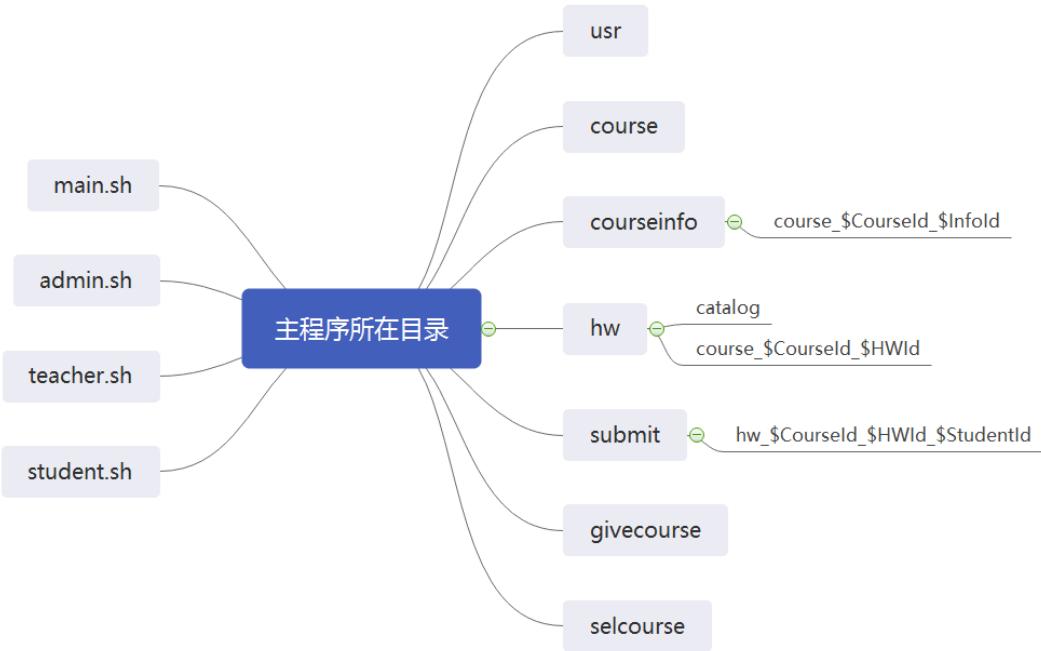
程序共分为4个子程序：main.sh、admin.sh、teacher.sh、student.sh。其中main.sh为主程序，会根据用户身份分别调用后面三个程序。

用户进入程序后，输入账号和密码登录，之后根据用户的权限进入不同的模式（管理员、教师、学生）。各模式的功能模块如下图所示。





文件（数据库）结构如下：



说明：

- usr 储存用户，包括账号、密码、权限、姓名；
- course 储存课程，包括课程号、课程名；
- courseinfo 储存课程信息，每条信息存放在该目录下的一个单独文件中，文件名中的\$CourseId 为课程号，InfoId 为课程信息编号；
- hw 储存课程作业信息，每个作业存放在该目录下的一个单独文件中，文件名中的\$CourseId 为课程号，HWId 为作业编号；目录下的catalog 文件中存放了作业对应课程号、作业编号及起止时间；
- submit 储存作业提交信息，每个作业的每个提交存放在该目录下的一个单独文件中，文件名中的 \$CourseId 为课程号，HWId 为作业编号，StudentId 为学生学号；
- givecourse 储存开课情况，包括教师工号、课程号；
- selcourse 储存选课情况，包括学生学号、课程号。

## 4、实现方法

- 用户、课程、教师对课程信息、作业的增删改查：
  - 增加：对于用户、课程等数据，直接用echo输出到文件结尾即可；对于课程信息、作业等数据，相当于在对应目录下新建一个文件，课程信息还需要将相应信息输出到catalog文件的末尾。
  - 修改：对于用户、课程等数据，先利用cat \$file | grep -v 从对应文件中选出未被修改的数据，将其写入到一个临时文件中，再将修改后的数据输出到临时文件的末尾，最后用mv 指令将临时文件改名回原文件名即可。对于课程信息、作业等数据，利用vi修改相应文件，课程信息还需要再利用类似的方法修改catalog文件。
  - 删除：对于用户、课程等数据，先利用cat \$file | grep -v 从对应文件中选出未被修改的数据，将其写入到一个临时文件中，再用mv 指令将临时文件改名回原文件名即可；对于课程信息、作业等数据，相当于在对应目录下删除一个文件，课程信息还需要从catalog文件中删除对应信息。**值得注意的是，删除的操作需要考虑数据的关联性，例如删除学生账户时，同时需要删除该学生提交的所有作业。**
  - 查询：对于用户、课程等数据，有两种实现方法：(1) 利用for循环逐行读入文件内容，与查询内容进行对比；(2) 利用cat \$file | grep 进行查找。前者适用于较复杂的查找，根据不同的场景选择不同的实现方式。对于课程信息、作业等数据，若存在catalog

文件则直接在该文件中查找，否则相当于查找目录下的文件名，有两种实现方式：(1)用find命令即可；(2)先利用for循环遍历ls \$path的结果，得到该目录下的每个文件名，再进行具体操作。后者适用于较复杂的查找，同样根据不同的场景选择不同的实现方式。

- 学生对作业的增删改查：
  - 增加：相当于在submit目录下新建一个文件。**注意此时需要特别判断提交时间是否在作业开放时间范围内，实现方式是先判断时间格式是否合法，再将时间利用date -d指令转化成时间戳之后再进行比较。**
  - 修改：相当于修改submit目录下的对应文件。
  - 删除：相当于删除submit目录下的对应文件。
  - 学生查询：利用cat指令将submit目录下的对应文件打印即可。
  - 教师查询：首先查询对应课程下所有的作业号（从hw/catalog中查询）；然后查询所有该课程的选课名单（从selcourse文件中查询）；最后对于每个作业，枚举每个学生，判断对应学生的作业是否存在，即判断submit目录下是否存在对应文件名的作业。
- 绑定/解绑教师与课程：在givecourse文件中添加/删除信息即可，添加则直接将新信息输出到文件末尾，否则利用cat \$file | grep -v从对应文件中选出未被修改的数据，将其写入到一个临时文件中，再用mv指令将临时文件改名回原文件名即可。
- 绑定学生与课程：除文件名变为selcourse外，其他同绑定/解绑教师与课程。

## 5、改进与讨论

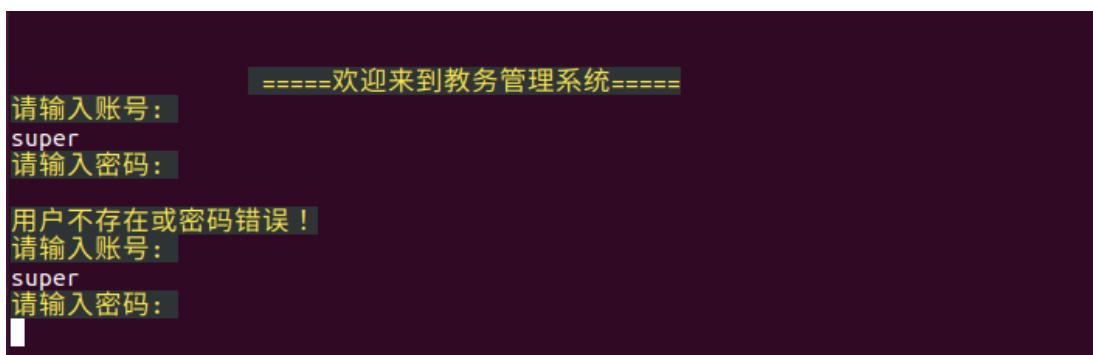
本程序在上述功能中都有着不错的效果，并且另外实现了第2节中的功能亮点，但仍有一些可改进之处，例如：

- 密码在储存时可以不储存明文，利用hash/加盐等多种方法进行处理；
- 对程序中途崩溃/中途退出的情况虽有考虑但并不周全，可以考虑添加一个日志文件进行记录便于恢复；
- 文件存储与SQL数据库存储的比较：文件存储有着发挥Linux文本处理能力强的内在特性的优势，储存方便，清晰简洁，便于各功能实现。相比之下SQL的优势在于数据一致性好、便于恢复、可以处理较大规模的数据。我认为两者都有各自的优势，对于用shell语言编写一个简易的作业管理系统这一实验，我认为使用前者更为清晰、简洁；但当系统功能逐渐复杂、数据规模逐渐增大之时，SQL则能更好地发挥出自身优势。

## 实验结果

### 1、主程序及管理员模式

1. 登录界面，另外实现了用户密码不回显：（初始状态时仅有一个超级管理员super，密码为super）



2. 管理员登录成功，进入初始菜单界面：

=====欢迎您, super ! =====

=====当前登录账号 : super 当前权限 : 管理员 =====

- 1)管理教师信息
- 2)管理课程信息
- 3)修改用户密码
- q)退出

请选择 :

3. 管理员进入管理教师信息界面，并新增二名教师。操作完成后，输入"!"返回上级。

=====当前登录账号 : super 当前权限 : 管理员 =====

- 1)管理教师信息
- 2)管理课程信息
- 3)修改用户密码
- q)退出

请选择 :

1

- 1)创建教师账号
- 2)修改教师信息
- 3)删除教师账号
- 4)查询教师信息
- b)返回上级
- q)退出

请选择 :

1

请输入教师工号 (输入'!'取消)

10001

请输入教师姓名 (输入'!'取消)

ZhangSan

请输入教师密码 (输入'!'取消)

10001ZS

添加成功 !

请输入教师工号 (输入'!'取消)

10002

请输入教师姓名 (输入'!'取消)

MikeGeorge

请输入教师密码 (输入'!'取消)

10002MG

添加成功 !

请输入教师工号（输入'!'取消）

!

- 1) 创建教师账号
- 2) 修改教师信息
- 3) 删除教师账号
- 4) 查询教师信息
- b) 返回上级
- q) 退出

请选择：

4. 管理员修改教师信息。此处演示的是管理员仅修改教师姓名，不修改教师密码的情况。

- 1) 创建教师账号
- 2) 修改教师信息
- 3) 删除教师账号
- 4) 查询教师信息
- b) 返回上级
- q) 退出

请选择：

2

请输入要修改的教师工号（输入'!'取消）

10001

请输入教师姓名（输入'!'取消，输入'/'不修改此项）

LiSi

请输入教师密码（输入'!'取消，输入'/'不修改此项）

/

修改成功！

5. 管理员查询教师信息，包括按工号查找、按姓名查找、查找全部记录，其中前两种查找方式均支持模糊查找即正则表达式匹配查找。

- 1)按工号查找
- 2)按姓名查找
- 3)查找全部记录
- b)返回上级
- q)退出

请选择 :

1

请输入教师工号 (输入 '!' 取消)

10\*

工号 : 10002 姓名 : MikeGeorge

工号 : 10001 姓名 : LiSi

查询成功, 共 2 条信息 !

请输入教师工号 (输入 '!' 取消)

2\$

工号 : 10002 姓名 : MikeGeorge

查询成功, 共 1 条信息 !

请输入教师工号 (输入 '!' 取消)

^2

查询失败, 无相关用户 !

- 1)按工号查找
- 2)按姓名查找
- 3)查找全部记录
- b)返回上级
- q)退出

请选择 :

2

请输入教师姓名 (输入'!'取消)

Lisi

工号 : 10001 姓名 : Lisi

查询成功, 共 1 条信息 !

请输入教师姓名 (输入'!'取消)

ZhangSan

查询失败, 无相关用户 !

请输入教师姓名 (输入'!'取消)

工号 : 10002 姓名 : MikeGeorge

工号 : 10001 姓名 : Lisi

查询成功, 共 2 条信息 !

- 1)按工号查找
- 2)按姓名查找
- 3)查找全部记录
- b)返回上级
- q)退出

请选择 :

3

工号 : 10002 姓名 : MikeGeorge

工号 : 10001 姓名 : Lisi

查询成功, 共 2 条记录 !

6. 删除教师 , 再进行查询后发现教师已不存在。

- 1) 创建教师账号
- 2) 修改教师信息
- 3) 删除教师账号
- 4) 查询教师信息
- b) 返回上级
- q) 退出

请选择 :

3

请输入要删除的教师工号 (输入'!'取消)

10001

删除成功 !

- 1) 创建教师账号
- 2) 修改教师信息
- 3) 删除教师账号
- 4) 查询教师信息
- b) 返回上级
- q) 退出

请选择 :

4

- 1) 按工号查找
- 2) 按姓名查找
- 3) 查找全部记录
- b) 返回上级
- q) 退出

请选择 :

3

工号 : 10002 姓名 : MikeGeorge

查询成功, 共 1 条记录 !

7. 进入课程管理界面, 新建课程。 ( 此处仅显示新增一门课程的结果, 实际新增了7门课程, 课程号分别为CS001-CS007 )

1)管理教师信息  
2)管理课程信息  
3)修改用户密码  
q)退出

请选择 :

2

1)创建课程  
2)修改课程  
3)删除课程  
4)查询课程  
5)绑定课程与教师账户  
6)解绑课程与教师账户  
b)返回上级  
q)退出

请选择 :

1

请输入课程号 (输入'!'取消)

CS001

请输入课程名称 (输入'!'取消)

ComputationTheory

添加成功 !

8. 进入课程管理界面 , 修改课程。

```
1)创建课程
2)修改课程
3)删除课程
4)查询课程
5)绑定课程与教师账户
6)解绑课程与教师账户
b)返回上级
q)退出

请选择：
2

请输入课程号（输入'!'取消）
CS005
请输入课程名称（输入'!'取消， 输入'/'不修改此项）
Python
修改成功！

请输入课程号（输入'!'取消）
CS008
请输入课程名称（输入'!'取消， 输入'/'不修改此项）
TensorFlow
修改失败，课程不存在！
```

9. 删除课程。

```
1)创建课程
2)修改课程
3)删除课程
4)查询课程
5)绑定课程与教师账户
6)解绑课程与教师账户
b)返回上级
q)退出

请选择：
3

请输入课程号（输入'!'取消）
CS003
删除成功！

请输入课程号（输入'!'取消）
CS008
删除失败，课程不存在！
```

10. 管理员查询课程信息，包括按课程号查找、按课程名查找、查找全部记录，其中前两种查找方式均支持模糊查找即正则表达式匹配查找。

1) 创建课程  
2) 修改课程  
3) 删除课程  
4) 查询课程  
5) 绑定课程与教师账户  
6) 解绑课程与教师账户  
b) 返回上级  
q) 退出

请选择 :

4

1) 按课程号查找  
2) 按课程名查找  
3) 查询所有课程  
b) 返回上级  
q) 退出

请选择 :

3

CS001 ComputationTheory  
CS002 DigitLogicDesign  
CS004 OOP  
CS006 Java  
CS007 Basic  
CS005 Python

查询成功, 共 6 条记录 !

1) 按课程号查找  
2) 按课程名查找  
3) 查询所有课程  
b) 返回上级  
q) 退出

请选择 :

1

请输入课程号 (输入'!'取消)

CS00[246]  
CS002 DigitLogicDesign  
CS004 OOP  
CS006 Java

查询成功, 共 3 条记录 !

请输入课程号 (输入'!'取消)

CS009

查询失败, 无相关课程 !

- 1)按课程号查找
- 2)按课程名查找
- 3)查询所有课程
- b)返回上级
- q)退出

请选择 :

2

请输入课程名称 (输入'!'取消)

D\*D

CS002 DigitLogicDesign

查询成功, 共 1 条记录 !

请输入课程名称 (输入'!'取消)

.Logic\*

CS002 DigitLogicDesign

查询成功, 共 1 条记录 !

请输入课程名称 (输入'!'取消)

OOP

CS004 OOP

查询成功, 共 1 条记录 !

11. 绑定教师与课程。当教师不存在或课程不存在或绑定记录已存在时均会报错。

- 1)创建课程
- 2)修改课程
- 3)删除课程
- 4)查询课程
- 5)绑定课程与教师账户
- 6)解绑课程与教师账户
- b)返回上级
- q)退出

请选择 :

5

请输入课程号 (输入'!'取消)

CS001

请输入教师账号 (输入'!'取消)

10001

添加成功 !

请输入课程号 (输入'!'取消)

CS002

请输入教师账号 (输入'!'取消)

100002

教师不存在 !

```
请输入课程号 (输入'!'取消)
CS008
请输入教师账号 (输入'!'取消)
10003
课程不存在！

请输入课程号 (输入'!'取消)
CS001
请输入教师账号 (输入'!'取消)
10001
添加失败，记录已存在！
```

12. 解绑教师与课程。当信息不存在时会报错。

```
1)创建课程
2)修改课程
3)删除课程
4)查询课程
5)绑定课程与教师账户
6)解绑课程与教师账户
b)返回上级
q)退出

请选择：
6

请输入课程号 (输入'!'取消)
CS001
请输入教师账号 (输入'!'取消)
10001
删除成功！

请输入课程号 (输入'!'取消)
CS002
请输入教师账号 (输入'!'取消)
10002
删除失败，记录不存在！

请输入课程号 (输入'!'取消)
CS001
请输入教师账号 (输入'!'取消)
10001
删除失败，记录不存在！
```

## 2、教师模式

1. 教师登录后进入主菜单，如下图所示。若教师需要对学生或课程信息进行管理，需要先选择课程，如下图所示。（此处显示的是初始状态即暂无开课的情况）

=====欢迎您， ZhangSan ! =====

=====当前登录账号：10001 当前权限：教师=====

- 1)管理所开课程
- 2)修改用户密码
- q)退出

请选择：

1  
您暂无开课，请联系管理员添加！

2. 用管理员身份课程后，再次登陆将显示当前教师所有开课，并要求用户从中选择课程。

=====欢迎您， ZhangSan ! =====

=====当前登录账号：10001 当前权限：教师=====

- 1)管理所开课程
- 2)修改用户密码
- q)退出

请选择：

1  
1) CS001 ComputationTheory  
您当前共有1门课程，请选择：（输入!返回）  
1  
当前操作课程：CS001 ComputationTheory

- 1)管理学生账户
- 2)管理课程信息
- 3)布置作业或实验
- b)返回上级
- q)退出

请选择：

3. 创建学生账户，如下图所示。（此处仅列出创建一名学生账户，实际新建了4名学生账户，学号分别为20001-20004）

1)管理学生账户  
2)管理课程信息  
3)布置作业或实验  
b)返回上级  
q)退出

请选择 :

1

1)创建学生账户  
2)修改学生账户  
3)删除学生账户  
4)导入学生账到课程  
5)按学号查询学生账户  
b)返回上级  
q)退出

请选择 :

1

请输入学生学号 (输入'!'取消)

20001

请输入学生姓名 (输入'!'取消)

hyw

请输入学生密码 (输入'!'取消)

hyw

添加成功 !

4. 修改学生账户 , 如下图所示。

1)创建学生账户  
2)修改学生账户  
3)删除学生账户  
4)导入学生账到课程  
5)按学号查询学生账户  
b)返回上级  
q)退出

请选择 :

2

请输入学生学号 (输入'!'取消)

20002

请输入学生姓名 (输入'!'取消, 输入'/'不修改此项)

mcz

请输入学生密码 (输入'!'取消, 输入'/'不修改此项)

/

修改成功 !

5. 删除学生账户 , 如下图所示。

```
1)创建学生账户
2)修改学生账户
3)删除学生账户
4)导入学生账户到课程
5)按学号查询学生账户
b)返回上级
q)退出

请选择：
3

请输入学生学号（输入'!'取消）
20003
删除成功！

请输入学生学号（输入'!'取消）
20005
删除失败，记录不存在！
```

6. 导入学生账户：先新建一个文件tmp，并输入学生信息，然后在管理界面中导入，如下图所示。

```
teacher.sh      ×          main.sh      ×          tmp      ×
1 20005 stu5 yhf
2 20006 stu6 cjo
3 20007 stu7 zsl

1)创建学生账户
2)修改学生账户
3)删除学生账户
4)导入学生账户到课程
5)按学号查询学生账户
b)返回上级
q)退出

请选择：
4

请输入要导入的文件路径（输入'!'取消）
./tmp
正在尝试导入：20005 yhf
添加成功！
正在尝试导入：20006 cjo
添加成功！
正在尝试导入：20007 zsl
添加成功！
导入完成，共成功导入 3 条记录！
```

7. 输入学生账号，即可按学号查询学生账户。当学生账户存在时，还会另外显示学生是否已选课。如下图所示。

```
1)创建学生账户  
2)修改学生账户  
3)删除学生账户  
4)导入学生账户到课程  
5)按学号查询学生账户  
b)返回上级  
q)退出

请选择：  
5

请输入学生学号（输入'!!'取消）  
20002  
学号：20002 姓名：mcz  
该学生已选课！

请输入学生学号（输入'!!'取消）  
20003  
学号：20003 姓名：myh  
该学生未选课！

请输入学生学号（输入'!!'取消）  
20009  
查询失败，用户不存在！
```

8. 在课程管理界面中选择新建课程信息。如下图所示。

```
当前操作课程：cs001 ComputationTheory

1)管理学生账户  
2)管理课程信息  
3)布置作业或实验  
b)返回上级  
q)退出

请选择：  
2

1)新建课程信息  
2)编辑课程信息  
3)删除课程信息  
4)显示课程信息  
b)返回上级  
q)退出

请选择：  
1  
请键入信息，输入一行一个字符'!!'结束！  
Test:  
Hello World!  
!  
新建成功，当前信息编号为1！
```

9. 在课程管理界面中选择编辑课程信息，输入课程信息编号后进入vi界面进行编辑。如下图所示。

- 1)新建课程信息
- 2)编辑课程信息
- 3)删除课程信息
- 4)显示课程信息
- b)返回上级
- q)退出

请选择：

2

请输入课程信息编号（输入'!'取消）

1

编辑成功！

10. 删除课程信息后查询课程信息，此时输出提示没有课程信息，说明已经成功删除。

```
1)新建课程信息
2)编辑课程信息
3)删除课程信息
4)显示课程信息
b)返回上级
q)退出

请选择：
3

请输入课程信息编号 (输入'!'取消)
1
删除成功！

请输入课程信息编号 (输入'!'取消)
!

1)新建课程信息
2)编辑课程信息
3)删除课程信息
4)显示课程信息
b)返回上级
q)退出

请选择：
4
查询失败，无课程信息！
```

11. ( 新建数条课程信息后 ) 显示所有课程信息 , 如下图所示。

```
3)删除课程信息
4)显示课程信息
b)返回上级
q)退出

请选择：
4
1:
Test:
Hello World!

2:
2
Hello World!

3:
3:
This is Message No.3!

4:
This is a three-line message.
This is a three-line message.
This is a three-line message.
```

12. 返回上级 , 进入“布置作业或实验”菜单 , 新建作业或实验 , 如下图所示。

```
1)管理学生账户  
2)管理课程信息  
3)布置作业或实验  
b)返回上级  
q)退出
```

请选择 :

3

```
1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出
```

请选择 :

1

请输入起始日期 (格式为YYYY-MM-DD, 输入'!!'取消)

2020-08-01

请输入结束日期 (格式为YYYY-MM-DD, 输入'!!'取消)

2020-08-20

请键入信息, 输入一行一个字符'!!'结束 !

Exp02

This is Exp02!

!

新建成功, 当前作业编号为2 !

13. 进入编辑作业实验界面，输入作业编号及相关信息，之后会进入vi界面修改作业内容，如下图所示。

```
Exp02  
This is Exp02!
```

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

" ./hw/course\_CS001\_2" 2 lines, 21 characters

1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

2

请输入作业实验编号 (输入'!'取消)

2

请输入作业起始时间 (格式为YYYY-MM-DD, 输入'!'取消, 输入'/'不修改此项)

/

请输入作业结束时间 (格式为YYYY-MM-DD, 输入'!'取消, 输入'/'不修改此项)

2020-08-25

编辑成功 !

请输入作业实验编号 (输入'!'取消)

3

请输入作业起始时间 (格式为YYYY-MM-DD, 输入'!'取消, 输入'/'不修改此项)

/

请输入作业结束时间 (格式为YYYY-MM-DD, 输入'!'取消, 输入'/'不修改此项)

/

编辑失败, 信息不存在 !

14. 显示作业实验 , 如下图所示。

3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

4

作业编号 : 1 起始日期 : 2020-08-01 结束日期 : 2020-08-31

Exp01

This is Exp01.

作业编号 : 2 起始日期 : 2020-08-01 结束日期 : 2020-08-25

Exp02

This is Exp02!

15. 删除作业实验 , 再进行显示发现作业已成功删除 , 如下图所示。

1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

3

请输入作业实验编号 (输入 '!' 取消)

3

删除失败，作业不存在！

请输入作业实验编号 (输入 '!' 取消)

1

删除成功！

1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

4

作业编号 : 2 起始日期 : 2020-08-01 结束日期 : 2020-08-25

Exp02

This is Exp02!

(注：查找、打印学生作业完成情况这一功能将在下一节中演示)

### 3、学生模式

1. 学生登录后进入主菜单，如下图所示。若学生需要对课程信息进行管理，那么同样需要先选择课程，如下图所示。

=====欢迎您， hyw ! =====

=====当前登录账号：20001 当前权限：学生=====

- 1)选择课程
- 2)修改用户密码
- q)退出

请选择：

1

1) CS001 ComputationTheory

您当前共有1门课程，请选择：(输入!返回)

1

当前操作课程：CS001 ComputationTheory

- 1)新建作业
- 2)编辑作业
- 3)删除作业
- 4)查询作业
- b)返回上级
- q)退出

请选择：

2. 选择课程后新建作业，如下图所示。若作业编号不存在会报错。

当前操作课程：CS001 ComputationTheory

- 1)新建作业
- 2)编辑作业
- 3)删除作业
- 4)查询作业
- b)返回上级
- q)退出

请选择：

1

请输入作业编号（输入'''返回）！

2

请键入信息，输入一行一个字符'''结束！

Test

!

新建成功，作业已保存至./submit/hw\_CS001\_2\_20001！

请选择：

1

请输入作业编号（输入'''返回）！

3

新建失败，作业不存在或不在提交时间！

3. 选择编辑作业，输入作业编号后进入vi编辑界面，保存并退出后将显示编辑成功。如下图所示。

- 1)新建作业
- 2)编辑作业
- 3)删除作业
- 4)查询作业
- b)返回上级
- q)退出

请选择：

2

请输入作业编号（输入'!'返回）！

2

编辑成功！

4. 查询作业，将显示作业的信息及提交状态，由于之前已经提交过该作业，因此作业状态为“已提交”，并且同时显示了作业的提交的路径。如下图所示。

- 1)新建作业
- 2)编辑作业
- 3)删除作业
- 4)查询作业
- b)返回上级
- q)退出

请选择：

4

作业编号 : 2 起始日期 : 2020-08-01 结束日期 : 2020-08-25  
作业状态 : 已提交, 路径为 ./submit/hw\_CS001\_2\_20001

Exp02

This is Exp02!

5. 进入删除作业界面，输入作业编号后即可删除作业，再使用查询作业功能，可以看到提交状态已经变为“未提交”，说明作业被成功删除，如下图所示。

```
1)新建作业  
2)编辑作业  
3)删除作业  
4)查询作业  
b)返回上级  
q)退出

请选择：  
3  
请输入作业编号（输入'!'返回）！  
2  
删除成功！

1)新建作业  
2)编辑作业  
3)删除作业  
4)查询作业  
b)返回上级  
q)退出

请选择：  
4  
作业编号：2 起始日期：2020-08-01 结束日期：2020-08-25  
作业状态：未提交

Exp02  
This is Exp02!
```

6. 教师界面打印学生完成情况：再次在学生模式下提交作业，然后用教师账号重新登录进入教师模式，进入“布置作业或实验-打印学生完成情况”，即显示所有学生的完成情况，可以看到学号“20001”对应的状态为“已提交”，说明作业提交成功。

```
1)新建作业  
2)编辑作业  
3)删除作业  
4)查询作业  
b)返回上级  
q)退出

请选择：  
1  
请输入作业编号（输入'!'返回）！  
2  
请键入信息，输入一行一个字符'!'结束！  
This is Homework2.  
!  
新建成功，作业已保存至./submit/hw_CS001_2_20001！
```

b)布置作业或实验  
b)返回上级  
q)退出

请选择 :

3

1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

6

请输入作业实验编号 (输入'!'取消)

2

学号 : 20001 状态 : 已提交  
学号 : 20002 状态 : 未提交  
学号 : 20004 状态 : 未提交  
学号 : 20005 状态 : 未提交  
学号 : 20006 状态 : 未提交  
学号 : 20007 状态 : 未提交

共有6名学生选课, 其中完成作业1人 !

7. 教师界面查找学生完成情况 : 输入对应的作业编号和学生学号 , 若学生已提交作业 , 则可以直接查看学生提交作业的内容 ; 否则将进行报错 , 如下图所示。

1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

5

请输入作业实验编号 (输入'!'取消)

2

请输入学生学号 (输入'!'取消)

20001

This is Homework2.

请输入作业实验编号 (输入'!'取消)

2

请输入学生学号 (输入'!'取消)

20002

查找失败, 作业不存在或学生未递交 !

8. 在教师界面修改作业提交截止时间至当前时间之前 , 再用学生账户登录后发现作业提交失败。  
( 实验时间为2020-08-13 , 在作业截止时间之后 )

1)新建作业实验  
2)编辑作业实验  
3)删除作业实验  
4)显示作业实验  
5)查看学生作业  
6)打印学生完成情况  
b)返回上级  
q)退出

请选择 :

1

请输入起始日期 (格式为YYYY-MM-DD, 输入'!!'取消)

2020-08-01

请输入结束日期 (格式为YYYY-MM-DD, 输入'!!'取消)

2020-08-10

请键入信息, 输入一行一个字符'!!'结束!

Exp01

This is Exp01 which is due on 2020-08-10.

!

新建成功, 当前作业编号为1!

=====欢迎您, hyw ! =====

=====当前登录账号 : 20001 当前权限 : 学生 =====

1)选择课程  
2)修改用户密码  
q)退出

请选择 :

1

1) CS001 ComputationTheory

您当前共有1门课程, 请选择: (输入!返回)

1

当前操作课程 : cs001 ComputationTheory

1)新建作业  
2)编辑作业  
3)删除作业  
4)查询作业  
b)返回上级  
q)退出

请选择 :

1

请输入作业编号 (输入'!!'返回) !

1

新建失败, 作业不存在或不在提交时间 !

9. 修改密码 : 输入旧密码和新密码即可修改密码, 若旧密码错误则修改失败。此处的密码同样做了不回显的处理。(在管理员模式和教师模式下也有此功能, 此处仅以学生模式下为例展示)

```
=====欢迎您， hyw ! =====

=====当前登录账号：20001 当前权限：学生=====

1)选择课程
2)修改用户密码
q)退出

请选择：
2

请输入旧密码（输入'!'取消）
请输入新密码（输入'!'取消）
修改成功！

请输入旧密码（输入'!'取消）
请输入新密码（输入'!'取消）
修改失败，旧密码错误！
```

## 完整代码

### main.sh

```
#!/bin/bash
#####
#程序名: main.sh (对应实验2第4题)
#作者: 黄彦玮
#学号: 3180102067
#说明: 图书管理系统-主程序
#完成时间: 2020-07-31
#####

#权限: 1:管理员 2:教师 3:学生
#检查账号密码是否匹配
check_pwd() {
    p_account=$1
    p_passwd=$2
    #超级管理员特判
    if [ $p_account = 'super' ] && [ $p_passwd = 'super' ]
    then
        name='super'
        return 1
    fi
    #判断账号信息是否存在
    usr_file=".:/usr"
    if [ ! -f $usr_file ]
    then
        return 0
    fi
    #扫描账号信息的文件, 对比账号密码
    while read ac pw au nm
    do
```

```

        if [ $ac = $p_account ] && [ $pw = $p_passwd ]
        then
            name=$nm
            return $au
        fi
    done <${usr_file}
    return 0
}

clear
echo -e "\n\n \t\t \033[40;93m =====欢迎来到教务管理系统===== \033[0m \t\t"
while true
do
    #读入账号
    echo -e "\033[40;93m请输入账号: \033[0m"
    #用户没输入就一直读
    while true
    do
        read account
        if test -n $account
        then
            break
        fi
    done
    #读入密码
    echo -e "\033[40;93m请输入密码: \033[0m"
    while true
    do
        read -s passwd
        if test -n $passwd
        then
            break
        fi
    done
    echo
    #检查账号密码是否匹配
    check_pwd $account $passwd
    auth=$?
    #账号不存在或密码错误
    if [ $auth -eq 0 ]
    then
        echo -e "\033[40;93m用户不存在或密码错误! \033[0m"
    else
        break
    fi
done

#根据权限执行不同的脚本
case $auth in
    1) ./admin.sh $account $name;;
    2) ./teacher.sh $account $name;;
    3) ./student.sh $account $name;;
esac

```

## admin.sh

```
#!/bin/bash
#####
#程序名: admin.sh (对应实验2第4题)
#作者: 黄彦玮
#学号: 3180102067
#说明: 图书管理系统-管理员部分
#完成时间: 2020-07-31
#####

#参数数量不等于2, 异常错误
if [ $# -ne 2 ]
then
    exit 1
fi

account=$1
name=$2
typeset -i itemcnt=0

#自定义读入, 忽略回车, 遇到输入的当行有字符即返回
myRead() {
    while true
    do
        read Buffer
        if [ -n $Buffer ]
        then
            break
        fi
        if [[ $Buffer =~ \ + ]]
        then
            echo -e "\033[40;93m输入含非法字符, 请重新输入! \033[0m"
        fi
    done
}

#新建教师账户
CreateTeacherAccount() {
    usr_file=". ./usr"
    touch ${usr_file}
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入教师工号 (输入'!' 取消) \033[0m"
        #读入教师工号
        myRead
        TeacherId=$Buffer
        #教师工号为'!'则返回
        if [ $TeacherId = '!' ]
        then
            return 0
        fi
    done
}
```

```

echo -e "\033[40;93m请输入教师姓名（输入'!'取消）\033[0m"
#读入教师姓名
myRead
TeacherName=$Buffer
#教师姓名为'!'则返回
if [ $TeacherName = '!' ]
then
    return 0
fi
echo -e "\033[40;93m请输入教师密码（输入'!'取消）\033[0m"
#读入教师密码
myRead
TeacherPwd=$Buffer
#教师密码为'!'则返回
if [ $TeacherPwd = '!' ]
then
    return 0
fi
flag=0
#扫描账号信息的文件，判断账号是否存在
while read ac pw au nm
do
    #账号已存在，插入失败
    if [ $ac = $TeacherId ]
    then
        flag=1
        echo -e "\033[40;93m添加失败，账号已存在！\033[0m"
        break
    fi
done <${usr_file}
#账号不存在，插入成功
if [ $flag -eq 0 ]
then
    echo $TeacherId $TeacherPwd 2 $TeacherName >>${usr_file}
    echo -e "\033[40;93m添加成功！\033[0m"
fi
done
}

#修改教师账户
ModifyTeacherAccount () {
usr_file=".:/usr"
touch ${usr_file}
while true
do
    echo -e "\n"
    echo -e "\033[40;93m请输入要修改的教师工号（输入'!'取消）\033[0m"
    #读入教师工号
    myRead
    TeacherId=$Buffer
    #教师工号为'!'则返回
    if [ $TeacherId = '!' ]
    then
        return 0
    fi
}

```

```

echo -e "\033[40;93m请输入教师姓名（输入'!'取消，输入'/'不修改此项）\033[0m"
#读入教师姓名
myRead
TeacherName=$Buffer
#教师姓名为'!'则返回
if [ $TeacherName = '!' ]
then
    return 0
fi
echo -e "\033[40;93m请输入教师密码（输入'!'取消，输入'/'不修改此项）\033[0m"
#读入教师密码
myRead
TeacherPwd=$Buffer
#教师密码为'!'则返回
if [ $TeacherPwd = '!' ]
then
    return 0
fi
flag=0
#扫描账号信息的文件，判断账号是否存在
while read ac pw au nm
do
    #账号已存在，修改
    if [ $ac = $TeacherId ]
    then
        flag=1
        #权限非教师，删除失败
        if [ $au -ne 2 ]
        then
            echo -e "\033[40;93m修改失败，操作的用户非教师用户！\033[0m"
        else
            #若不为'/'则修改
            if [ $TeacherPwd != '/' ]
            then
                pw=$TeacherPwd
            fi
            if [ $TeacherName != '/' ]
            then
                nm=$TeacherName
            fi
            #先删除工号对应的行
            cat ${usr_file} | grep -v "^$ac " >> ${usr_file}Tmp
            mv ${usr_file}Tmp ${usr_file}
            #再新增一行表示新的账号信息
            echo $ac $pw $au $nm >>${usr_file}
            echo -e "\033[40;93m修改成功！\033[0m"
        fi
        break
    fi
done <${usr_file}
#账号不存在，修改失败
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m修改失败，用户信息不存在！\033[0m"
fi

```

```

        done
    }

#删除教师账户
DeleteTeacherAccount() {
    usr_file=". /usr"
    touch ${usr_file}
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入要删除的教师工号（输入'!'取消）\033[0m"
        #读入教师工号
        myRead
        TeacherId=$Buffer
        #教师工号为'!'则返回
        if [ $TeacherId = '!' ]
        then
            return 0
        fi
        flag=0
        #扫描账号信息的文件，判断账号是否存在
        while read ac pw au nm
        do
            #账号已存在，删除
            if [ $ac = $TeacherId ]
            then
                flag=1
                if [ $au -ne 2 ]
                then
                    echo -e "\033[40;93m删除失败，操作的用户非教师用户！\033[0m"
                else
                    #删除工号对应的行
                    cat ${usr_file} | grep -v ^$TeacherId >> ${usr_file}Tmp
                    mv ${usr_file}Tmp ${usr_file}
                    echo -e "\033[40;93m删除成功！\033[0m"
                fi
                break
            fi
        done <${usr_file}
        #账号不存在，删除失败
        if [ $flag -eq 0 ]
        then
            echo -e "\033[40;93m删除失败，用户不存在！\033[0m"
        fi
    done
}

#按账号查询教师
QueryTeacherAccountById() {
    usr_file=". /usr"
    touch ${usr_file}
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入教师工号（输入'!'取消）\033[0m"

```

```

#读入教师工号
myRead
TeacherId=$Buffer
#教师工号为'!'则返回
if [ $TeacherId = '!' ]
then
    return 0
fi
#满足要求的记录数
itemcnt=0
#扫描账号信息的文件，判断账号是否存在
while read ac pw au nm
do
    #账号已存在且权限正确，输出相关信息
    if [[ $ac =~ $TeacherId ]] && [ $au -eq 2 ]
    then
        ((itemcnt++))
        echo -e "\033[40;93m工号: $ac 姓名: $nm\033[0m"
    fi
done <${usr_file}
#无符号要求信息，查询失败
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m查询失败，无相关用户！\033[0m"
#查询成功，输出信息总数
else
    echo -e "\033[40;93m查询成功，共 $itemcnt 条信息！\033[0m"
fi
done
}

#按姓名查询教师
QueryTeacherAccountByName() {
    usr_file=". /usr"
    touch ${usr_file}
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入教师姓名（输入'!'取消）\033[0m"
        #读入教师姓名
        myRead
        TeacherName=$Buffer
        #教师工号为'!'则返回
        if [ $TeacherName = '!' ]
        then
            return 0
        fi
        #满足要求的记录数
        itemcnt=0
        #扫描账号信息的文件，判断账号是否存在
        while read ac pw au nm
        do
            #账号已存在且权限正确，输出相关信息
            if [[ $nm =~ $TeacherName ]] && [ $au -eq 2 ]
            then

```

```

        ((itemcnt++))
        echo -e "\033[40;93m工号: $ac 姓名: $nm\033[0m"
    fi
done <${usr_file}
#无符号要求信息，查询失败
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m查询失败，无相关用户！\033[0m"
#查询成功，输出信息总数
else
    echo -e "\033[40;93m查询成功，共 $itemcnt 条信息！\033[0m"
fi
done
}

#查询全部教师账户
QueryTeacherAccountALL() {
    usr_file=".:/usr"
    touch ${usr_file}
#满足要求的记录数
itemcnt=0
#扫描账号信息的文件，判断账号是否存在
while read ac pw au nm
do
#账号已存在且权限正确，输出相关信息
    if [ $au -eq 2 ]
    then
        ((itemcnt++))
        echo -e "\033[40;93m工号: $ac 姓名: $nm\033[0m"
    fi
done <${usr_file}
#无符号要求信息，查询失败
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m查询失败，无相关用户！\033[0m"
#查询成功，输出信息总数
else
    echo -e "\033[40;93m查询成功，共 $itemcnt 条记录！\033[0m"
    return 0
fi
return 0
}

#查询教师账户
QueryTeacherAccount() {
while true
do
    echo -e "\n"
    echo -e "\033[40;93m\t"1\t\033[0m按工号查找"\033[0m"
    echo -e "\033[40;93m\t"2\t\033[0m按姓名查找"\033[0m"
    echo -e "\033[40;93m\t"3\t\033[0m查找全部记录"\033[0m"
    echo -e "\033[40;93m\t"b\t\033[0m返回上级"\033[0m"
    echo -e "\033[40;93m\t"q\t\033[0m退出"\033[0m"
    echo -e "\n"
    echo -e "\033[40;93m请选择: "\033[0m"

```

```

    read choice
    case $choice in
        1) QueryTeacherAccountById;;
        2) QueryTeacherAccountByName;;
        3) QueryTeacherAccountAll;;
        b) return 0;;
        q) exit 0;;
        *) echo -e "\033[40;93m"输入非法, 请重新输入! "\033[0m";;
    esac
done
}

#教师信息管理
ManT() {
while true
do
    echo -e "\n"
    echo -e "\033[40;93m\t"1\)\ 创建教师账号"\033[0m"
    echo -e "\033[40;93m\t"2\)\ 修改教师信息"\033[0m"
    echo -e "\033[40;93m\t"3\)\ 删除教师账号"\033[0m"
    echo -e "\033[40;93m\t"4\)\ 查询教师信息"\033[0m"
    echo -e "\033[40;93m\t"b\)\ 返回上级"\033[0m"
    echo -e "\033[40;93m\t"q\)\ 退出"\033[0m"
    echo -e "\n"
    echo -e "\033[40;93m请选择: "\033[0m"
read choice
case $choice in
    1) CreateTeacherAccount;;
    2) ModifyTeacherAccount;;
    3) DeleteTeacherAccount;;
    4) QueryTeacherAccount;;
    b) return 0;;
    q) exit 0;;
    *) echo -e "\033[40;93m"输入非法, 请重新输入! "\033[0m";;
esac
done
}

#新建课程
CreateCourse() {
cfile="../course"
touch $cfile
while true
do
    echo -e "\n"
    echo -e "\033[40;93m请输入课程号 (输入'!'取消) \033[0m"
    #读入课程号
    myRead
    CourseId=$Buffer
    #教师姓名为'!'则返回
    if [ $CourseId = '!' ]
    then
        return 0
    fi
    echo -e "\033[40;93m请输入课程名称 (输入'!'取消) \033[0m"
}

```

```

#读入课程名称
myRead
CourseName=$Buffer
#课程号名称'!'则返回
if [[ $CourseName = '!' ]]
then
    return 0
fi
flag=0
#扫描课程信息的文件，判断课程是否存在
while read cid nm
do
    #课程已存在
    if [[ $cid = $CourseId ]]
    then
        flag=1
        echo -e "\033[40;93m添加失败，课程号已存在！\033[0m"
        break
    fi
done <${cfile}

#课程不存在，添加课程
if [[ $flag -eq 0 ]]
then
    #将新课程添加到数据文件的末尾
    echo $CourseId $CourseName >> ${cfile}
    echo -e "\033[40;93m添加成功！\033[0m"
fi
done
}

#修改课程
ModifyCourse() {
    cfile="./course"
    touch $cfile
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程号（输入'!'取消）\033[0m"
        #读入课程号
        myRead
        CourseId=$Buffer
        #教师姓名为'!'则返回
        if [[ $CourseId = '!' ]]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入课程名称（输入'!'取消，输入'/'不修改此项）\033[0m"
        #读入课程名称
        myRead
        CourseName=$Buffer
        #课程号名称'!'则返回
        if [[ $CourseName = '!' ]]
        then
            return 0
        fi
    done
}

```

```

        fi
        flag=0
#扫描课程信息的文件，判断课程是否存在
while read cid nm
do
    #课程存在，进行修改
    if [ $cid = $CourseId ]
    then
        flag=1
        if [[ $CourseName != '/' ]]
        then
            nm=$CourseName
        fi
        #先删除课程号对应的行
        cat $cfile | grep -v "^$cid " >> ${cfile}Tmp
        mv ${cfile}Tmp ${cfile}
        #再新增一行表示新的账号信息
        echo $cid $nm >>${cfile}
        echo -e "\033[40;93m修改成功！\033[0m"
        break
    fi
done <${cfile}
#课程不存在，修改失败
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m修改失败，课程不存在！\033[0m"
fi
done
}

```

```

#删除课程
DeleteCourse() {
    cfile="../course"
    touch $cfile
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程号（输入'!'取消）\033[0m"
        #读入课程号
        myRead
        CourseId=$Buffer
        #课程号为'!'则返回
        if [[ $CourseId = '!' ]]
        then
            return 0
        fi
        flag=0
#扫描账号信息的文件，判断账号是否存在
while read cid nm
do
    #账号已存在，删除
    if [ $cid = $CourseId ]
    then
        flag=1
        #删除账号对应的行

```

```

        cat ${cfile} | grep -v "^\$CourseId " >> ${cfile}Tmp
        mv ${cfile}Tmp ${cfile}
        echo -e "\033[40;93m删除成功! \033[0m"
        break
    fi
done <${cfile}
#账号不存在，删除失败
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m删除失败，课程不存在! \033[0m"
fi
done
}

#按课程号查询课程
QueryCourseById() {
    cfile="./course"
    touch $cfile
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程号（输入'!'取消）\033[0m"
        #读入课程号
        myRead
        CourseId=$Buffer
        #课程号为'!'则返回
        if [ $CourseId = '!' ]
        then
            return 0
        fi
        #符合条件的记录数
        itemcnt=0
        #扫描课程信息的文件
        while read cid nm
        do
            #课程存在，输出记录
            if [[ $cid =~ $CourseId ]]
            then
                ((itemcnt++))
                echo -e "\033[40;93m$cid $nm\033[0m"
            fi
        done <${cfile}
        #课程不存在，删除失败
        if [ $itemcnt -eq 0 ]
        then
            echo -e "\033[40;93m查询失败，无相关课程! \033[0m"
        else
            echo -e "\033[40;93m查询成功，共 $itemcnt 条记录! \033[0m"
        fi
    done
}

#按课程名查询课程
QueryCourseByName() {
    cfile="./course"

```

```

touch $cfile
while true
do
    echo -e "\n"
    echo -e "\033[40;93m请输入课程名称（输入'!'取消）\033[0m"
    #读入课程名称
    myRead
    CourseName=$Buffer
    #课程名称为'!'则返回
    if [ $CourseName = '!' ]
    then
        return 0
    fi
    #符合条件的记录数
    itemcnt=0
    #扫描课程信息的文件
    while read cid nm
    do
        #课程存在，输出记录
        if [[ $nm =~ $CourseName ]]
        then
            ((itemcnt++))
            echo -e "\033[40;93m$cid $nm\033[0m"
        fi
    done <${cfile}
    #课程不存在
    if [ $itemcnt -eq 0 ]
    then
        echo -e "\033[40;93m查询失败，无相关课程！\033[0m"
    else
        echo -e "\033[40;93m查询成功，共$itemcnt 条记录！\033[0m"
    fi
done
}

#查询全部课程
QueryCourseALL() {
    cfile=./course
    touch $cfile
    while true
    do
        #符合条件的记录数
        itemcnt=0
        #扫描课程信息的文件
        while read cid nm
        do
            #输出记录
            ((itemcnt++))
            echo -e "\033[40;93m$cid $nm\033[0m"
        done <${cfile}
        #课程不存在
        if [ $itemcnt -eq 0 ]
        then
            echo -e "\033[40;93m查询失败，无相关课程！\033[0m"
        else

```

```

        echo -e "\033[40;93m查询成功，共 $itemcnt 条记录！\033[0m"
        return 0
    fi
done
}

#查询课程信息
QueryCourse() {
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m\t"1\033[0m)按课程号查找"\033[0m"
        echo -e "\033[40;93m\t"2\033[0m)按课程名查找"\033[0m"
        echo -e "\033[40;93m\t"3\033[0m)查询所有课程"\033[0m"
        echo -e "\033[40;93m\t"b\033[0m)返回上级"\033[0m"
        echo -e "\033[40;93m\t"q\033[0m)退出"\033[0m"
        echo -e "\n"
        echo -e "\033[40;93m请选择：" "\033[0m"
        read choice
        case $choice in
            1) QueryCourseById;;
            2) QueryCourseByName;;
            3) QueryCourseALL;;
            b) return 0;;
            q) exit 0;;
            *) echo -e "\033[40;93m输入非法，请重新输入！"\033[0m;;
        esac
    done
}

#绑定课程和教师账号
LinkCourseAndTeacher() {
    cfile="./givecourse"
    touch $cfile
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程号（输入'!'取消）\033[0m"
        #读入课程号
        myRead
        CourseId=$Buffer
        #教师姓名为'!'则返回
        if [ $CourseId = '!' ]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入教师账号（输入'!'取消）\033[0m"
        #读入教师账号
        myRead
        TeacherId=$Buffer
        #课程号名称'!'则返回
        if [ $TeacherId = '!' ]
        then
            return 0
        fi

```

```

#检查课程是否存在
flag=$(cat "./course" | grep "^$CourseId" | wc -l)
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m课程不存在! \033[0m"
    continue
fi
#检查教师是否存在
flag=$(cat "./usr" | grep "^$TeacherId" | wc -l)
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m教师不存在! \033[0m"
    continue
fi
#扫描选课信息的文件
itemcnt=$(cat $cfile | grep "$CourseId\\ $TeacherId" | wc -l)
#选课信息不存在，添加信息
if [ $itemcnt -eq 0 ]
then
    #将新课程添加到数据文件的末尾
    echo $CourseId $TeacherId >> $cfile
    echo -e "\033[40;93m添加成功! \033[0m"
else
    echo -e "\033[40;93m添加失败，记录已存在! \033[0m"
fi
done
}

#解绑课程与教师账号
CancelLinkCourseAndTeacher() {
    cfile="./givecourse"
    touch $cfile
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程号（输入'!'取消）\033[0m"
        #读入课程号
        myRead
        CourseId=$Buffer
        #教师姓名为'!'则返回
        if [ $CourseId = '!' ]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入教师账号（输入'!'取消）\033[0m"
        #读入教师账号
        myRead
        TeacherId=$Buffer
        #课程号名称'!'则返回
        if [ $TeacherId = '!' ]
        then
            return 0
        fi
        flag=0
        #扫描选课信息的文件

```

```

itemcnt=$(cat $cfile | grep "^\$CourseId\$TeacherId" | wc -l)
#选课信息不存在，删除失败
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m删除失败，记录不存在！\033[0m"
else
    #利用反向选择删除绑定信息
    cat $cfile | grep -v "^\$CourseId\$TeacherId" >> "${cfile}tmp"
    mv "${cfile}tmp" "$cfile"
    echo -e "\033[40;93m删除成功！\033[0m"
fi
done
}

#课程信息管理
ManC() {
while true
do
    echo -e "\n"
    echo -e "\033[40;93m\t1\033[0m) 创建课程\033[0m"
    echo -e "\033[40;93m\t2\033[0m) 修改课程\033[0m"
    echo -e "\033[40;93m\t3\033[0m) 删除课程\033[0m"
    echo -e "\033[40;93m\t4\033[0m) 查询课程\033[0m"
    echo -e "\033[40;93m\t5\033[0m) 绑定课程与教师账户\033[0m"
    echo -e "\033[40;93m\t6\033[0m) 解绑课程与教师账户\033[0m"
    echo -e "\033[40;93m\tb\033[0m) 返回上级\033[0m"
    echo -e "\033[40;93m\tq\033[0m) 退出\033[0m"
    echo -e "\n"
    echo -e "\033[40;93m请选择：\033[0m"
    read choice
    case $choice in
        1) CreateCourse;;
        2) ModifyCourse;;
        3) DeleteCourse;;
        4) QueryCourse;;
        5) LinkCourseAndTeacher;;
        6) CancelLinkCourseAndTeacher;;
        b) return 0;;
        q) exit 0;;
        *) echo -e "\033[40;93m输入非法，请重新输入！\033[0m";;
    esac
done
}

#修改密码
ChangePwd() {
cfile=".:/usr"
touch cfile
if [ $account = "super" ]
then
    echo -e "\033[40;93m当前账户不可修改密码！\033[0m"
    return 0
fi
while true
do

```

```

echo -e "\n"
echo -e "\033[40;93m请输入旧密码（输入'!'取消）\033[0m"
read -s oldpwd
echo
if [[ $oldpwd = '!' ]]
then
    return 0
fi
echo -e "\033[40;93m请输入新密码（输入'!'取消）\033[0m"
read -s newpwd
echo
if [[ $newpwd = '!' ]]
then
    return 0
fi
#扫描账号信息的文件，对比账号密码
while read ac pw au nm
do
    if [[ $ac = $account ]]
    then
        if [[ $pw = $oldpwd ]]
        then
            pw=$newpwd
            #从旧文件中删除原信息
            cat $cfile | grep -v "^$account" > "${cfile}Tmp"
            mv "${cfile}Tmp" "$cfile"
            #将新信息附到文件最后
            echo $ac $pw $au $nm >> "$cfile"
            echo -e "\033[40;93m修改成功！\033[0m"
            return 0
        else
            echo -e "\033[40;93m修改失败，旧密码错误！\033[0m"
            break
        fi
    fi
done <${cfile}
done
}

clear
echo -e "\n\n \t\t\t \033[40;93m =====欢迎您， $name ! =====\033[0m"
echo -e "\n\n \t\t \033[40;93m=====当前登录账号： $account  当前权限： 管理员===== \033[0m"

while true
do
    echo -e "\n"
    echo -e "\033[40;93m\t"1\033[0m") 管理教师信息"\033[0m"
    echo -e "\033[40;93m\t"2\033[0m") 管理课程信息"\033[0m"
    echo -e "\033[40;93m\t"3\033[0m") 修改用户密码"\033[0m"
    echo -e "\033[40;93m\t"q\033[0m") 退出"\033[0m"
    echo -e "\n"
    echo -e "\033[40;93m"请选择："\033[0m"
read choice
case $choice in
    1) ManT;;

```

```

    2) ManC;;
    3) ChangePwd;;
    q) exit 0;;
*) echo -e "\033[40;93m输入非法, 请重新输入! \033[0m";;
esac
done

```

## teacher.sh

```

#!/bin/bash
#####
#程序名: teacher.sh (对应实验2第4题)
#作者: 黄彦玮
#学号: 3180102067
#说明: 图书管理系统-教师部分
#完成时间: 2020-07-31
#####

#参数数量不等于2, 异常错误
if [ $# -ne 2 ]
then
    exit 1
fi

account=$1
name=$2
typeset -i itemcnt=0
typeset -i itemcnttwo=0

#自定义读入, 忽略回车, 遇到输入的当行有字符即返回
myRead() {
    while true
    do
        read Buffer
        if [ -n $Buffer ]
        then
            break
        fi
        if [[ $Buffer =~ \+ ]]
        then
            echo -e "\033[40;93m输入含非法字符, 请重新输入! \033[0m"
        fi
    done
}

#日期格式检查
DateCheck() {
    #利用date命令判断日期是否合法
    if echo $1 | grep -Eq "[0-9]{4}-[0-9]{2}-[0-9]{2}" && date -d $1 +%Y%m%d > /dev/null 2>&1
    then
        return 1
    else
        return 0
}

```

```

        fi
    }

#读入时间
myReadTime() {
    #参数表示是否需要过滤'/'(用于编辑作业)
    tmp=$#
    while true
    do
        read Buffer
        #调用函数判断读入日期是否合法
        DateCheck $Buffer
        #日期是否合法
        isTimeOK=$?
        #无特殊过滤要求
        if [ $tmp -eq 0 ]
        then
            if [ $isTimeOK -ne 1 ] && [ $Buffer != '!' ]
            then
                echo -e "\033[40;93m格式错误, 请重新输入! \033[0m"
            else
                break
            fi
        else
            if [ $isTimeOK -ne 1 ] && [ $Buffer != '!' ] && [ $Buffer != '/' ]
            then
                echo -e "\033[40;93m格式错误, 请重新输入! \033[0m"
            else
                break
            fi
        fi
    done
}

#新建/导入一名学生的账户(在新建账户和导入账户的函数中调用)
CreateOneStudent() {
    StudentId=$1
    StudentPwd=$2
    StudentName=$3
    CourseId=$4
    cfile=".selcourse"
    usr_file=".usr"
    flag=0
    touch $cfile
    touch ${usr_file}
    #判断学生账户是否存在
    while read ac pw au nm
    do
        #账号已存在
        if [ $ac = $StudentId ]
        then
            flag=1
            #判断其他信息是否一致, 若不一致则报错
            if [ $pw != $StudentPwd ] || [ $au -ne 3 ] || [ $nm != $StudentName ]
            then

```

```

        echo -e "\033[40;93m添加失败，账号信息与已有数据不符，请联系学生核对！\033[0m"
        return 0
    fi
    break
fi
done < ${usr_file}

#账号不存在，则需要先新建账号
if [ $flag -eq 0 ]
then
    echo $StudentId $StudentPwd 3 $StudentName >> ${usr_file}
fi

#查询选课信息是否已经存在
itemcnt=$(cat $cfile | grep "^\$CourseId\$ $StudentId\$" | wc -l)
#选课信息已经存在，报错
if [ $itemcnt -ne 0 ]
then
    echo -e "\033[40;93m添加失败，选课信息已存在！\033[0m"
    return 0
fi

#添加选课信息
echo $CourseId $StudentId >> $cfile
echo -e "\033[40;93m添加成功！\033[0m"
return 1
}

```

```

#新建学生账户
CreateStudent() {
    CourseId=$1
    cfile="./selcourse"
    usr_file="./usr"
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入学生学号（输入'!'取消）\033[0m"
        #读入学生学号
        myRead
        StudentId=$Buffer
        #学生学号为'!'则返回
        if [ $StudentId = '!' ]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入学生姓名（输入'!'取消）\033[0m"
        #读入学生姓名
        myRead
        StudentName=$Buffer
        #学生姓名为'!'则返回
        if [ $StudentName = '!' ]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入学生密码（输入'!'取消）\033[0m"
        #读入学生密码

```

```

myRead
StudentPwd=$Buffer
#学生密码为'!'则返回
if [ $StudentPwd = '!' ]
then
    return 0
fi
#调用函数插入账户
CreateOneStudent $StudentId $StudentPwd $StudentName $CourseId
done
}

#修改学生账户
ModifyStudent () {
usr_file=".:/usr"
while true
do
    echo -e "\n"
    echo -e "\033[40;93m请输入学生学号（输入'!'取消）\033[0m"
    #读入学生学号
    myRead
    StudentId=$Buffer
    #学生学号为'!'则返回
    if [ $StudentId = '!' ]
    then
        return 0
    fi
    echo -e "\033[40;93m请输入学生姓名（输入'!'取消，输入'/'不修改此项）\033[0m"
    #读入学生姓名
    myRead
    StudentName=$Buffer
    #学生姓名为'!'则返回
    if [ $StudentName = '!' ]
    then
        return 0
    fi
    echo -e "\033[40;93m请输入学生密码（输入'!'取消，输入'/'不修改此项）\033[0m"
    #读入学生密码
    myRead
    StudentPwd=$Buffer
    #学生密码为'!'则返回
    if [ $StudentPwd = '!' ]
    then
        return 0
    fi
    flag=0
    #判断学生账户是否存在
    while read ac pw au nm
    do
        #账号已存在
        if [ $ac = $StudentId ]
        then
            flag=1
            #特判不修改的情况
            if [ $StudentPwd != '/' ]

```

```

        then
            pw=$StudentPwd
        fi
        if [ $StudentName != '/' ]
        then
            nm=$StudentName
        fi
        #先删除旧信息，再添加新信息
        cat ${usr_file} | grep -v "^$ac" >> "${usr_file}Tmp"
        mv "${usr_file}Tmp" "${usr_file}"
        echo $ac $pw $au $nm >> ${usr_file}
        echo -e "\033[40;93m修改成功！\033[0m"
        break
    fi
done <${usr_file}
#账号不存在，修改失败
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m修改失败，用户不存在！\033[0m"
fi
done
}

#删除学生账户
DeleteStudent () {
    CourseId=$1
    cfile="./selcourse"
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入学生学号（输入'!'取消）\033[0m"
        #读入学生学号
        myRead
        StudentId=$Buffer
        #学生学号为'!'则返回
        if [ $StudentId = '!' ]
        then
            return 0
        fi
        #查询记录是否存在
        itemcnt=$(cat $cfie | grep "^$CourseId\$ $StudentId\$" | wc -l)
        #若不存在报错，若存在则删除
        if [ $itemcnt -eq 0 ]
        then
            echo -e "\033[40;93m删除失败，记录不存在！\033[0m"
        else
            cat $cfie | grep -v "^$CourseId\$ $StudentId\$" >> "${cfie}Tmp"
            mv "${cfie}Tmp" "${cfie}"
            echo -e "\033[40;93m删除成功！\033[0m"
        fi
    done
}

#导入学生账户
ImportStudent () {

```

```

CourseId=$1
cfile=".selcourse"
while true
do
    echo -e "\n"
    echo -e "\033[40;93m请输入要导入的文件路径（输入'!'取消）\033[0m"
    #读入文件路径
    myRead
    FilePath=$Buffer
    #文件路径为'!'则返回
    if [ $FilePath = '!' ]
    then
        return 0
    fi
    #文件不存在则报错
    if [ ! -f $FilePath ]
    then
        echo -e "\033[40;93m导入失败，文件不存在！\033[0m"
    fi
    #插入成功的记录数
    tmp=0
    #读取文件内容并尝试插入
    while read ac pw nm
    do
        echo -e "\033[40;93m正在尝试导入：$ac $nm\033[0m"
        #调用函数插入
        CreateOneStudent $ac $pw $nm $CourseId
        #插入成功，则加入统计信息
        if [ $? -eq 1 ]
        then
            ((tmp++))
        fi
    done <$FilePath
    echo -e "\033[40;93m导入完成，共成功导入 $tmp 条记录！\033[0m"
done
}

#按学号查询学生账户
QueryStudentById() {
    CourseId=$1
    cfile=".selcourse"
    usr_file=".usr"
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入学生学号（输入'!'取消）\033[0m"
        #读入学生学号
        myRead
        StudentId=$Buffer
        #学生学号为'!'则返回
        if [ $StudentId = '!' ]
        then
            return 0
        fi
        flag=0

```

```

#查询学生账户信息
while read ac pw au nm
do
    #学号符合，输出姓名信息
    if [ $ac = $StudentId ] && [ $au -eq 3 ]
    then
        flag=1
        echo -e "\033[40;93m学号：$ac 姓名：$nm\033[0m"
        break
    fi
done <${usr_file}
#账户不存在，报错
if [ $flag -eq 0 ]
then
    echo -e "\033[40;93m查询失败，用户不存在！\033[0m"
    continue
fi
#查询记录是否存在
itemcnt=$(cat $cfile | grep "^\$CourseId\$ \$StudentId\$" | wc -l)
#若信息存在说明已选课，反之未选课
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m该学生未选课！\033[0m"
else
    echo -e "\033[40;93m该学生已选课！\033[0m"
fi
done
}

#管理学生账户
ManS() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m\t"1') 创建学生账户"\033[0m"
        echo -e "\033[40;93m\t"2') 修改学生账户"\033[0m"
        echo -e "\033[40;93m\t"3') 删除学生账户"\033[0m"
        echo -e "\033[40;93m\t"4') 导入学生账户到课程"\033[0m"
        echo -e "\033[40;93m\t"5') 按学号查询学生账户"\033[0m"
        echo -e "\033[40;93m\t"b') 返回上级"\033[0m"
        echo -e "\033[40;93m\t"q') 退出"\033[0m"
        echo -e "\n"
        echo -e "\033[40;93m请选择：" "\033[0m"
        read choice
        case $choice in
            1) CreateStudent $CourseId;;
            2) ModifyStudent;;
            3) DeleteStudent $CourseId;;
            4) ImportStudent $CourseId;;
            5) QueryStudentById $CourseId;;
            b) return 0;;
            q) exit 0;;
            *) echo -e "\033[40;93m输入非法，请重新输入！"\033[0m;;
        esac
    done
}

```

```

        done
    }

#新建课程信息
CreateCourseInfo() {
    CourseId=$1
    cfile="./courseinfo"
    #新建课程信息文件夹
    if [ ! -d $cfile ]
    then
        mkdir $cfile
    fi
    itemcnt=1
    #找到一个合适的编号
    while true
    do
        cfile="./courseinfo/course_${CourseId}_${itemcnt}"
        if [ ! -f $cfile ]
        then
            echo -e "\033[40;93m请键入信息，输入一行一个字符'!'结束！"\033[0m"
            #读入信息内容
            while true
            do
                read tmp
                if [[ $tmp = '!' ]]
                then
                    break
                fi
                #将新内容写入文件
                echo $tmp >> $cfile
            done
            echo -e "\033[40;93m新建成功，当前信息编号为$itemcnt！"\033[0m"
            break
        fi
        ((itemcnt++))
    done
}

#编辑课程信息
EditCourseInfo() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程信息编号（输入'!'取消）\033[0m"
        #读入信息编号
        myRead
        CourseInfoId=$Buffer
        #信息编号为'!'则返回
        if [ $CourseInfoId = '!' ]
        then
            return 0
        fi
        cfile="./courseinfo/course_${CourseId}_${CourseInfoId}"
        #若信息文件存在则可以编辑，否则报错
    done
}

```

```

        if [ -f $cfile ]
        then
            vi $cfile
            echo -e "\033[40;93m"编辑成功! "\033[0m"
        else
            echo -e "\033[40;93m"编辑失败, 信息不存在! "\033[0m"
        fi
    done
}

#删除课程信息
DeleteCourseInfo() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入课程信息编号 (输入'!'取消) \033[0m"
        #读入信息编号
        myRead
        CourseInfoId=$Buffer
        #信息编号为'!'则返回
        if [ $CourseInfoId = '!' ]
        then
            return 0
        fi
        cfile="../courseinfo/course_${CourseId}_${CourseInfoId}"
        #判断信息文件是否存在,若是则删除,否则报错
        if [ -f $cfile ]
        then
            rm $cfile
            echo -e "\033[40;93m"删除成功! "\033[0m"
        else
            echo -e "\033[40;93m"删除失败, 信息不存在! "\033[0m"
        fi
    done
}

#显示课程信息
ListCourseInfo() {
    CourseId=$1
    cfile="../courseinfo"
    #文件夹不存在,直接返回
    if [ ! -d $cfile ]
    then
        echo -e "\033[40;93m"查询失败, 无课程信息! "\033[0m"
        return 0
    fi
    #统计有多少条对应课程号的信息
    itemcnt=0
    #扫描文件夹下所有对应课程号的信息文件
    for cinfo in $(ls $cfile)
    do
        if [[ $cinfo =~ ^course\_${CourseId}\_ ]]
        then
            ((itemcnt++))
        fi
    done
}

```

```

        #输出文件内容
        echo -e "\033[40;93m"${cinfo#course\_${CourseId}\_}\: ; cat
"./courseinfo/$cinfo"; echo -e "\033[0m"
    fi
done
#若无课程信息，输出提示
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m"查询失败，无课程信息！"\033[0m"
fi
}

#管理课程信息
ManCInfo() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m\t'1') 新建课程信息"\033[0m"
        echo -e "\033[40;93m\t'2') 编辑课程信息"\033[0m"
        echo -e "\033[40;93m\t'3') 删除课程信息"\033[0m"
        echo -e "\033[40;93m\t'4') 显示课程信息"\033[0m"
        echo -e "\033[40;93m\t'b') 返回上级"\033[0m"
        echo -e "\033[40;93m\t'q') 退出"\033[0m"
        echo -e "\n"
        echo -e "\033[40;93m请选择："\033[0m"
        read choice
        case $choice in
            1) CreateCourseInfo $CourseId;;
            2) EditCourseInfo $CourseId;;
            3) DeleteCourseInfo $CourseId;;
            4) ListCourseInfo $CourseId;;
            b) return 0;;
            q) exit 0;;
            *) echo -e "\033[40;93m输入非法，请重新输入！"\033[0m;;
        esac
    done
}

#新建作业实验
CreateHW() {
    CourseId=$1
    cfile=".hw"
    #新建作业文件夹
    if [ ! -d $ofile ]
    then
        mkdir $ofile
    fi
    touch "${ofile}/catalog"
    #输入起止日期
    echo -e "\n"
    echo -e "\033[40;93m请输入起始日期（格式为YYYY-MM-DD，输入'!'取消）"\033[0m"
myReadTime
StartTime=$Buffer
#日期'!'则返回
}

```

```

if [ $StartTime = '!' ]
then
    return 0
fi
#输入结束日期
echo -e "\033[40;93m请输入结束日期（格式为YYYY-MM-DD，输入'!'取消）\033[0m"
myReadTime
EndTime=$Buffer
#日期'!'则返回
if [ $EndTime = '!' ]
then
    return 0
fi
itemcnt=1
#找到一个合适的编号
while true
do
    cfile="../hw/course_${CourseId}_${itemcnt}"
    #文件不存在说明找到了
    if [ ! -f $cfile ]
    then
        echo -e "\033[40;93m请输入信息，输入一行一个字符`\\!`结束！\033[0m"
        #读入作业信息
        while true
        do
            read tmp
            if [[ $tmp = '!' ]]
            then
                break
            fi
            #将新信息写入文件
            echo $tmp >> $cfile
        done
        #将起止时间写入catalog
        echo $CourseId $itemcnt $StartTime $EndTime >> "./hw/catalog"
        echo -e "\033[40;93m新建成功，当前作业编号为$itemcnt！\033[0m"
        break
    fi
    ((itemcnt++))
done
}

#编辑作业实验
EditHW() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入作业实验编号（输入'!'取消）\033[0m"
        #读入作业编号
        myRead
        HWId=$Buffer
        #作业编号为'!'则返回
        if [ $HWId = '!' ]
        then

```

```

        return 0
    fi
    echo -e "\033[40;93m请输入作业起始时间（格式为YYYY-MM-DD，输入'!'取消，输入'/'不修改此
项）\033[0m"
    #特殊过滤，需要过滤'/'号用于特殊判断，因此需要带参数，下同
    myReadTime 0
    StartTime=$Buffer
    #起始时间为'!'则返回
    if [ $StartTime = '!' ]
    then
        return 0
    fi
    echo -e "\033[40;93m请输入作业结束时间（格式为YYYY-MM-DD，输入'!'取消，输入'/'不修改此
项）\033[0m"
    myReadTime 0
    EndTime=$Buffer
    #起始时间为'!'则返回
    if [ $EndTime = '!' ]
    then
        return 0
    fi
    cfile="../hw/course_${CourseId}_${HWId}"
    #若信息文件存在则可以编辑，否则报错
    if [ -f $cfile ]
    then
        vi $cfile
        while read cid aid st ed
        do
            #从catalog文件中查询到作业原来的起止时间
            if [[ $cid = $CourseId ]] && [[ $aid = $HWId ]]
            then
                if [[ $StartTime = '/' ]]
                then
                    StartTime=$st
                fi
                if [[ $EndTime = '/' ]]
                then
                    EndTime=$ed
                fi
                break
            fi
        done <"../hw/catalog"
        #从旧文件中删除相关信息，再将更新后的信息写到文件末尾
        cat "./hw/catalog" | grep -v "^$CourseId\ $HWId" >> "./hw/catalogTmp"
        mv "./hw/catalogTmp" "./hw/catalog"
        echo $CourseId $HWId $StartTime $EndTime >> "./hw/catalog"
        echo -e "\033[40;93m编辑成功！\033[0m"
    else
        echo -e "\033[40;93m编辑失败，信息不存在！\033[0m"
    fi
done
}

#删除作业实验
DeleteHW() {

```

```

CourseId=$1
while true
do
    echo -e "\n"
    echo -e "\033[40;93m请输入作业实验编号（输入'!'取消）\033[0m"
    #读入作业编号
    myRead
    HWId=$Buffer
    #作业编号为'!'则返回
    if [[ $HWId = '!' ]]
    then
        return 0
    fi

    cfile="../hw/course_${CourseId}_${HWId}"
    #若信息文件存在则删除，否则报错
    if [ -f $cfile ]
    then
        #删除作业文件
        rm $cfile
        #从catalog中删除相关信息
        cat "../hw/catalog" | grep -v "${CourseId}\| ${HWId}" >> "../hw/catalogTmp"
        mv "../hw/catalogTmp" "../hw/catalog"
        #还要从submit里删除所有提交信息
        #如果submit文件夹不存在的话要先创建
        if [ ! -d "./submit" ]
        then
            mkdir "./submit"
        fi
        #再从submit文件夹中搜索对应的文件
        find "./submit" -type f -name "${CourseId}_${HWId}" | xargs rm 2>/dev/null
        echo -e "\033[40;93m删除成功！\033[0m"
    else
        echo -e "\033[40;93m删除失败，作业不存在！\033[0m"
    fi
done
}

#显示作业实验
ListHW() {
    CourseId=$1
    cfile="../hw/catalog"
    while read cid aid st ed
    do
        #课程号不符合则直接跳过
        if [ $cid != $CourseId ]
        then
            continue
        fi
        echo -e "\033[40;93m作业编号: $aid 起始日期: $st 结束日期: $ed\033[0m"
        echo -e "\033[40;93m; cat "../hw/course_${cid}_${aid}; echo -e "\033[0m"
        echo
    done <$cfile
}

```

```

#查询学生作业完成情况
QueryStudentHW() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入作业实验编号（输入'!'取消）\033[0m"
        #读入作业编号
        myRead
        HWId=$Buffer
        #作业编号为'!'则返回
        if [ $HWId = '!' ]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入学生学号（输入'!'取消）\033[0m"
        #读入学号
        myRead
        Sid=$Buffer
        #学号为'!'则返回
        if [ $Sid = '!' ]
        then
            return 0
        fi

        cfile=".~/submit/hw_${CourseId}_${HWId}_${Sid}"
        #若信息文件存在则显示
        if [ -f $cfile ]
        then
            echo -e "\033[40;93m"; cat $cfile ; echo -e "\033[0m"
        else
            echo -e "\033[40;93m"查找失败，作业不存在或学生未递交！"\033[0m"
        fi
    done
}

#打印学生作业完成情况
ListStudentHW() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入作业实验编号（输入'!'取消）\033[0m"
        #读入作业编号
        myRead
        HWId=$Buffer
        #作业编号为'!'则返回
        if [ $HWId = '!' ]
        then
            return 0
        fi
        #已完成作业人数
        itemcnt=0
        #选课人数
        itemcnt2=0

```

```

while read cid sid
do
    if [ $cid = $CourseId ]
    then
        ((itemcnttwo++))
        tmp=`find ./submit -name "hw_${CourseId}_${HWId}_${sid}" | wc -l`
        if [ $tmp -ne 0 ]
        then
            ((itemcnt++))
            echo -e "\033[40;93m学号: $sid 状态: 已提交\033[0m"
        else
            echo -e "\033[40;93m学号: $sid 状态: 未提交\033[0m"
        fi
    fi
done <"./selcourse"
echo -e "\033[40;93m共有$itemcnttwo名学生选课, 其中完成作业$itemcnt人! \033[0m"
done
}

#管理作业实验
ManA() {
    CourseId=$1
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m\t'1') 新建作业实验"\033[0m"
        echo -e "\033[40;93m\t'2') 编辑作业实验"\033[0m"
        echo -e "\033[40;93m\t'3') 删除作业实验"\033[0m"
        echo -e "\033[40;93m\t'4') 显示作业实验"\033[0m"
        echo -e "\033[40;93m\t'5') 查看学生作业"\033[0m"
        echo -e "\033[40;93m\t'6') 打印学生完成情况"\033[0m"
        echo -e "\033[40;93m\t'b') 返回上级"\033[0m"
        echo -e "\033[40;93m\t'q') 退出"\033[0m"
        echo -e "\n"
        echo -e "\033[40;93m请选择: "\033[0m"
        read choice
        case $choice in
            1) CreateHW $CourseId;;
            2) EditHW $CourseId;;
            3) DeleteHW $CourseId;;
            4) ListHW $CourseId;;
            5) QueryStudentHW $CourseId;;
            6) ListStudentHW $CourseId;;
            b) return 0;;
            q) exit 0;;
            *) echo -e "\033[40;93m输入非法, 请重新输入! "\033[0m;;
        esac
    done
}

#管理所开课程
ManC() {
    CourseId=$1
    while true
    do

```

```

echo -e "\n"
echo -e "\033[40;93m\t'1')' 管理学生账户"\033[0m"
echo -e "\033[40;93m\t'2')' 管理课程信息"\033[0m"
echo -e "\033[40;93m\t'3')' 布置作业或实验"\033[0m"
echo -e "\033[40;93m\t'b')' 返回上级"\033[0m"
echo -e "\033[40;93m\t'q')' 退出"\033[0m"
echo -e "\n"
echo -e "\033[40;93m请选择: "\033[0m"
read choice
case $choice in
    1) ManS $CourseId;;
    2) ManCInfo $CourseId;;
    3) ManA $CourseId;;
    b) return 0;;
    q) exit 0;;
    *) echo -e "\033[40;93m输入非法, 请重新输入! "\033[0m;;
esac
done
}

```

```

#按课程号查询课程
QueryCourseById() {
    CourseId=$1
    cfile="./course"
    touch $cfile
    Buffer=""
    #扫描课程信息的文件
    while read cid nm
    do
        #课程存在, 输出记录
        if [[ $cid =~ $CourseId ]]
        then
            Buffer=$nm
            return 0
        fi
    done <${cfile}
}

```

```

#选择课程
ChooseCourse() {
    cfile="./givecourse"
    #开课数量
    itemcnt=0
    #开课课程号与名称列表
    courselist=()
    namelist=()
    while read cid tid
    do
        if [ $account = $tid ]
        then
            #获取课程名称
            QueryCourseById $cid
            CourseName=$Buffer
            #将课程号和课程名称存入数组
            courselist[$itemcnt]=$cid
            namelist[$itemcnt]=$CourseName
            ((itemcnt++))
        fi
    done <${cfile}
}

```

```

        namelist[$itemcnt]=$Buffer
        ((itemcnt++))
        echo -e "\033[40;93m$itemcnt') $cid $CourseName"\033[0m"
    fi
done <$cfile
if [ $itemcnt -eq 0 ]
then
    echo -e "\033[40;93m您暂无开课, 请联系管理员添加! "\033[0m"
    return 0
else
    echo -e "\033[40;93m您当前共有$itemcnt门课程, 请选择: (输入'!'返回) "\033[0m"
    while true
    do
        read choice
        #选择为'!'则返回
        if [ $choice = '!' ]
        then
            return 0
        fi
        #选择合法, 进行课程操作
        if [ "$choice" -gt 0 ] && [ "$choice" -le $itemcnt ] 2>/dev/null ;then
            ((choice--))
            echo -e "\033[40;93m当前操作课程: ${courselist[$choice]}"
${namelist[$choice]}\033[0m"
            ManC ${courselist[$choice]}
        else
            echo -e "\033[40;93m输入非法, 请重新输入! "\033[0m"
        fi
    done
fi
}

#修改密码
ChangePwd() {
    cfile=".:/usr"
    touch cfile
    if [ $account = "super" ]
    then
        echo -e "\033[40;93m当前账户不可修改密码! \033[0m"
        return 0
    fi
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入旧密码 (输入'!'取消) \033[0m"
        read -s oldpwd
        echo
        if [[ $oldpwd = '!' ]]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入新密码 (输入'!'取消) \033[0m"
        read -s newpwd
        echo
        if [[ $newpwd = '!' ]]

```

```

        then
            return 0
        fi
        #扫描账号信息的文件，对比账号密码
        while read ac pw au nm
        do
            if [[ $ac = $account ]]
            then
                if [[ $pw = $oldpwd ]]
                then
                    pw=$newpwd
                    #从旧文件中删除原信息
                    cat ${cfile} | grep -v "^$account" > "${cfile}Tmp"
                    mv "${cfile}Tmp" "${cfile}"
                    #将新信息附到文件最后
                    echo $ac $pw $au $nm >> "$cfile"
                    echo -e "\033[40;93m修改成功! \033[0m"
                    return 0
                else
                    echo -e "\033[40;93m修改失败，旧密码错误! \033[0m"
                    break
                fi
            fi
        done <${cfile}
    done
}

clear
echo -e "\n\n \t\t \033[40;93m =====欢迎您， $name ! =====\033[0m"
echo -e "\n\n \t \033[40;93m=====当前登录账号： $account 当前权限： 教师===== \033[0m"

while true
do
    echo -e "\n"
    echo -e "\033[40;93m\t'1' )' 管理所开课程"\033[0m"
    echo -e "\033[40;93m\t'2' )' 修改用户密码"\033[0m"
    echo -e "\033[40;93m\t'q' )' 退出"\033[0m"
    echo -e "\n"
    echo -e "\033[40;93m请选择：" "\033[0m"
    read choice
    case $choice in
        1) ChooseCourse;;
        2) ChangePwd;;
        q) exit 0;;
        *) echo -e "\033[40;93m输入非法，请重新输入！ "\033[0m;;
    esac
done

```

## student.sh

```

#!/bin/bash
#####
#程序名： student.sh （对应实验2第4题）

```

```

#作者: 黄彦玮
#学号: 3180102067
#说明: 图书管理系统-学生部分
#完成时间: 2020-07-31
#####
#####



#参数数量不等于2, 异常错误
if [ $# -ne 2 ]
then
    exit 1
fi

account=$1
name=$2
typeset -i itemcnt=0

#自定义读入, 忽略回车, 遇到输入的当行有字符即返回
myRead() {
    while true
    do
        read Buffer
        if [ -n $Buffer ]
        then
            break
        fi
        if [[ $Buffer =~ \ + ]]
        then
            echo -e "\033[40;93m输入含非法字符, 请重新输入! \033[0m"
        fi
    done
}

#日期格式检查
DateCheck() {
    #利用date命令判断日期是否合法
    if echo $1 | grep -Eq "[0-9]{4}-[0-9]{2}-[0-9]{2}" && date -d $1 +%Y%m%d > /dev/null 2>&1
    then
        return 1
    else
        return 0
    fi
}

#读入时间
myReadTime() {
    #参数表示是否需要过滤'/' (用于编辑作业)
    tmp=$#
    while true
    do
        read Buffer
        #调用函数判断读入日期是否合法
        DataCheck $Buffer
        #无特殊过滤要求
        if [ $tmp -eq 0 ]
        then

```

```

        if [ $? -ne 1 ] && [ $Buffer != '!' ]
        then
            echo -e "\033[40;93m格式错误, 请重新输入! \033[0m"
        else
            break
        fi
    else
        if [ $? -ne 1 ] && [ $Buffer != '!' ] && [ $Buffer != '/' ]
        then
            echo -e "\033[40;93m格式错误, 请重新输入! \033[0m"
        else
            break
        fi
    fi
done
}

#查询作业是否存在&是否在有效期
checkHW() {
    CourseId=$1
    HWId=$2
    while read cid aid st ed
    do
        #从catalog文件中查询作业信息
        if [ $cid = $CourseId ] && [ $aid = $HWId ]
        then
            #获得当前时间, 并转化成时间戳
            currenttime=$(date +%s)
            #将开始时间和结束时间都转化成时间戳
            sttime=$(date -d "$st" +%s)
            edtime=$(date -d "$ed" +%s)
            if [ $currenttime -ge $sttime ] && [ $currenttime -le $edtime ]
            then
                return 1
            fi
        fi
    done <"./hw/catalog"
    return 0
}

#新建作业
CreateWork() {
    CourseId=$1
    while true
    do
        echo -e "\033[40;93m请输入作业编号(输入'!'返回)! \033[0m"
        myRead
        HWId=$Buffer
        if [[ $HWId = '!' ]]
        then
            return 0
        fi
        #判断作业是否存在且可提交
        checkHW $CourseId $HWId
    done
}

```

```

#作业合法
if [ $? -eq 1 ]
then
    path=". ./submit/hw_${CourseId}_${HWId}_${account}"
    if [ ! -f $path ]
    then
        touch $path
        echo -e "\033[40;93m请键入信息，输入一行一个字符'!\'结束！"\033[0m"
        #读入信息内容
        while true
        do
            read tmp
            if [[ $tmp = '!' ]]
            then
                break
            fi
            #将信息写入对应文件
            echo $tmp >> $path
        done
        echo -e "\033[40;93m新建成功，作业已保存至$path！"\033[0m"
    else
        echo -e "\033[40;93m新建失败，作业已存在！"\033[0m"
    fi
else
    #作业不合法，重新输入
    echo -e "\033[40;93m新建失败，作业不存在或不在提交时间！"\033[0m"
fi
done
}

#编辑作业
EditWork() {
    CourseId=$1
    while true
    do
        echo -e "\033[40;93m请输入作业编号（输入'!'返回）！"\033[0m"
        myRead
        HWId=$Buffer
        if [[ $HWId = '!' ]]
        then
            return 0
        fi
        #判断作业是否存在且可提交
        checkHW $CourseId $HWId
        #作业合法
        if [ $? -eq 1 ]
        then
            path=". ./submit/hw_${CourseId}_${HWId}_${account}"
            if [ -f $path ]
            then
                vi $path
                echo -e "\033[40;93m编辑成功！"\033[0m"
            else
                echo -e "\033[40;93m编辑失败，作业不存在！"\033[0m"
            fi
        else
            echo -e "\033[40;93m作业不存在或不可编辑！"\033[0m"
        fi
    done
}

```

```

        else
            #作业不合法，重新输入
            echo -e "\033[40;93m新建失败，作业不存在或不在提交时间！\033[0m"
        fi
    done
}

#删除作业
DeleteWork()
{
    CourseId=$1
    while true
    do
        echo -e "\033[40;93m请输入作业编号（输入'!'返回）！\033[0m"
        myRead
        HWId=$Buffer
        if [[ $HWId = '!' ]]
        then
            return 0
        fi
        #判断作业是否存在且可提交
        checkHW $CourseId $Buffer
        #作业合法
        if [ $? -eq 1 ]
        then
            path=./submit/hw_${CourseId}_${HWId}_${account}"
            #作业存在
            if [ -f $path ]
            then
                rm $path
                echo -e "\033[40;93m删除成功！\033[0m"
            else
                echo -e "\033[40;93m删除失败，作业不存在！\033[0m"
            fi
        else
            #作业不合法，重新输入
            echo -e "\033[40;93m新建失败，作业不存在或不在提交时间！\033[0m"
        fi
    done
}

#查询作业
QueryWork() {
    CourseId=$1
    cfile=./hw/catalog"
    while read cid aid st ed
    do
        #课程号不符合则直接跳过
        if [[ $cid != $CourseId ]]
        then
            continue
        fi
        path=./submit/hw_${CourseId}_${aid}_${account}"
        if [ -f $path ]
        then

```

```

        isfinished="已提交， 路径为$path"
    else
        isfinished="未提交"
    fi
    echo -e "\033[40;93m作业编号: $aid 起始日期: $st 结束日期: $ed"\033[0m"
    echo -e "\033[40;93m作业状态: $isfinished"\033[0m"
    echo -e "\033[40;93m"; cat ".$hw/course_${cid}_${aid}"; echo -e "\033[0m"
done <$cfile
}

#修改密码
ChangePwd() {
    cfile=". /usr"
    touch cfile
    if [ $account = "super" ]
    then
        echo -e "\033[40;93m当前账户不可修改密码! \033[0m"
        return 0
    fi
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m请输入旧密码 (输入'!'取消) \033[0m"
        read -s oldpwd
        echo
        if [[ $oldpwd = '!' ]]
        then
            return 0
        fi
        echo -e "\033[40;93m请输入新密码 (输入'!'取消) \033[0m"
        read -s newpwd
        echo
        if [[ $newpwd = '!' ]]
        then
            return 0
        fi
        #扫描账号信息的文件，对比账号密码
        while read ac pw au nm
        do
            if [[ $ac = $account ]]
            then
                if [[ $pw = $oldpwd ]]
                then
                    pw=$newpwd
                    #从旧文件中删除原信息
                    cat $cfile | grep -v "^$account" > "${cfile}Tmp"
                    mv "${cfile}Tmp" "$cfile"
                    #将新信息附到文件最后
                    echo $ac $pw $au $nm >> "$cfile"
                    echo -e "\033[40;93m修改成功! \033[0m"
                    return 0
                else
                    echo -e "\033[40;93m修改失败， 旧密码错误! \033[0m"
                    break
                fi
            fi
        done
    done
}

```

```

        fi
    done <${cfile}
done
}

#管理作业
ManWork() {
    CourseId=$1
    #如果submit文件夹不存在的话要先创建
    if [ ! -d "./submit" ]
    then
        mkdir "./submit"
    fi
    while true
    do
        echo -e "\n"
        echo -e "\033[40;93m\t"1\033[0m"新建作业"\033[0m"
        echo -e "\033[40;93m\t"2\033[0m"编辑作业"\033[0m"
        echo -e "\033[40;93m\t"3\033[0m"删除作业"\033[0m"
        echo -e "\033[40;93m\t"4\033[0m"查询作业"\033[0m"
        echo -e "\033[40;93m\t"b\033[0m"返回上级"\033[0m"
        echo -e "\033[40;93m\t"q\033[0m"退出"\033[0m"
        echo -e "\n"
        echo -e "\033[40;93m请选择: "\033[0m"
        read choice
        case $choice in
            1) CreateWork $CourseId;;
            2) EditWork $CourseId;;
            3) DeleteWork $CourseId;;
            4) QueryWork $CourseId;;
            b) return 0;;
            q) exit 0;;
            *) echo -e "\033[40;93m输入非法, 请重新输入! "\033[0m;;
        esac
    done
}

#按课程号查询课程
QueryCourseById() {
    CourseId=$1
    cfile="./course"
    touch $cfile
    Buffer=""
    #扫描课程信息的文件
    while read cid nm
    do
        #课程存在, 输出记录
        if [[ $cid =~ $CourseId ]]
        then
            Buffer=$nm
            return 0
        fi
    done <${cfile}
}

```

```

#选择课程
ChooseCourse() {
    cfile="./selcourse"
    #选课数量
    itemcnt=0
    #选课课程号与名称列表
    courselist=()
    namelist=()
    while read cid sid
    do
        if [ $account = $sid ]
        then
            #获取课程名称
            QueryCourseById $cid
            CourseName=$Buffer
            #将课程号和课程名称存入数组
            courselist[$itemcnt]=$cid
            namelist[$itemcnt]=$Buffer
            ((itemcnt++))
            echo -e "\033[40;93m$itemcnt')' $cid $CourseName"\033[0m"
        fi
    done <$cfile
    if [ $itemcnt -eq 0 ]
    then
        echo -e "\033[40;93m您暂无选课,请联系教师添加!" "\033[0m"
        return 0
    else
        echo -e "\033[40;93m您当前共有$itemcnt门课程,请选择: (输入'!'返回) "\033[0m"
        while true
        do
            read choice
            #选择为'!'则返回
            if [ $choice = '!' ]
            then
                return 0
            fi
            #选择合法,进行课程操作
            if [ "$choice" -gt 0 ] && [ "$choice" -le $itemcnt ] 2>/dev/null ;then
                ((choice--))
                echo -e "\033[40;93m当前操作课程: ${courselist[$choice]}"
                ${namelist[$choice]}\033[0m"
                ManWork ${courselist[$choice]}
            else
                echo -e "\033[40;93m输入非法,请重新输入!" "\033[0m"
            fi
        done
    fi
}

clear
echo -e "\n\n \t\t \033[40;93m =====欢迎您, $name! =====\033[0m"
echo -e "\n\n \t \033[40;93m=====当前登录账号: $account 当前权限: 学生===== \033[0m"

```

```

while true
do
    echo -e "\n"
    echo -e "\033[40;93m\t"1\)\)选择课程"\033[0m"
    echo -e "\033[40;93m\t"2\)\)修改用户密码"\033[0m"
    echo -e "\033[40;93m\t"q\)\)退出"\033[0m"
    echo -e "\n"
    echo -e "\033[40;93m"请选择: "\033[0m"
    read choice
    case $choice in
        1) ChooseCourse;;
        2) ChangePwd;;
        q) exit 0;;
        *) echo -e "\033[40;93m"输入非法, 请重新输入! "\033[0m";;
    esac
done

```

## Exp2-5

( 30分 ) 使用任何一种程序设计语言实现一个shell程序的基本功能。

shell 或者命令行解释器是操作系统中最基本的用户接口。写一个简单的shell 程序——**myshell**，它具有以下属性：

(一) 这个shell 程序必须支持以下内部命令：bg、cd、clr、dir、echo、exec、exit、environ、fg、help、jobs、pwd、quit、set、shift、test、time、umask、unset。部分命令解释如下：

1) cd ——把当前默认目录改变为。如果没有参数，则显示当前目录。如该目录不存在，会出现合适的错误信息。这个命令也可以改变PWD 环境变量。

2) pwd ——显示当前目录。

3) time ——显示当前时间

4) clr ——清屏。

5) dir ——列出目录的内容。

6) environ ——列出所有的环境变量。

7) echo ——在屏幕上显示并换行（多个空格和制表符可能被缩减为一个空格）。

8) help ——显示用户手册，并且使用more 命令过滤。

9) quit ——退出shell。

10) shell 的环境变量应该包含shell=/myshell，其中/myshell 是可执行程序shell 的完整路径（不是你的目录下的路径，而是它执行程序的路径）。

(二) 其他的命令行输入被解释为程序调用，shell 创建并执行这个程序，并作为自己的子进程。程序的执行的环境变量包含一下条目：

parent=/myshell。

(三) shell 必须能够从文件中提取命令行输入，例如shell 使用以下命令行被调用：

myshell batchfile

这个批处理文件应该包含一组命令集，当到达文件结尾时shell退出。很明显，如果shell被调用时没有使用参数，它会在屏幕上显示提示符请求用户输入。

(四) shell 必须支持I/O 重定向，stdin 和stdout，或者其中之一，例如命令行为：

```
programname arg1 arg2 < inputfile > outputfile
```

使用arg1 和arg2 执行程序programname，输入文件流被替换为inputfile，输出文件流被替换为outputfile。

stdout 重定向应该支持以下内部命令：dir、environ、echo、help。

使用输出重定向时，如果重定向字符是>，则创建输出文件，如果存在则覆盖之；如果重定向字符为>>，也会创建输出文件，如果存在则添加到文件尾。

(五) shell 必须支持后台程序执行。如果在命令行后添加&字符，在加载完程序后需要立刻返回命令行提示符。

(六) 必须支持管道(“|”)操作。

(七) 命令行提示符必须包含当前路径。

## 设计文档

### 1、设计思想

本实验需要实现一个简易的shell，并且不使用任何现有代码，需求如题目中所示。我们使用C语言进行编写，调用相关的系统调用进行实现。经过分析需求可知，需求（七）说明了用户界面，显然最先处理；在此之后，需求（三）主要是与读入相关，它决定了程序是从标准输入读入指令还是从文件中读入，应当最先处理；需求（一）（二）（四）（五）主要解释了各指令的作用，其中需求（四）决定了指令执行时的输入和输出来源，需求（五）决定了指令执行时父进程的状态，但这些都可以在指令执行过程中的细节处理，整个指令的执行这一过程可以看作一个整体；需求（六）中的管道操作实际就是将若干个这样的整体过程串联起来，因此也需要优先处理。同时，根据使用经验可知，需求中列出的指令并不是所有都可以通过新建进程来执行的（例如cd指令属于内嵌指令，新建进程执行是没有意义的），这也需要特殊处理。

综合上述分析，我们可以得出以下设计思想：

1. 先打印用户界面（包括命令行提示符）
2. 根据参数个数从标准输入/文件中读取指令
3. 判断是否有管道，若有管道，则从左至右顺序执行
4. 对于每个单个指令（不含管道）的执行，先处理重定向和前后台，再执行指令具体内容
5. 对于内嵌指令直接执行，其他指令新建子进程执行，同时根据前后台的不同控制父进程的执行

具体实现时，遵循模块化由上到下逐步细化的规则，设计相应的函数逐步实现。

### 2、设计亮点

- 高度还原bash的shell界面
- 较为完善的错误处理
- 友好、还原度高的信息提示
- 在需求基础上另外新增了少量指令
- 完善地实现了重定向、管道、前后台切换三大困难点
- 良好的进程管理

### 3、功能模块

shell内部可以分为四个模块：用户交互模块(main)、指令解释及控制模块(controller)、指令执行模块(mycmd)、辅助模块(proc\_list)。

用户交互模块实现以下功能：

- 打印命令提示符
- 若调用shell时含参数，从参数对应的文件中读入指令，否则从标准输入中读入并提示用户输入指令
- 处理执行指令返回的错误信息
- 回收僵尸进程

指令解释及控制模块实现以下功能：

- 按照空格分割指令，提取指令中的信息，包括是否含有管道、是否有重定向、前台/后台、指令类型等信息
- 对于管道指令，新建子进程并递归执行
- 实现指令的输入输出重定向
- 对于内嵌指令，直接调用指令执行模块的函数执行，否则新建进程执行
- 对于前后台指令采取不同的执行方式，即控制父进程是否阻塞

指令执行模块实现以下功能：

- 执行指令的具体内容，即根据指令类型、参数实现不同的具体功能

辅助模块实现以下功能：

- 管理当前进程的子进程表，便于进程控制，也便于bg、fg、jobs等指令的执行
- 管理信号处理，包括捕捉Ctrl+Z等信号，用于控制子进程的停止

### 4、实现方法

**用户交互模块**的实现相对简单，只需要根据参数个数（c语言中的argc）从标准输入或文件中读取指令即可，其中对于从标准输入中读入的情况，需要写一个无限循环用于读入指令，每次读入指令前输出命令提示符进行提示。指令执行完成后，err\_proc()函数会对指令返回的错误代码进行处理，并输出相应的错误信息。最后，recycle\_proc()函数将会调用系统调用函数回收所有的僵尸进程，并从进程表中删除相应的进程。

**指令解释及控制模块**较为复杂，以下具体说明其工作原理：

- 指令解释：首先以空格或换行符“\n”作为分隔符将源命令字符串分割成若干子字符串，之后利用**有限状态机**进行解释。例如根据第一个子串可以知道指令类型，之后再根据不同的指令类型提取不同的参数，并同时判断参数的个数、格式等是否合法。
- 指令控制：我们设计一个函数execom(l, r)表示执行下标在[l, r]区间内的子字符串组成的指令（可能含有管道），execom\_without\_pipe(l, r)表示执行下标在[l, r]区间内的子字符串组成的指令且不含管道。其具体执行过程如以下伪代码所示：

```
execom(l, r):
    if( 管道不存在 )
        execom_without_pipe(l, r);
    else if( 管道不合法 )
        return error_code; //返回错误代码
    else
        pip_pos = 第一个管道符"|"所在下标;
        新建一个管道;
```

```

新建一个进程;
if( 进程创建失败 )
    return error_code; //返回错误代码
else if( 子进程 )
    将标准输出重定向到管道输出端;
    execom(l, pip_pos); //执行最左侧的管道指令
    错误处理后退出;
else //父进程
    将标准输入重定向到管道输入端;
    等待子进程终止;
    恢复I/O重定向;
    if( 子进程正常退出 )
        execom(pip_pos + 1, r); //递归处理剩余命令
    else
        return error_code; //返回错误代码

execom_without_pipe(l, r):
    if( 内嵌指令 )
        处理重定向;
        //直接调用相关函数执行
    else
        新建一个进程;
        if( 进程创建失败 )
            return error_code; //返回错误代码
        else if( 子进程 )
            处理重定向;
            //调用相关函数执行
        else //父进程
            将子进程加入进程表;
            if( 后台进程 )
                打印子进程信息后退出;
            else
                注册信号函数; //用于捕捉Ctrl+Z信号使子进程停止
                等待子进程停止(Stopped)或终止(Terminated);
                恢复信号函数;
                if( 子进程正常终止 )
                    从进程表中删除进程;
                    return error_code; //返回错误代码
                else if( 子进程停止 )
                    更新进程表后退出;
                else //子进程异常退出
                    return error_code; //返回错误代码

```

注：本实验中认定的内嵌指令（即必须由主进程直接执行的指令）包括：cd、clr、quit、exit、bg、fg、set、unset、umask、exec。

### **指令执行模块调用相关系统调用实现，以下分指令说明实现方法：**

- cd：调用系统调用chdir()实现。其中主目录的路径可以用系统调用getenv()获取HOME环境变量得到。
- pwd：用系统调用getenv()获取PATH环境变量得到。
- clr：输出特殊符号达到清屏效果。
- time：调用time()、ctime()等函数实现。

- dir : 先调用chdir()切换到对应目录，然后调用readdir()实现，遍历其结果并根据不同文件类型进行不同的输出。
- environ : 使用Linux C自带的外部变量environ，输出该指针指向的内容实现。
- echo : 直接输出即可。
- exit/quit : 直接退出即可。
- jobs : 输出进程表中的项的信息。
- bg : 修改进程表中对应项的信息。若子进程已停止，则向该进程发送SIGCONT信号令其继续执行。
- fg : 先修改进程表中对应项信息，然后注册信号函数并阻塞父进程。子进程终止或停止后根据返回值进行不同的操作。若子进程已停止，则向该进程发送SIGCONT信号令其继续执行。
- set/unset : 分别调用系统调用函数setenv()/unsetenv()实现
- umask : 调用系统调用umask()实现。
- test/help/shift : 简单模拟即可
- exec : 相当于执行execom()。
- sleep : 调用sleep()实现。
- cat/more : 调用文件流I/O相关系统调用实现。
- 外部指令 : 调用系统调用execvp()实现。

**辅助模块**的实现中，进程表部分直接采用数组进行实现；为捕捉Ctrl+Z信号，需要调用系统调用函数signal()进行函数注册，将信号SIGTSTP（按Ctrl+Z时自动向进程发送的信号）的处理函数重定向到自己设计的函数。该函数的内容是：当该信号触发时，需要向子进程发送SIGSTOP信号，并修改进程表。

除了以上各模块的基本实现方法之外，对于管道、重定向、前后台控制这三大难点的实现还需要注意以下细节的实现：

- 对于文件重定向，实现的方式是：首先打开对应文件，得到文件描述符fd。然后调用dup(0)新建一个文件描述符指向标准输入。接着调用close(0)关闭文件描述符0，再调用dup(fd)时，由于dup()函数新建的文件描述符是当前可用的文件描述符中数值最小的一个，因此文件描述符fd将与文件描述符0指向相同，这就将输入重定向到了对应文件。输出也可类似操作。当指令执行完成后，需要将输入和输出恢复到标准输入和标准输出，这时的实现方法是：调用close(fd)关闭文件对应描述符，之后调用dup2(stdin\_copy, 0)将文件描述符0重新定向到标准输入，其中stdin\_copy是之前dup(0)所新建的指向标准输入的文件描述符。
- 对于管道中的父进程和子进程中的数据传输，同样需要用到重定向，其实现方法是：首先调用pipe(file\_pipes[2])新建管道，此时父进程和子进程分别拥有了两个各自的文件描述符，其中file\_pipes[0]用于读入，file\_pipes[1]用于输出。根据之前的实现原理可知，子进程的输出需要和父进程的输入相关联。为此，子进程可以先关闭file\_pipes[0]（因为子进程不需要从file\_pipes[0]读入），同时父进程也可以先关闭file\_pipes[1]。之后用类似的方法将子进程的file\_pipes[1]与标准输出相关联、父进程的file\_pipes[0]和标准输入相关联。这样一来，子进程向标准输出中输出的内容就可以被父进程从标准输入中读出来了。最后，当指令执行完成后，采用类似的方法进行恢复。
- 对于前后台控制，其本质区别在于父进程是否阻塞。对于前台进程，父进程调用waitpid(pid, &stat\_val, 0)函数等待子进程终止（其中pid是子进程的进程编号，stat\_val储存子进程的退出结果），调用该函数时，若子进程正在运行，则父进程是阻塞的，这就实现了使进程看起来是在前台运行。由于子进程也可能在Ctrl+Z的触发下进入停止(Stopped)状态，此时子进程并未终止(Terminated)，所以此时函数调用需要写成waitpid(pid, &stat\_val, WUNTRACED)，以允许子进程在停止的状态下waitpid()函数也可以正常返回。对于后台进程，则直接输出信息即可。在回收进程时，可以调用waitpid(pid, &stat\_val, WNOHANG)以不阻塞的方式查看子进程是否终止，若已终止则waitpid()函数会自动对进程进行回收，这就实现了僵尸进程回收，节省了系统资源。

## 5、讨论与改进

- 题目中需求可以进一步具体化，实现更多的指令、根据更多的参数实现更详细的功能；
- 错误信息的分类可以更加细致，错误信息也可以更加具体；
- 前后台控制也可以使用捕捉SIGCHLD信号实现（子进程在终止时会向父进程发送SIGCHLD信号），可能更加方便。

# 用户手册

## 1、简介

本程序myshell是一个简易的Linux Shell程序，采用C语言编写。它可以接受用户遵循一定语法输入的命令行，将其解析后实现不同的功能。目前，本程序已实现了传统的Linux Bash的部分功能。具体功能请参考“功能说明”一节。

本程序所有代码及功能均在Ubuntu 20.04 64位系统上实验运行通过。

作者：黄彦玮（浙江大学竺可桢学院混合班2018级学生）

版本：v1.0

完成日期：2020-08-18

## 2、版本特性

- 支持bg、cd、clr、dir、echo、exec、exit、environ、fg、help、jobs、pwd、quit、set、shift、test、time、umask、unset、sleep、cat、more等多个指令
- 支持外部程序调用
- 支持从文件中提取命令行输入
- 支持I/O重定向
- 支持进程前台/后台运行、进程前后台切换
- 支持管道指令

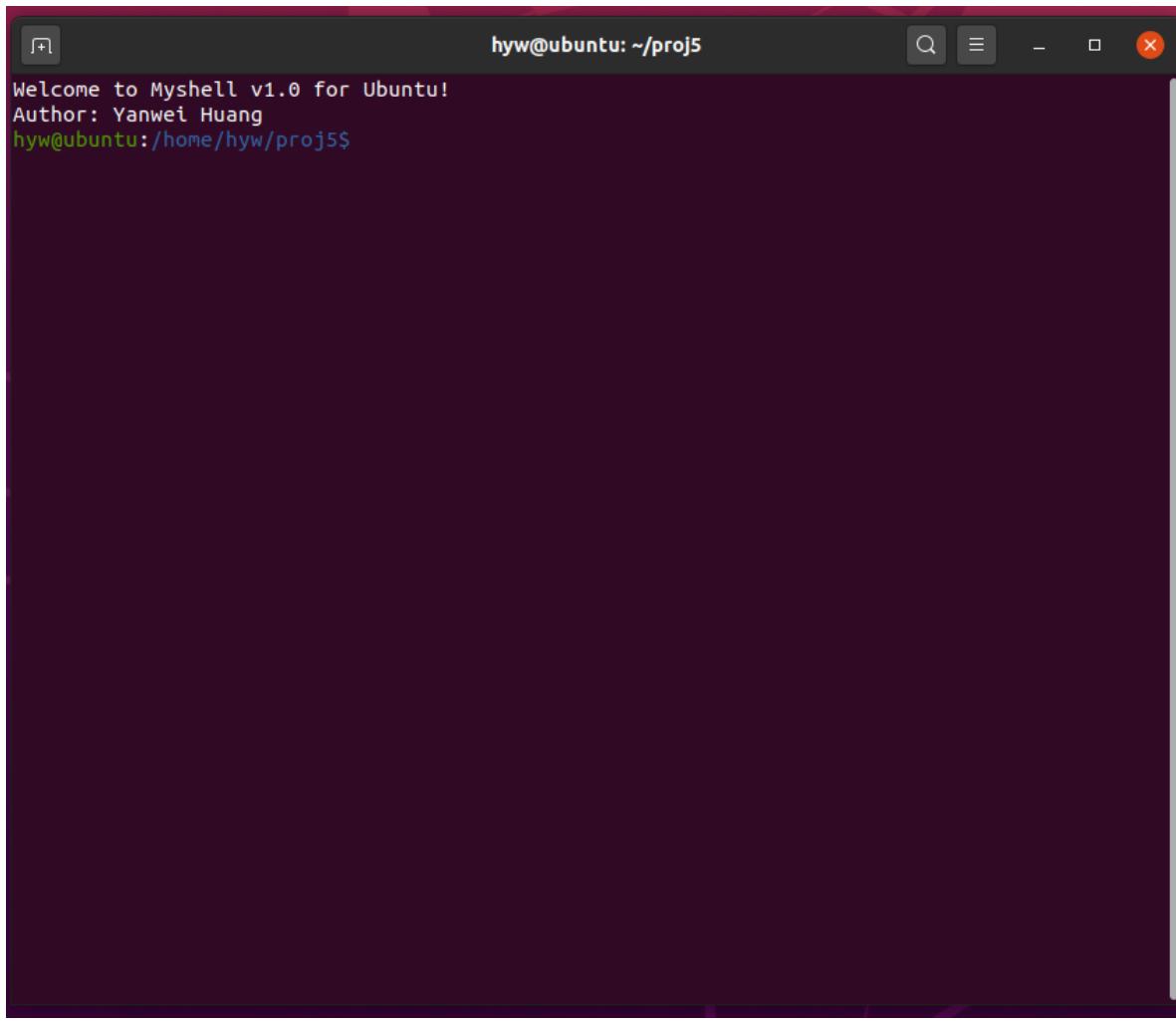
## 3、安装方式

在bash中在Makefile和其他源文件所在文件夹下键入命令行make即可一键编译，之后运行生成的myshell文件即可。

若想要清除生成的编译文件，请键入命令行make clean清除。

## 4、功能说明

运行程序后，进入主界面，如下图所示。最上方的两行为欢迎语。用户可以在shell中输入指令，指令的格式及含义请参考“指令说明”一章。在输入指令前都会有相应的命令提示符，包括了用户名、主机名、当前路径等信息。



hyw@ubuntu: ~/proj5

Welcome to Myshell v1.0 for Ubuntu!  
Author: Yanwei Huang  
hyw@ubuntu:/home/hyw/proj5\$

随后，用户可以输入想要输入的指令，按回车结束。以“cd ..”为例，该指令的含义是返回上级目录。执行后的效果如下图所示。从行首的命令提示符可以看出，当前路径已经成功返回了上一级。

类似地，用户也可以输入其他指令。工作完成后，用户输入"exit"或"quit"指令即可退出程序。

## 5、指令说明

(注-符号说明：\${val\_name}表示变量中的值，如\$str表示str这一变量的值；[ ]中的参数表示可选参数；{}中的参数表示该参数的值为大括号中值的其中之一。)

myshell支持的基本指令如下：

### 1. cd

作用：改变当前目录

用法：cd [path]

解释：当\$path参数存在时，切换到\$path对应的目录。否则切换到主目录。

### 2. pwd

作用：打印当前目录

用法：pwd

### 3. clr

作用：清除屏幕内容

用法：clr

### 4. time

作用：打印当前时间

用法：time

## 5. dir

作用：列出指定目录下的内容，包括文件、目录等

用法：dir \$path

解释：打印\$path目录下的内容。

## 6. environ

作用：列出所有的环境变量（注：环境变量的含义可参考附录）

用法：environ

## 7. echo

作用：在屏幕上打印其参数并换行（多个空格可能被缩减为一个空格）

用法：echo [\$para1] [\$para2] [...]

解释：依次打印\$para1、\$para2、...的内容，并在最后输出一个空行。多个空格可能被缩减为一个空格。若无参数则直接输出一个空行。

## 8. exit

作用：退出myshell程序

用法：exit

## 9. quit

作用：退出myshell程序

用法：quit

## 10. jobs

作用：打印当前进程下的所有作业（子进程）的信息（注：进程相关概念的含义可参考附录）

用法：jobs

## 11. bg

作用：将当前进程中指定作业（子进程）设定为后台运行

用法：bg \$id

解释：将作业号为\$id的作业（子进程）设定为后台运行，其中\$id可以通过jobs指令查询得到。

## 12. fg

作用：将当前进程中指定作业（子进程）设定为前台运行

用法：fg \$id

解释：将作业号为\$id的作业（子进程）设定为前台运行，其中\$id可以通过jobs指令查询得到。

## 13. set

作用：打印环境变量&修改环境变量

用法：set [\$env\_name \$env\_val]

解释：无参数时，打印所有的环境变量。有2个参数时，将\$env\_name对应的环境变量的值修改为\$env\_val，若环境变量不存在则会被新建。

## 14. unset

作用：删除环境变量

用法：unset \$env\_name

解释：删除\$env\_name对应的环境变量。

## 15. umask

作用：打印掩码&修改掩码（注：掩码的含义可参考附录）

用法：umask [\$val]

解释：无参数时直接打印掩码，有参数时将掩码修改为\$val对应的值。

#### 16. test

作用：用于字符串比较

用法：test \$str1 {"=", "!=","=="} \$str2

解释：无参数时直接打印掩码，有参数时将掩码修改为\$val对应的值。

#### 17. help

作用：用于输出指令帮助信息（本指令还支持more指令过滤，见第21条more指令）

用法：help \$cmd

解释：输出指令\$cmd的帮助信息。

#### 18. shift

作用：用于参数移位，多用于管道（注：管道的含义请参考本节后半部分）

用法：... | shift [\$shr]

解释：读入参数后向左移动\$shr个参数后输出，无参数时默认移动1个参数。

例：

```
$ echo a b c d | shift 2
```

```
c d
```

#### 19. exec

作用：将当前进程替换为exec后的指令

用法：exec \$cmd

解释：将当前进程替换为\$cmd中的指令，执行后退出。

#### 20. sleep

作用：使进程休眠

用法：sleep \$val

解释：使进程休眠\$val秒。

#### 21. cat

作用：打印文件内容

用法：cat \$path

解释：打印\$path对应文件的内容。

#### 22. more

作用：过滤输入文本

用法1：more [path]

用法2：... | more

解释：输出\$path中的文件内容，若无参数则从标准输入读入内容并输出（无参数时建议用于管道）。对于行数较多的文件，每次显示一屏幕的内容，之后将会提示"more?"等待用户输入。若用户输入"q"则结束指令，若用户输入空格则显示下一屏，若用户输入回车则显示下一行。

例：

```
$help cd | more
```

```
#文本内容
```

```
$q
```

```
#退出
```

myshell还支持如下调用：

1. 除上述命令外，其他命令将被解释为外部程序调用，myshell 创建并执行这个程序，并作为自己的子进程。

例如：

```
$gedit tmp.c
```

#打开gedit编辑界面

2. myshell 能够从文件中提取命令行输入，例如myshell使用以下命令行被调用：

```
$myshell batchfile
```

这个批处理文件应该包含一组命令集，myshell会依次执行文件中的指令，当到达文件结尾时shell退出。

3. myshell支持I/O重定向（注：重定向相关概念的含义可参考附录），"<"表示重定向标准输入，之后的参数表示用于输入的文件；">"和">>"表示重定向表示输出，之后的参数表示用于输出的文件。其中，">"和">>"后的文件若不存在则会自动创建，若存在，重定向符号为">"时会覆盖文件内容，重定向符号为">>"时会输出到文件结尾。例如：

```
$shift 2 < input > output
```

若input中的内容为a b c，则文件output中的内容为c。

4. myshell支持后台程序执行（注：进程相关概念的含义可参考附录），只需要在命令结尾加上参数"&"即可。

5. myshell支持管道操作，（注：管道相关概念的含义可参考附录），进程和进程之间用"|"隔开。

例如：

```
$echo a b c | shift
```

```
b c
```

## 6、信号

myshell目前支持捕捉两种信号：SIGTSTP和SIGINT。用户可以通过向shell中输入Ctrl+Z和Ctrl+C来向进程发送这两种信号。这两种信号的作用分别是：SIGTSTP可以用于使前台运行的子进程停止；SITINT可以使当前运行的进程终止。（注：信号、停止、终止、前后台进程、子进程等概念可参考附录）

## 7、附录：Linux Shell 相关概念解释

1. 命令：由shell执行的遵循特定语法、具有特定含义的语句称为命令，例如cd ..就是一条可以由shell执行用于返回上级目录的命令。
2. 参数：命令的第一个单词通常是命令类型，后面的各个单词即为参数。例如指令dir /usr 中的/usr就是参数。
3. 标准输入/标准输出：执行一个shell命令行时通常会自动打开三个标准文件，即标准输入文件（stdin），通常对应终端的键盘；标准输出文件（stdout）和标准错误输出文件（stderr），这两个文件都对应终端的屏幕。进程将从标准输入文件中得到输入数据，将正常输出数据输出到标准输出文件，而将错误信息送到标准错误文件中。
4. I/O重定向：通常情况下shell从标准输入中读取数据，向标准输出输出数据；但在一些情况下可能会从其他来源（如某个文件）中读取数据，又或向某个来源输出数据，这时就可以将标准输入（输出）与目标输入（输出）源关联起来，这一操作就是I/O重定向。例如，当某个命令的输出被重定向到一个文件时，该命令中所有原本输出到标准输出的文本将不显示在屏幕上，而将直接输出到文件中。

5. 环境变量：环境变量一般是指在操作系统中用来指定操作系统运行环境的一些参数，它包含了一个或者多个应用程序所将使用到的信息。例如PATH环境变量，当要求系统运行一个程序而没有告诉它程序所在的完整路径时，系统除了在当前目录下面寻找此程序外，还应到PATH中指定的路径去找。用户通过设置环境变量，来更好的运行进程。
6. 进程：进程是一个具有一定独立功能的程序关于某个数据集合的一次运行活动，它是操作系统动态执行的基本单元。狭义地说，进程可以被认为是正在运行的程序的实例。
7. 父进程和子进程：父进程指已创建一个或多个子进程的进程。在UNIX里，除了进程0（即PID=0的交换进程）以外的所有进程都是由其他进程使用系统调用fork创建的，这里调用fork创建新进程的进程即为父进程，而相对应的为其创建出的进程则为子进程，因而除了进程0以外的进程都只有一个父进程，但一个进程可以有多个子进程。
8. 前台进程&后台进程：前台进程是当前正在使用的程序，后台进程是在当前没有使用的但是也在运行的进程。两者的显示区别在于父进程是否阻塞。前台进程和后台进程可以互相转换。Linux中将前台进程挂到后台的意思就是，退出程序但是让这个程序依然进行运行，而不是退出当前程序只能关闭。默认情况下，我们启动的每一个进程都是前台进程，它从键盘获得输入并发送它的输出到屏幕。当一个进程运行在前台时，我们不能在同一命令行提示符下运行任何其他命令（启动任何其他进程），只有将进程放在后台运行的好处时，我们才可以继续运行其他命令。
9. 掩码：掩码（umask）是用来指定目前用户在新建文件或者目录时候的权限默认值，为一个4位8进制数。一般而言，每个文件或目录的权限包括当前用户、组用户和其他用户的读、写、执行权限（共9个），掩码二进制中的后9位就决定了新建文件或目录时的这9个默认权限。对于文件，用666减去掩码得到一个3位8进制数，写成二进制后，高三位表示当前用户的读、写、执行权限，中间三位表示组用户的读、写、执行权限，低三位表示其他用户的读、写、执行权限。对于目录，用777减去掩码得到一个3位8进制数，写成二进制后，高三位表示当前用户的读、写、执行权限，中间三位表示组用户的读、写、执行权限，低三位表示其他用户的读、写、执行权限。
10. 管道：管道是一种通信机制，通常用于进程间的通信，它表现出来的形式将前面每一个进程的输出直接作为下一个进程的输入。管道命令使用“|”作为界定符号。
11. 信号：信号是进程间通讯的一种有限制的方式。它是一种异步的通知机制，用来提醒进程一个事件已经发生。当一个信号发送给一个进程，该进程正常的控制流程就将被中断。如果进程定义了信号的处理函数，那么它将被执行，否则就执行默认的处理函数。

## 实验结果

1. 如图所示，输入make进行编译，之后输入./myshell运行程序。

```
hyw@ubuntu:~/pr5$ cd pr5
hyw@ubuntu:~/pr5$ make
gcc -c main.c -o main.o
main.c: In function ‘main’:
main.c:58:13: warning: implicit declaration of function ‘err_proc’; did you mean
‘errproc’? [-Wimplicit-function-declaration]
  58 |         err_proc(res);
     |         ^
     |         errproc
main.c: At top level:
main.c:141:6: warning: conflicting types for ‘err_proc’
  141 | void err_proc(int err)
     |         ^
     |
main.c:58:13: note: previous implicit declaration of ‘err_proc’ was here
  58 |         err_proc(res);
     |         ^
     |         errproc
gcc -c proc_list.c -o proc_list.o
gcc -c controller.c -o controller.o
controller.c: In function ‘execom’:
controller.c:78:13: warning: implicit declaration of function ‘err_proc’; did yo
u mean ‘errproc’? [-Wimplicit-function-declaration]
  78 |         err_proc(res);
     |         ^
     |         errproc
controller.c: In function ‘execom_without_pipe’:
controller.c:172:16: warning: implicit declaration of function ‘execlr’; did you
mean ‘execlp’? [-Wimplicit-function-declaration]
  172 |         return execlr(l, newr);
     |         ^
     |         execlp
gcc -c mycmd.c -o mycmd.o
mycmd.c: In function ‘exeexec’:
mycmd.c:289:5: warning: implicit declaration of function ‘err_proc’; did you mea
n ‘errproc’? [-Wimplicit-function-declaration]
  289 |     err_proc(res);
     |     ^
     |     errproc
gcc main.o proc_list.o controller.o mycmd.o -o myshell
hyw@ubuntu:~/pr5$ ./myshell
```

2. 运行程序后，进入欢迎界面。

```
Welcome to Myshell v1.0 for Ubuntu!
Author: Yanwei Huang
hyw@ubuntu:/home/hyw/pr5$
```

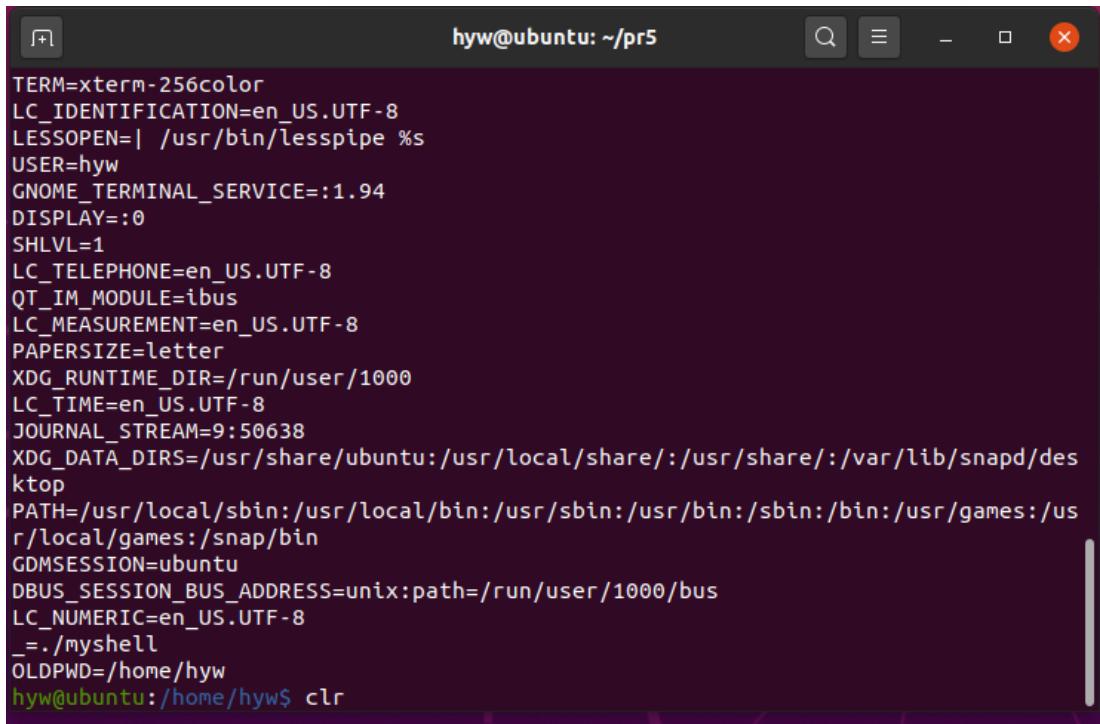
3. 测试第一组指令，如下图所示。

```
hyw@ubuntu: ~/pr5
Welcome to Myshell v1.0 for Ubuntu!
Author: Yanwei Huang
hyw@ubuntu:/home/hyw/pr5$ dir .
main.o
mycmd.o
main.c
controller.c
Makefile
myshell
tmp
mycmd.c
proc_list.c
controller.h
main.h
controller.o
mycmd.h
proc_list.o
proc_list.h
hyw@ubuntu:/home/hyw/pr5$ cd ..
hyw@ubuntu:/home/hyw$ pwd
/home/hyw
hyw@ubuntu:/home/hyw$ time
Wed Aug 19 14:16:40 2020
hyw@ubuntu:/home/hyw$
```

#### 4. 测试第二组指令。

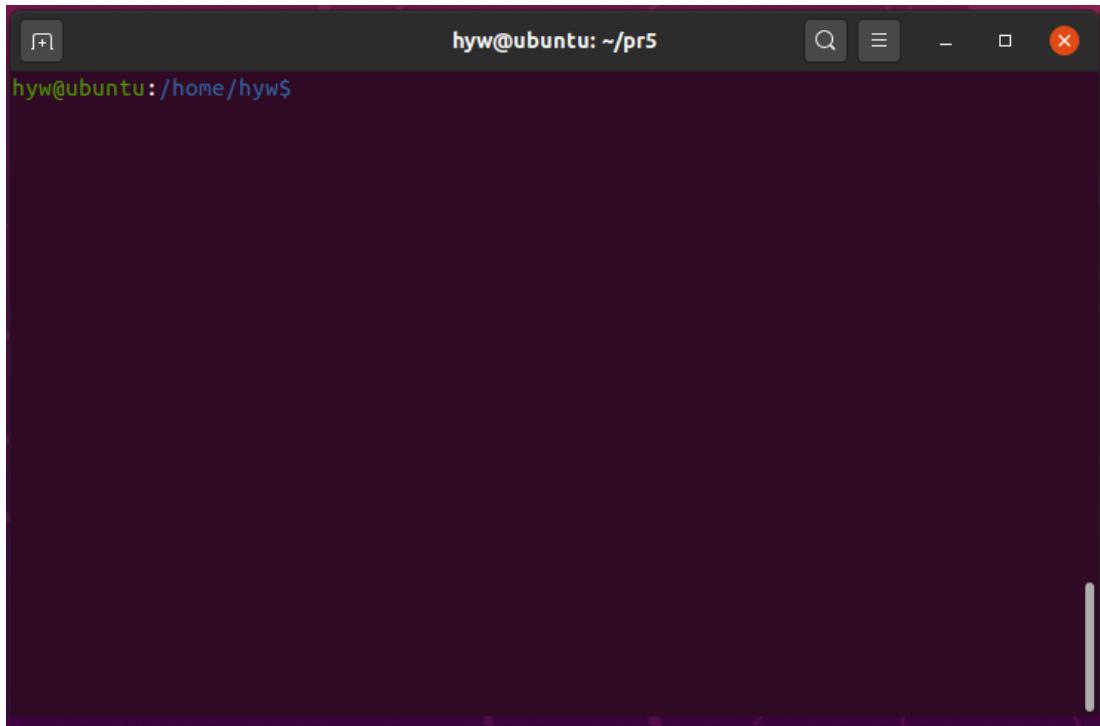
```
hyw@ubuntu:~/pr5
hyw@ubuntu:~/pr5
hyw@ubuntu:/home/hyw$ environ
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1897,unix/ubuntu:/tmp/.ICE-unix/189
7
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
LANGUAGE=en_US:en
QT4_IM_MODULE=ibus
LC_ADDRESS=en_US.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1790
GTK_MODULES=gail:atk-bridge
PWD=/home/hyw/pr5
LOGNAME=hyw
XDG_SESSION_DESKTOP=ubuntu
```

(environ列出所有环境变量，条目很多只列出部分)



```
hyw@ubuntu: ~/pr5
TERM=xterm-256color
LC_IDENTIFICATION=en_US.UTF-8
LESSOPEN=| /usr/bin/lesspipe %s
USER=hyw
GNOME_TERMINAL_SERVICE=:1.94
DISPLAY=:0
SHLVL=1
LC_TELEPHONE=en_US.UTF-8
QT_IM_MODULE=ibus
LC_MEASUREMENT=en_US.UTF-8
PAPERSIZE=letter
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=en_US.UTF-8
JOURNAL_STREAM=9:50638
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
LC_NUMERIC=en_US.UTF-8
_=./myshell
OLDPWD=/home/hyw
hyw@ubuntu:/home/hyw$ clr
```

(clr清屏，如下图)



```
hyw@ubuntu: ~/pr5
hyw@ubuntu:/home/hyw$
```

5. 测试第三组指令。

```
hyw@ubuntu:/home/hyw/pr5$ sleep 5
^Z[1]5169 Stopped sleep
hyw@ubuntu:/home/hyw/pr5$ jobs
[1]5169 Stopped sleep
hyw@ubuntu:/home/hyw/pr5$ bg 1
[5169] sleep
hyw@ubuntu:/home/hyw/pr5$
[Finished]5169
hyw@ubuntu:/home/hyw/pr5$ sleep 5 &
[1]5171 Running sleep
hyw@ubuntu:/home/hyw/pr5$ jobs
[1]5171 Running sleep
hyw@ubuntu:/home/hyw/pr5$ fg 1
[1]5171 Running sleep
hyw@ubuntu:/home/hyw/pr5$
```

解释：sleep 5 新建一个前台进程，按Ctrl+Z令其停止，此时输出被停止的进程的信息。输入jobs列出当前所有子进程。bg 1表示让当前作业号为1的进程后台运行，即将之前的sleep 5转入后台。一段时间后，子进程终止，此时输入回车，shell自动回收僵尸进程，输出回收的进程信息。之后sleep 5 &创建一个后台进程，再输入jobs列出当前所有子进程，最后fg 1将进程sleep 5转入前台，一段时间后进程终止，输出命令提示符。

#### 6. 测试第四组指令。

```
hyw@ubuntu:/home/hyw/pr5$ umask
0002
hyw@ubuntu:/home/hyw/pr5$ umask 0003
hyw@ubuntu:/home/hyw/pr5$ umask
0003
hyw@ubuntu:/home/hyw/pr5$ umask 0002
hyw@ubuntu:/home/hyw/pr5$ umask
0002
hyw@ubuntu:/home/hyw/pr5$ test aaa = aaa
True
hyw@ubuntu:/home/hyw/pr5$ test aaa == bbb
False
hyw@ubuntu:/home/hyw/pr5$ test aaa != aaa
False
hyw@ubuntu:/home/hyw/pr5$ test aaa != bbb
True
hyw@ubuntu:/home/hyw/pr5$ help dir
dir -- List the files under a given directory
Usage: dir $path
hyw@ubuntu:/home/hyw/pr5$ exit
hyw@ubuntu:~/pr5$
```

(exit退出shell回到bash)

#### 7. 测试第五组指令。

```
hyw@ubuntu:~/home/hyw/pr5$ set
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1897,unix/ubuntu:/tmp/.ICE-unix/189
7
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
LANGUAGE=en_US:en
QT4_IM_MODULE=ibus
LC_ADDRESS=en_US.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1790
GTK_MODULES=gail:atk-bridge
PWD=/home/hyw/pr5
LOGNAME=hyw
XDG_SESSION_DESKTOP=ubuntu
```

(输入set打印所有环境变量)

```
hyw@ubuntu:~/home/hyw/pr5$ set USER hhh
hyw@ubuntu:~/home/hyw/pr5$ set
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1897,unix/ubuntu:/tmp/.ICE-unix/189
7
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
LANGUAGE=en_US:en
QT4_IM_MODULE=ibus
LC_ADDRESS=en_US.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1790
GTK_MODULES=gail:atk-bridge
PWD=/home/hyw/pr5
LOGNAME=hyw
```

(set后加参数表示修改环境变量，如下图所示，环境变量USER的值已被改变成hhh)

```
hyw@ubuntu: ~/pr5
TERM=xterm-256color
LC_IDENTIFICATION=en_US.UTF-8
LESSOPEN=| /usr/bin/lesspipe %s
USER=hhh
GNOME_TERMINAL_SERVICE=:1.94
DISPLAY=:0
SHLVL=1
LC_TELEPHONE=en_US.UTF-8
QT_IM_MODULE=ibus
LC_MEASUREMENT=en_US.UTF-8
PAPERSIZE=letter
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=en_US.UTF-8
JOURNAL_STREAM=9:50638
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
LC_NUMERIC=en_US.UTF-8
_=./myshell
OLDPWD=/home/hyw
hyw@ubuntu:/home/hyw/pr5$
```

```
hyw@ubuntu:/home/hyw/pr5$ unset USER
hyw@ubuntu:/home/hyw/pr5$ set
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1897,unix/ubuntu:/tmp/.ICE-unix/1897
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
LANGUAGE=en_US:en
QT4_IM_MODULE=ibus
LC_ADDRESS=en_US.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1790
GTK_MODULES=gail:atk-bridge
PWD=/home/hyw/pr5
LOGNAME=hyw
```

(输入unset删除环境变量，由下图，可以看到USER环境变量已经被删除)

```
XDG_SESSION_CLASS=user
TERM=xterm-256color
LC_IDENTIFICATION=en_US.UTF-8
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_TERMINAL_SERVICE=:1.94
DISPLAY=:0
SHLVL=1
LC_TELEPHONE=en_US.UTF-8
QT_IM_MODULE=ibus
LC_MEASUREMENT=en_US.UTF-8
PAPERSIZE=letter
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=en_US.UTF-8
JOURNAL_STREAM=9:50638
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
LC_NUMERIC=en_US.UTF-8
_=./myshell
OLDPWD=/home/hyw
hyw@ubuntu:/home/hyw/pr5$
```

8. 测试第六组命令。

```
Welcome to Myshell v1.0 for Ubuntu!
Author: Yanwei Huang
hyw@ubuntu:/home/hyw/pr5$ cat ./tmp.sh
ls
ps

hyw@ubuntu:/home/hyw/pr5$ exec ./tmp.sh
controller.c main.c Makefile mycmd.h proc_list.c tmp
controller.h main.h mycmd.c myshell proc_list.h tmp.sh
  PID TTY      TIME CMD
 4931 pts/0    00:00:01 bash
 5594 pts/0    00:00:00 myshell
 5598 pts/0    00:00:00 sh
 5600 pts/0    00:00:00 ps
hyw@ubuntu:~/pr5$
```

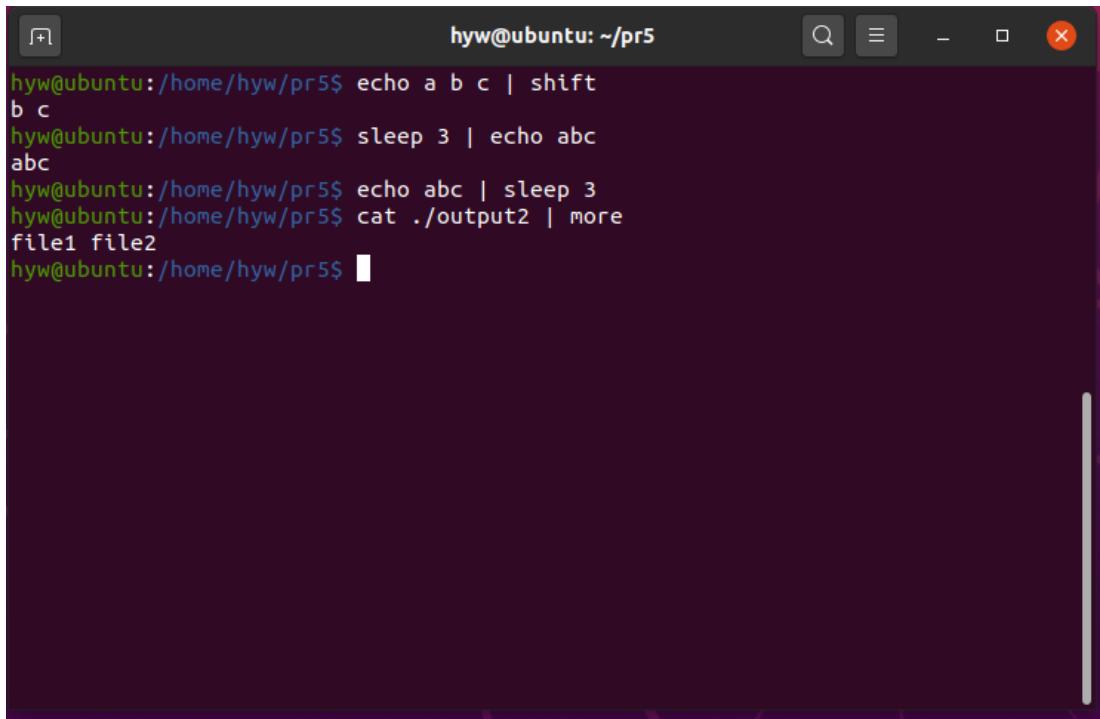
9. 执行外部命令+从文件中提取命令行输入

```
Welcome to Myshell v1.0 for Ubuntu!
Author: Yanwei Huang
hyw@ubuntu:/home/hyw/pr5$ ./tmp.sh
controller.c main.c Makefile mycmd.h proc_list.c tmp
controller.h main.h mycmd.c myshell proc_list.h tmp.sh
    PID TTY          TIME CMD
    4931 pts/0    00:00:01 bash
    5568 pts/0    00:00:00 myshell
    5571 pts/0    00:00:00 sh
    5573 pts/0    00:00:00 ps
hyw@ubuntu:/home/hyw/pr5$ find . -name *.sh
./tmp.sh
hyw@ubuntu:/home/hyw/pr5$ quit
hyw@ubuntu:~/pr5$ myshell tmp.sh
controller.c main.c Makefile mycmd.h proc_list.c tmp
controller.h main.h mycmd.c myshell proc_list.h tmp.sh
    PID TTY          TIME CMD
    4931 pts/0    00:00:01 bash
    5575 pts/0    00:00:00 myshell
    5577 pts/0    00:00:00 ps
hyw@ubuntu:~/pr5$
```

## 10. I/O重定向

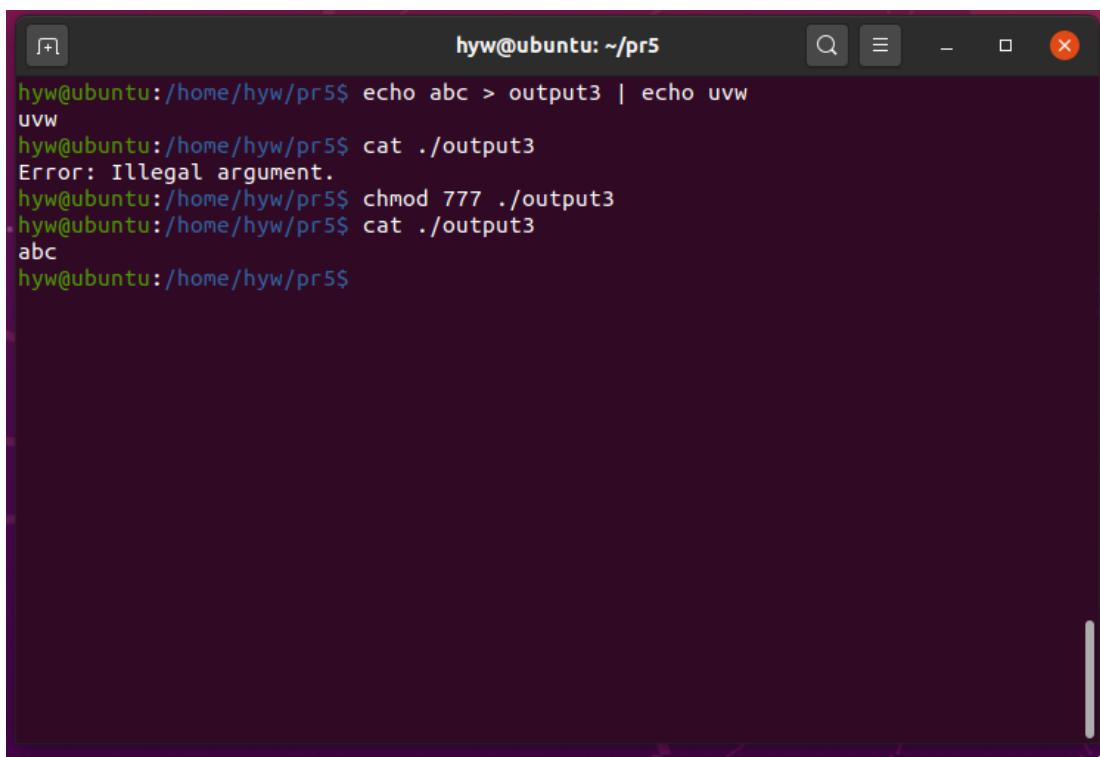
```
Welcome to Myshell v1.0 for Ubuntu!
Author: Yanwei Huang
hyw@ubuntu:/home/hyw/pr5$ cat output
abc
hyw@ubuntu:/home/hyw/pr5$ dir ./test > output
hyw@ubuntu:/home/hyw/pr5$ cat output
file3
file1
file2
hyw@ubuntu:/home/hyw/pr5$ dir ./test >> output
hyw@ubuntu:/home/hyw/pr5$ cat output
file3
file1
file2
file3
file1
file2
hyw@ubuntu:/home/hyw/pr5$ shift 4 < ./output > ./output2
hyw@ubuntu:/home/hyw/pr5$ chmod 777 ./output2
hyw@ubuntu:/home/hyw/pr5$ cat ./output2
file1 file2
hyw@ubuntu:/home/hyw/pr5$
```

## 11. 管道



```
hyw@ubuntu:~/pr5$ echo a b c | shift  
b c  
hyw@ubuntu:~/pr5$ sleep 3 | echo abc  
abc  
hyw@ubuntu:~/pr5$ echo abc | sleep 3  
hyw@ubuntu:~/pr5$ cat ./output2 | more  
file1 file2  
hyw@ubuntu:~/pr5$
```

管道中同样支持重定向，如下图所示。



```
hyw@ubuntu:~/pr5$ echo abc > output3 | echo uvw  
uvw  
hyw@ubuntu:~/pr5$ cat ./output3  
Error: Illegal argument.  
hyw@ubuntu:~/pr5$ chmod 777 ./output3  
hyw@ubuntu:~/pr5$ cat ./output3  
abc  
hyw@ubuntu:~/pr5$
```

## 12. 用more语句控制的阅读

输入environ | more , 按回车

```
hyw@ubuntu:~/pr5$ environ | more
```

```
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1897,unix/ubuntu:/tmp/.ICE-unix/1897
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
LANGUAGE=en_US:en
QT4_IM_MODULE=ibus
LC_ADDRESS=en_US.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1790
GTK_MODULES=gail:atk-bridge
PWD=/home/hyw/pr5
LOGNAME=hyw
XDG_SESSION_DESKTOP=ubuntu
```

```
more? [
```

输入空格，进入下一屏。

```
hyw@ubuntu: ~/pr5
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS_ERROR;JS_LOG
WINDOWPATH=2
HOME=/home/hyw
USERNAME=hyw
IM_CONFIG_PHASE=1
LC_PAPER=en_US.UTF-8
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd
=40;33:01:or=40;31:01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*
.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.m
p4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35
:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=0
more?■
```

输入回车，显示下一行。

```
hyw@ubuntu: ~/pr5
WINDOWPATH=2
HOME=/home/hyw
USERNAME=hyw
IM_CONFIG_PHASE=1
LC_PAPER=en_US.UTF-8
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd
=40;33:01:or=40;31:01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*
.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.m
p4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35
:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=0
more?■
1;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm
=01;35:*.emf=01;35:
```

输入字符"q"，退出阅读。

```

hyw@ubuntu: ~/pr5
HOME=/home/hyw
USERNAME=hyw
IM_CONFIG_PHASE=1
LC_PAPER=en_US.UTF-8
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z
=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.t
bz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.m
p4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35
:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=0
more?
1;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm
=01;35:*.emf=01;35:
more?q
hyw@ubuntu:/home/hyw/pr5$ 
```

## 完整代码

### Makefile

```

myshell: main.o proc_list.o controller.o mycmd.o
        gcc main.o proc_list.o controller.o mycmd.o -o myshell

main.o: main.h main.c proc_list.h
       gcc -c main.c -o main.o

controller.o: main.h mycmd.h proc_list.h controller.c controller.h
       gcc -c controller.c -o controller.o

mycmd.o: mycmd.c main.h proc_list.h mycmd.h
       gcc -c mycmd.c -o mycmd.o

proc_list.o: proc_list.h proc_list.c main.h
       gcc -c proc_list.c -o proc_list.o

clean:
        @rm -f *.o myshell

```

### main.h

```

#pragma once
#define BUF 100
#define MAX_PROC 64
#define MAX_ARG 20
#define MAX_LINE 64

```

```

#define PAGE_LEN 22

//错误代码
enum ErrMsg
{
    EXIT_OK,
    CURPATH_FAILED,
    HOSTNAME_FAILED,
    USRNAME_FAILED,
    ARG_TOO_MANY,
    ARG_TOO_FEW,
    ARG_WRONG,
    QUIT,
    FORK_FAILURE,
    SUBPROCESS_FAILURE,
    INFILE_MISSING,
    OUTFILE_MISSING,
    INFILE_DUPLICATED,
    OUTFILE_DUPLICATED,
    INFILE_NOT_EXIST,
    INFILE_CANNOT_READ,
    OUTFILE_CANNOT_WRITE,
    HOME_CANNOT_GET,
    SET_ERROR,
    UNSET_ERROR,
    CMD_ILLEGAL
};

//运行状态
enum ProcStatus
{
    RUNNING,
    STOPPED
};

int getusername(char *usr);
int getworkdir(char *dir);
void errproc(int err);

```

## main.c

```

/*
程序名: main.c
作者: 黄彦玮
学号: 3180102067
说明: myshell主程序、用户交互模块
完成时间: 2020-08-18
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

```

```
#include <fcntl.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <dirent.h>
#include <pwd.h>
#include "main.h"
#include "proc_list.h"
#include "controller.h"

char username[BUF];           //用户名
char hostname[BUF];          //主机名
char dirname[BUF];            //当前目录名
char cmd_raw[BUF];            //命令源字符串
char cmd_file[MAX_LINE][BUF]; //命令集（从外部文件中读入的）

int stdin_copy; //stdin文件描述符的备份，用于重定向还原

void init();

int main(int argc, char *argv[])
{
    init();
    FILE *fp;
    //判断参数个数
    if (argc > 2)
    {
        printf("Error: Too many arguments!\n");
        return 0;
    }
    else if (argc == 2) //参数为2，从文件中读入
    {
        fp = fopen(argv[1], "r");
        if (fp == NULL)
        {
            printf("Error: The file '%s' cannot be opened!\n", argv[1]);
            return 0;
        }
        memset(cmd_file, 0, sizeof(cmd_file)); //清空命令集
        int lines = 0;
        while (fgets(cmd_file[lines], BUF, fp) != NULL)
            lines++;
        int i;
        for (i = 0; i < lines; i++)
        {
            //分割指令，返回值为参数个数
            int num = ssplit(cmd_file[i]);
            //执行指令
            int res = execom(0, num);
            //处理错误信息
            err_proc(res);
            //exit/quit指令退出shell
        }
    }
}
```

```
        if (res == QUIT)
            return 0;
        //回收僵尸进程
        recycle_proc();
    }
    return 0;
}

//以下为参数为1的情况，从标准输入中读入命令
system("clear");
//打印欢迎信息
fprintf(stdout, "Welcome to Myshell v1.0 for Ubuntu!\nAuthor: Yanwei Huang\n");
while (1)
{
    //获取用户名
    if (getusername(username) == -1)
    {
        fprintf(stderr, "Error: Cannot get user name.");
        exit(USRNAME_FAILED);
    }
    //获取主机名
    if (gethostname(hostname, BUF) == -1)
    {
        fprintf(stderr, "Error: Cannot get host name.");
        exit(HOSTNAME_FAILED);
    }
    //获取当前目录
    if (getworkdir(dirname) == -1)
    {
        fprintf(stderr, "Error: Cannot get current path.");
        exit(CURPATH_FAILED);
    }
    //输出命令提示符
    fprintf(stdout, "\033[32m%s@\%s\033[0m", username, hostname);
    fprintf(stdout, ":");
    fprintf(stdout, "\033[34m$ \$ \033[0m", dirname);
    memset(cmd_raw, 0, sizeof(cmd_raw));
    //从标准输入中读入命令
    fgets(cmd_raw, BUF, stdin);
    //分割指令，返回值为参数个数
    int num = ssplit(cmd_raw);
    //执行指令
    int res = execom(0, num);
    //处理错误信息
    err_proc(res);
    //quit指令退出shell
    if (res == QUIT)
        return 0;
    //回收僵尸进程
    recycle_proc();
}
fprintf(stdout, "ByeBye\n\n");
return 0;
}
```

```
//初始化，备份标准输入，用于重定向还原
void init()
{
    stdin_copy = dup(0);
    /*char *shell_env = getenv("SHELL");
    char *cur_path = getenv("PATH");
    char tmp[BUF];
    strcpy(tmp, shell_env);
    int len = strlen(tmp);
    strcpy(tmp + len, ":");

    len++;
    strcpy(tmp + len, cur_path);
    len = strlen(tmp);
    strcpy(tmp + len, "/myshell");
    setenv("SHELL", tmp, 1);*/
}

//获取用户名
int getusername(char *usr)
{
    struct passwd *pwd = getpwuid(getuid());
    strcpy(usr, pwd->pw_name);
}

//获取当前目录
int getworkdir(char *dir)
{
    char *tmp = getcwd(dir, BUF);
    if (tmp == NULL)
        return -1;
    return 0;
}

//错误信息处理
void err_proc(int err)
{
    switch (err)
    {
        case EXIT_OK:
        case QUIT:
            break;
        case ARG_TOO_MANY:
            fprintf(stderr, "Error: Too many arguments.\n");
            break;
        case ARG_TOO_FEW:
            fprintf(stderr, "Error: Too few arguments.\n");
            break;
        case ARG_WRONG:
            fprintf(stderr, "Error: Illegal argument.\n");
            break;
        case CURPATH_FAILED:
            fprintf(stderr, "Error: Cannot get current path.\n");
            break;
        case HOSTNAME_FAILED:
            fprintf(stderr, "Error: Cannot get host name.\n");
    }
}
```

```

        break;
case USRNAME_FAILED:
    fprintf(stderr, "Error: Cannot get user name.\n");
    break;
case FORK_FAILURE:
    fprintf(stderr, "Error: Failed to fork a process.\n");
    break;
case SUBPROCESS_FAILURE:
    fprintf(stderr, "Error: The subprocess exits unexpectedly.\n");
    break;
case INFILE_MISSING:
    fprintf(stderr, "Error: Input file needed when redirecting standard input.\n");
    break;
case OUTFILE_MISSING:
    fprintf(stderr, "Error: Output file needed when redirecting standard output.\n");
    break;
case INFILE_DUPLICATED:
    fprintf(stderr, "Error: More than one option is given when redirecting standard input.\n");
    break;
case OUTFILE_DUPLICATED:
    fprintf(stderr, "Error: More than one option is given when redirecting standard output.\n");
    break;
case INFILE_NOT_EXIST:
    fprintf(stderr, "Error: Input file doesn't exist when redirecting standard input.\n");
    break;
case INFILE_CANNOT_READ:
    fprintf(stderr, "Error: Input file cannot be read when redirecting standard input.\n");
    break;
case OUTFILE_CANNOT_WRITE:
    fprintf(stderr, "Error: Output file cannot be written when redirecting standard output.\n");
    break;
case HOME_CANNOT_GET:
    fprintf(stderr, "Error: Cannot get home path.\n");
    break;
case SET_ERROR:
    fprintf(stderr, "Error: Failed to set the environment variable.\n");
    break;
case UNSET_ERROR:
    fprintf(stderr, "Error: Failed to unset the environment variable.\n");
    break;
case CMD_ILLEGAL:
    fprintf(stderr, "Error: Unidentified command type.\n");
    break;
}
}

```

## controller.h

```
#pragma once
int sspli(char *cmd_raw);
int execom(int l, int r);
int execom_without_pipe(int l, int r);
```

## controller.c

```
/*
程序名: controller.c
作者: 黄彦玮
学号: 3180102067
说明: myshell命令解析与进程控制模块
完成时间: 2020-08-18
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <dirent.h>
#include <pwd.h>
#include "main.h"
#include "proc_list.h"
#include "controller.h"
#include "mycmd.h"

char arg[MAX_ARG][BUF];
extern int stdin_copy;
extern struct proc_item proc_list[MAX_PROC];

//将命令按空格分割
int sspli(char *cmd_raw)
{
    memset(arg, 0, sizeof(arg));
    //i:源字符串下标 len:源字符串长度 num:参数个数 tmp:当前参数长度 flag:当前扫描到的字符是否是参数的一部分
    int i = 0, len = strlen(cmd_raw), num = 0, tmp = 0, flag = 0;
    for (; i < len; i++)
    {
        if (cmd_raw[i] == ' ' || cmd_raw[i] == '\n') //空格或回车标志着一个参数的结束
        {
            flag = 0;
            tmp = 0;
        }
        else
        {
```

```

        if (!flag) //flag=0说明当前读到的字符是新参数的第一个字符
        {
            flag = 1;
            num++;
        }
        arg[num - 1][tmp++] = cmd_raw[i];
    }
}

//执行命令，参数下标范围为[1, r)
int execom(int l, int r)
{
    if (l >= r)
        return EXIT_OK; //正常退出
    int pip_pos;           //pip_pos:从左至右第一个管道符号的位置
    //判断命令中是否含有管道符号
    for (pip_pos = 1; pip_pos < r; pip_pos++)
    {
        if (strcmp(arg[pip_pos], "|") == 0)
            break;
    }
    //管道符号在最左侧或者最右侧都是不合法的
    if (pip_pos == 1 || pip_pos == r - 1)
        return ARG_TOO_FEW; //参数过少
    else if (pip_pos == r) //无管道，调用无管道执行函数
        return execom_without_pipe(l, r);
    int file_pipes[2];           //管道文件描述符
    pid_t fork_result;          //新建的子进程的进程编号pid
    if (pipe(file_pipes) == 0) //新建管道
    {
        fork_result = fork();   //新建子进程
        if (fork_result == -1) //新建进程失败
            return FORK_FAILURE;
        else if (fork_result == 0) //子进程
        {
            close(file_pipes[0]);           //关闭管道读入端，因为不会
用到
用到
端
述符
程，返回错误代码
        }
        else //父进程
        {
            close(file_pipes[1]); //关闭管道输出端，因为不会用到
            int stat_val;
            //前台进程，父进程阻塞
            waitpid(fork_result, &stat_val, WUNTRACED);
        }
    }
}

```

```

        if (WIFEXITED(stat_val)) //子进程正常退出（指正常通过exit()函数退出，或main()中
return 0退出)
    {
        int exit_code = WEXITSTATUS(stat_val); //获取子进程错误代码（即exit()中的
值）
        if (exit_code != EXIT_OK) //子进程的指令没有正常执行
        {
            close(file_pipes[0]); //关闭管道读入端
            return exit_code; //返回错误代码
        }
        else //子进程正常执行完成
        {
            dup2(file_pipes[0], 0); //重定向标准输入到管道读入
端，这样父进程就可以读入子进程输出的数据
            int res = execom(pip_pos + 1, r); //递归执行后续指令
            close(file_pipes[0]); //关闭管道输入端
            dup2(stdin_copy, 0); //恢复重定向，将0号文件描述符重定向到标准输入的备份
            return res;
        }
    }
    else
        return SUBPROCESS_FAILURE; //子进程终止或异常退出
}
}

//执行不带管道的指令，参数下标范围为[1,r)
int execom_without_pipe(int l, int r)
{
    if (l >= r)
        return EXIT_OK;
/*输入输出重定向处理*/
    int i;
    char infile[BUF]; //重定向输入文件名
    char outfile[BUF]; //重定向输出文件名
    int outflag = 0; //重定向符号为">"时为0，为">>"时为1
    int isbg = (strcmp(arg[r - 1], "&") == 0); //是否是后台指令，若是则为1
    int newr = r; //指令在重定向前的下标
    memset(infile, 0, sizeof(infile));
    memset(outfile, 0, sizeof(outfile));
//判断命令中是否含有重定向
    for (i = l; i < r; i++)
    {
        if (strcmp(arg[i], "<") == 0) //重定向输入
        {
            if (i + 1 >= r)
                return INFILE_MISSING; //输入文件不存在
            else if (infile[0])
                return INFILE_DUPLICATED; //有多个输入重定向
            else
                strcpy(infile, arg[i + 1]);
            if (newr == r)
                newr = i;
        }
    }
}

```

```

        if (strcmp(arg[i], ">") == 0 || strcmp(arg[i], ">>") == 0) //重定向输出
        {
            if (strcmp(arg[i], ">>") == 0)
                outflag = 1;
            if (i + 1 >= r)
                return OUTFILE_MISSING; //输出文件不存在
            else if (outfile[0])
                return OUTFILE_DUPLICATED; //有多个输出重定向
            else
                strcpy(outfile, arg[i + 1]);
            if (newr == r)
                newr = i;
        }
    }

    if (strcmp(arg[newr - 1], "&") == 0)
        newr--;
    if (infile[0]) //输入重定向存在
    {
        if (access(infile, F_OK) == -1)
            return INFILE_NOT_EXIST; //输入文件不存在
        else if (access(infile, R_OK) == -1)
            return INFILE_CANNOT_READ; //输入文件没有读权限
    }
    if (outfile[0]) //输出重定向存在
    {
        if (access(outfile, F_OK) != -1 && access(outfile, W_OK) == -1)
            return OUTFILE_CANNOT_WRITE; //输出文件存在但没有写权限
    }

/*指令执行*/
/*先执行内部指令（必须要在父进程执行的指令）*/
if (strcmp(arg[1], "cd") == 0)
    return execd(1, newr);
if (strcmp(arg[1], "clr") == 0)
    return execlr(1, newr);
if (strcmp(arg[1], "quit") == 0 || strcmp(arg[1], "exit") == 0)
    return exexit(1, newr);
if (strcmp(arg[1], "bg") == 0)
    return exebg(1, newr);
if (strcmp(arg[1], "fg") == 0)
    return exefg(1, newr);
if (strcmp(arg[1], "set") == 0)
    return exeset(1, newr);
if (strcmp(arg[1], "unset") == 0)
    return exeunset(1, newr);
if (strcmp(arg[1], "umask") == 0)
    return exeumask(1, newr);
if (strcmp(arg[1], "exec") == 0)
    return eeexec(1, newr);

/*外部指令，创建子进程后执行*/
//先处理重定向
int in_fd = 0, out_fd = 1;
if (infile[0]) //输入重定向存在

```

```

{
    in_fd = open(infile, O_RDONLY); //打开输入文件
    if (in_fd == -1)
        return INFILE_CANNOT_READ; //输入文件无法打开
}
if (outfile[0])
{
    //创建输出文件, outflag = 1时打开的文件是附加模式, 否则为截断模式
    if (outflag)
        out_fd = open(outfile, O_WRONLY | O_APPEND | O_CREAT /*, S_IRUSR | S_IWUSR |
S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH*/);
    else
        out_fd = open(outfile, O_WRONLY | O_TRUNC | O_CREAT /*, S_IRUSR | S_IWUSR |
S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH*/);
    if (out_fd == -1)
        return OUTFILE_CANNOT_WRITE; //输出文件存在但无写权限
}

//创建子进程
pid_t fork_result;
fork_result = fork();
if (fork_result == -1) //创建进程失败
    return FORK_FAILURE;
else if (fork_result == 0) //子进程
{
    //实现重定向
    if (infile[0])
    {
        close(0);
        dup(in_fd);
    }
    if (outfile[0])
    {
        close(1);
        dup(out_fd);
    }
    //执行指令
    if (strcmp(arg[1], "pwd") == 0)
        err_proc(exepwd(1, newr)), exit(0);
    else if (strcmp(arg[1], "dir") == 0)
        err_proc(exedir(1, newr)), exit(0);
    else if (strcmp(arg[1], "echo") == 0)
        err_proc(exeecho(1, newr)), exit(0);
    else if (strcmp(arg[1], "environ") == 0)
        err_proc(exeenviron(1, newr)), exit(0);
    else if (strcmp(arg[1], "help") == 0)
        err_proc(exehelp(1, newr)), exit(0);
    else if (strcmp(arg[1], "jobs") == 0)
        err_proc(exejobs(1, newr)), exit(0);
    else if (strcmp(arg[1], "shift") == 0)
        err_proc(exeshift(1, newr)), exit(0);
    else if (strcmp(arg[1], "test") == 0)
        err_proc(exetest(1, newr)), exit(0);
    else if (strcmp(arg[1], "time") == 0)
        err_proc(exetime(1, newr)), exit(0);
}

```

```

        else if (strcmp(arg[1], "sleep") == 0)
            err_proc(exesleep(1, newr)), exit(0);
        else if (strcmp(arg[1], "cat") == 0)
            err_proc(execat(1, newr)), exit(0);
        else if (strcmp(arg[1], "more") == 0)
            err_proc(exemore(1, newr)), exit(0);
        else //外部命令
    {
        char *tmp[MAX_ARG]; //提取外部命令
        int i;
        for (i = 1; i < r; i++)
        {
            tmp[i - 1] = (char *)malloc(BUF);
            strcpy(tmp[i - 1], arg[i]);
        }
        tmp[r - 1] = NULL;
        //调用系统调用执行外部命令
        int res = execvp(arg[1], tmp);
        if (res == -1)
            exit(CMD_ILLEGAL); //执行失败，指令类型不合法
        exit(0);
    }
}
else //父进程
{
    int workid = proc_add(fork_result, arg[1], isbg, RUNNING);
    if (!isbg) //前台进程需要等待结束
    {
        //注册信号函数，用于捕捉Ctrl+Z输入
        signal(SIGTSTP, CtrlZHandler);
        int stat_val;
        //前台进程，父进程阻塞。这里用了WUNTRACED是为了在子进程在Ctrl+Z作用下停止的情况下也能正常返回
        waitpid(fork_result, &stat_val, WUNTRACED);
        //恢复信号函数
        signal(SIGTSTP, SIG_DFL);
        if (WIFEXITED(stat_val)) //子进程正常退出（指正常通过exit()函数退出，或main()中return 0退出）
        {
            int exit_code = WEXITSTATUS(stat_val); //获取错误代码
            //err_proc(exit_code); //输出子进程错误信息（可选）
            proc_del(workid); //从进程表中删除对应的进程
            return EXIT_OK;
        }
        else if (WIFSTOPPED(stat_val)) //子进程停止（指子进程在Ctrl+Z作用下停止）
        {
            //输出子进程信息
            printf("[%d]%d Stopped %s\n", workid + 1, proc_list[workid].pid,
proc_list[workid].proc_name);
            return EXIT_OK;
        }
        else
            return SUBPROCESS_FAILURE; //子进程异常退出
    }
}

```

```

        }
    else
    {
        //后台进程直接打印进程信息
        printf("[%d] %d %s\n", workid + 1, proc_list[workid].pid,
(proc_list[workid].status == RUNNING) ? "Running" : "Stopped", proc_list[workid].proc_name);
        return EXIT_OK;
    }
}
}

```

## mycmd.h

```

#pragma once
int execom(int l, int r);
int execd(int l, int r);
int exedir(int l, int r);
int exetime(int l, int r);
int exepwd(int l, int r);
int execom_without_pipe(int l, int r);
int exeenvironment(int l, int r);
int execho(int l, int r);
int exexit(int l, int r);
int exejobs(int l, int r);
int exebg(int l, int r);
int exefg(int l, int r);
int exeset(int l, int r);
int exeunset(int l, int r);
int exumask(int l, int r);
int exexec(int l, int r);
int exetest(int l, int r);
int exeshift(int l, int r);
int exehelp(int l, int r);
int exesleep(int l, int r);
int exemore(int l, int r);
int execat(int l, int r);

```

## mycmd.c

```

/*
程序名: mycmd.c
作者: 黄彦玮
学号: 3180102067
说明: myshell命令执行模块
完成时间: 2020-08-18
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

```

```
#include <fcntl.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <dirent.h>
#include <pwd.h>
#include "mycmd.h"
#include "main.h"
#include "proc_list.h"

extern char **environ; //环境变量
extern char dirname[BUF]; //当前目录名称
extern int proc_num; //进程表大小
extern char arg[MAX_ARG][BUF]; //分割后的命令
extern struct proc_item proc_list[MAX_PROC]; //进程表
extern int stdin_copy; //stdin文件描述符的备份

//cd指令
int execd(int l, int r)
{
    //检查参数个数，下同
    if (r - 1 > 2)
        return ARG_TOO_MANY; //参数过多
    if (r - 1 == 2) //2个参数时，修改目录到参数对应目录
    {
        int res = chdir(arg[l + 1]);
        if (res)
            return ARG_WRONG; //参数错误（路径打不开）
    }
    else //1个参数时，切换到主目录
    {
        int res = chdir(getenv("HOME"));
        if (res)
            return HOME_CANNOT_GET; //无法获取主目录路径
    }
    return EXIT_OK; //正常退出
}

//clr指令
int execlr(int l, int r)
{
    if (r - 1 > 1)
        return ARG_TOO_MANY;
    //清屏
    printf("\033[1H\033[2J");
    return EXIT_OK;
}

//pwd指令
int exepwd(int l, int r)
{
    if (r - 1 > 1)
        return ARG_TOO_MANY;
```

```

//获取当前目录名称
if (getworkdir(dirname) == -1)
    return CURPATH_FAILED; //无法获取当前路径
//输出当前目录名称
printf("%s\n", dirname);
return EXIT_OK;
}

//time指令
int exetime(int l, int r)
{
    if (r - 1 > 1)
        return ARG_TOO_MANY;
    time_t *timep = malloc(sizeof(*timep));
    //获取时间戳并转化成字符形式
    time(timep);
    char *s = ctime(timep);
    printf("%s", s);
    return EXIT_OK;
}

//dir指令
int exedir(int l, int r)
{
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    else if (r - 1 < 2)
        return ARG_TOO_FEW; //参数过少
    DIR *dp;
    struct dirent *entry;
    struct stat statbuf;
    //调用系统调用，新建目录流
    if ((dp = opendir(arg[1 + 1])) == NULL)
        return ARG_WRONG;
    //切换到对应文件夹
    chdir(arg[1 + 1]);
    //依次从目录流中读取目录下文件信息
    while ((entry = readdir(dp)) != NULL)
    {
        //获取文件属性，d_name是文件名，statbuf是文件属性，其中statbuf.st_mode是文件类型
        lstat(entry->d_name, &statbuf);
        //根据不同的文件类型采取不同的输出方式
        if (S_ISDIR(statbuf.st_mode)) //目录，蓝色
        {
            if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
                continue;
            printf("\033[34m%s$ \033[0m\n", entry->d_name);
        }
        else if (S_ISLNK(statbuf.st_mode)) //符号链接，绿色
        {
            printf("\033[32m%s$ \033[0m\n", entry->d_name);
        }
        else if (S_ISREG(statbuf.st_mode)) //普通文件，白色
        {
            printf("%s\n", entry->d_name);
        }
    }
}

```

```

        }
        else //其他文件，红色
        {
            printf("\033[31m%s$ \033[0m\n", entry->d_name);
        }
    }

    //关闭目录流
    closedir(dp);
    return EXIT_OK;
}

//environ指令
int exeenviron(int l, int r)
{
    if (r - 1 > 1)
        return ARG_TOO_MANY;
    //从指向环境变量的外部变量environ中不断读取环境变量信息
    char **env = environ;
    while (*env)
    {
        printf("%s\n", *env);
        env++;
    }
    return EXIT_OK;
}

//echo指令
int exeecho(int l, int r)
{
    if (r - 1 == 1) //无参数直接输出回车
        printf("\n");
    else //有参数按顺序输出参数，多个空格合并
    {
        int i;
        for (i = l + 1; i < r; i++)
        {
            printf("%s", arg[i]);
            if (i != r - 1)
                printf(" ");
            else
                printf("\n");
        }
    }
    return EXIT_OK;
}

//exit指令或quit指令
int exeexit(int l, int r)
{
    return QUIT; //退出Shell
}

//jobs指令
int exejobs(int l, int r)
{

```

```

    if (r - 1 > 1)
        return ARG_TOO_MANY;
    int i;
    //遍历进程表，打印进程信息
    for (i = 0; i < proc_num; i++)
    {
        printf("[%d] %4d %s %s\n", i + 1, proc_list[i].pid, (proc_list[i].status == RUNNING) ? "Running" : "Stopped", proc_list[i].proc_name);
    }
    return EXIT_OK;
}

//bg指令
int exebg(int l, int r)
{
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    if (r - 1 < 2)
        return ARG_TOO_FEW;
    //将用户输入的作业号转换成int类型
    int id = atoi(arg[l + 1]);
    if (id > 0 && id <= proc_num)
    {
        //由于输出时作业号从1开始编号，而进程表中储存时从0开始编号，所以这里编号要-1
        id--;
        //对于停止的进程，先更新进程表，然后向其发送SIGCONT信号令其继续运行
        if (proc_list[id].pid && proc_list[id].status == STOPPED)
        {
            proc_list[id].status = RUNNING;
            proc_list[id].isbg = 1; //后台运行
            kill(proc_list[id].pid, SIGCONT);
            //输出进程信息
            printf("[%d] %s\n", proc_list[id].pid, proc_list[id].proc_name);
        }
        return EXIT_OK;
    }
    else
        return ARG_WRONG;
}

//fg指令
int exefg(int l, int r)
{
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    if (r - 1 < 2)
        return ARG_TOO_FEW;
    //将用户输入的作业号转换成int类型
    int id = atoi(arg[l + 1]);
    if (id > 0 && id <= proc_num)
    {
        //由于输出时作业号从1开始编号，而进程表中储存时从0开始编号，所以这里编号要-1
        id--;
        //后台进程转前台
        if (proc_list[id].pid && proc_list[id].isbg)

```

```

{
    //更新进程表
    proc_list[id].isbg = 0;
    //停止的进程，更新进程表后发送SIGCONT信号令其继续运行
    if (proc_list[id].status == STOPPED)
        kill(proc_list[id].pid, SIGCONT), proc_list[id].status = RUNNING;
    //输出后台进程信息
    printf("[%d] %4d %s\n", id + 1, proc_list[id].pid, (proc_list[id].status == RUNNING) ? "Running" : "Stopped", proc_list[id].proc_name);
    //注册信号函数，用于捕捉Ctrl+Z输入
    signal(SIGTSTP, CtrlZHandler);
    int stat_val;
    //前台进程，父进程阻塞。这里用了WUNTRACED是为了在子进程在Ctrl+Z作用下停止的情况下也能正常返回
    waitpid(proc_list[id].pid, &stat_val, WUNTRACED);
    //恢复信号函数
    signal(SIGTSTP, SIG_DFL);
    if (WIFEXITED(stat_val)) //子进程正常退出（指正常通过exit()函数退出，或main()中return 0退出）
    {
        int exit_code = WEXITSTATUS(stat_val); //获取错误代码
        //err_proc(exit_code); //输出子进程错误信息（可选）
        proc_del(id); //从进程表中删除对应的进程
        return EXIT_OK;
    }
    else if (WIFSTOPPED(stat_val)) //子进程停止（指子进程在Ctrl+Z作用下停止）
    {
        //输出子进程信息
        printf("[%d] %4d Stopped %s\n", id + 1, proc_list[id].pid, proc_list[id].proc_name);
        return EXIT_OK;
    }
    else
        return SUBPROCESS_FAILURE; //子进程异常退出
}
return EXIT_OK;
}

//set指令
int exeset(int l, int r)
{
    if (r - 1 == 1) //无参数输出环境变量
        return exenvironment(l, r);
    if (r - 1 < 3)
        return ARG_TOO_FEW;
    else if (r - 1 > 3)
        return ARG_TOO_MANY;
    //有参数设置环境变量
    int res = setenv(arg[1 + l], arg[1 + l + 1], 1);
    if (res == -1)

```

```

        return SET_ERROR; //设置失败
    else
        return EXIT_OK;
}

//unset指令
int exeunset(int l, int r)
{
    if (r - 1 < 2)
        return ARG_TOO_FEW;
    else if (r - 1 > 2)
        return ARG_TOO_MANY;
    //删除环境变量
    int res = unsetenv(arg[1 + 1]);
    if (res == -1)
        return UNSET_ERROR; //删除失败
    else
        return EXIT_OK;
}

//umask指令
int exumask(int l, int r)
{
    if (r - 1 < 2) //无参数输出旧掩码
    {
        mode_t mask;
        mask = umask(0002);           //先随便设一个新掩码，得到旧掩码
        umask(mask);                 //再恢复掩码值
        printf("%04d\n", mask); //输出旧掩码值
        return EXIT_OK;
    }
    else if (r - 1 > 2)
        return ARG_TOO_MANY;
    //有参数设置新掩码
    umask(atoi(arg[1 + 1]));
    return EXIT_OK;
}

//exec指令
int exexec(int l, int r)
{
    //调用系统调用执行，执行结束后退出
    int res = execom(l + 1, r);
    err_proc(res);
    return QUIT;
}

//test指令
int exetest(int l, int r)
{
    if (r - 1 < 4)
        return ARG_TOO_FEW;
    if (r - 1 > 4)
        return ARG_TOO_MANY;
    //比较字符串
}

```

```

    if (strcmp(arg[1 + 2], "!=") == 0 || strcmp(arg[1 + 2], "==") == 0)
    {
        printf(!strcmp(arg[1 + 1], arg[1 + 3]) ? "True\n" : "False\n");
    }
    else if (strcmp(arg[1 + 2], "!=") == 0)
    {
        printf(strcmp(arg[1 + 1], arg[1 + 3]) ? "True\n" : "False\n");
    }
    else
        return ARG_WRONG;
    return EXIT_OK;
}

//shift指令
int exeshift(int l, int r)
{
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    int num = 1; //默认移动1位
    if (r - 1 == 2)
        num = atoi(arg[1 + 1]); //获取移动位数
    char tmp[MAX_ARG][BUF];
    int arg_num = 0; //参数个数
    //从标准输入读取参数
    while (scanf("%s", tmp[arg_num]) != EOF)
        arg_num++;
    //输出移位结果
    int i;
    for (i = num; i < arg_num; i++)
    {
        printf("%s", tmp[i]);
        if (i != arg_num - 1)
            printf(" ");
        else
            printf("\n");
    }
    return EXIT_OK;
}

//help指令
int exehelp(int l, int r)
{
    if (r - 1 < 2)
        return ARG_TOO_FEW;
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    if (strcmp(arg[1 + 1], "cd") == 0)
    {
        printf("cd -- Change directory\nUsage: cd $path\n");
    }
    else if (strcmp(arg[1 + 1], "pwd") == 0)
    {
        printf("pwd -- Print working directory\nUsage: pwd\n");
    }
    else if (strcmp(arg[1 + 1], "bg") == 0)

```

```

{
    printf("bg -- Turn a process to be executed background\nUsage: bg <id> #<id> can be
derived from 'jobs' command\n");
}
else if (strcmp(arg[1 + 1], "clr") == 0)
{
    printf("clr -- Clear the screen\nUsage: clr\n");
}
else if (strcmp(arg[1 + 1], "time") == 0)
{
    printf("time -- Print the current time\nUsage: time\n");
}
else if (strcmp(arg[1 + 1], "dir") == 0)
{
    printf("dir -- List the files under a given directory\nUsage: dir $path\n");
}
else if (strcmp(arg[1 + 1], "environ") == 0)
{
    printf("environ -- List all environment variables\nUsage: env\n");
}
else if (strcmp(arg[1 + 1], "echo") == 0)
{
    printf("echo -- Print the parameters and then print an enter\nUsage: echo [para1]
[para2] ... \n");
}
else if (strcmp(arg[1 + 1], "exit") == 0)
{
    printf("exit -- Quit the shell\nUsage: exit\n");
}
else if (strcmp(arg[1 + 1], "quit") == 0)
{
    printf("quit -- Quit the shell\nUsage: quit\n");
}
else if (strcmp(arg[1 + 1], "jobs") == 0)
{
    printf("jobs -- Print all subprocesses\nUsage: jobs\n");
}
else if (strcmp(arg[1 + 1], "fg") == 0)
{
    printf("fg -- Turn a process to be executed foreground\nUsage: bg <id> #<id> can be
derived from 'jobs' command\n");
}
else if (strcmp(arg[1 + 1], "set") == 0)
{
    printf("set -- Print all environment variables / Set the given environment
variable\nUsage: set [$env_name $env_val]\n");
}
else if (strcmp(arg[1 + 1], "unset") == 0)
{
    printf("unset -- Delete the given environment variable\nUsage: unset [$env_name]\n");
}
else if (strcmp(arg[1 + 1], "umask") == 0)
{
    printf("umask -- Print the umask / Set the umask\nUsage: umask [$new_val]\n");
}
}

```

```

    else if (strcmp(arg[1 + 1], "test") == 0)
    {
        printf("test -- Compare the strings\nUsage: test $str1 {=,!==} $str2\n");
    }
    else if (strcmp(arg[1 + 1], "shift") == 0)
    {
        printf("shift -- Shift parameters to left\nUsage: shift [$val] (default: $val = 1)\n");
    }
    else if (strcmp(arg[1 + 1], "exec") == 0)
    {
        printf("exec -- Replace the current process with a given command\nUsage: exec $command\n");
    }
    else if (strcmp(arg[1 + 1], "sleep") == 0)
    {
        printf("sleep -- Sleep for some time\nUsage: sleep [$time] (in seconds)\n");
    }
    else if (strcmp(arg[1 + 1], "cat") == 0)
    {
        printf("cat -- Print a given file\nUsage: cat $path\n");
    }
    else if (strcmp(arg[1 + 1], "more") == 0)
    {
        printf("more -- Filter the output\nUsage: more [$path] (or) $cmd ... | more\n");
    }
}

//sleep指令
int exesleep(int l, int r)
{
    if (r - 1 < 2)
        return ARG_TOO_FEW;
    if (r - 1 > 2)
        return ARG_TOO_MANY;
//调用系统调用睡眠
    sleep(atoi(arg[1 + 1]));
    return EXIT_OK;
}

//cat指令
int execat(int l, int r)
{
    if (r - 1 < 2)
        return ARG_TOO_FEW;
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    FILE *fp;
    fp = fopen(arg[1 + 1], "r"); //打开文件流
    if (fp == NULL)
        return ARG_WRONG; //文件打开失败，参数错误
    char line[BUF];
    while (fgets(line, BUF, fp))
    {
        printf("%s", line); //逐行输出文件内容
}

```

```

    }

    fclose(fp); //关闭文件流
    return EXIT_OK;
}

//more指令
int exemore(int l, int r)
{
    if (r - 1 > 2)
        return ARG_TOO_MANY;
    char line[BUF];
    int num_of_lines = 0; //当前已经打印的行数（从一屏幕的开始开始计算）
    if (r - 1 == 2) //有参数，从文件中读入
    {
        FILE *fp;
        fp = fopen(arg[l + 1], "r"); //打开文件流
        if (fp == NULL)
            return ARG_WRONG;
        while (fgets(line, BUF, fp)) //从文件流中读入数据
        {
            if (num_of_lines == PAGE_LEN) //屏幕已满
            {
                printf("\033[32m\nmore?\033[0m"); //输出提示信息等待用户输入
                int reply;
                while ((reply = getc(stdin)) != EOF) //获得用户输入
                {
                    if (reply == 'q' || reply == ' ' || reply == '\n') //若不是这三种字符则输入不合法，要求用户继续输入
                    {
                        if (reply != '\n') //除回车外，要再额外读一个用户输入的回车符
                            getc(stdin);
                        break;
                    }
                }
                if (reply == 'q') //字符q表示退出阅读
                    break;
                else if (reply == ' ')
                    reply = PAGE_LEN;
                else if (reply == '\n') //回车表示显示下一行
                    reply = 1;
                num_of_lines -= reply;
            }
            //输出当前读到的行
            fputs(line, stdout);
            num_of_lines++;
        }
        fclose(fp);
        return EXIT_OK;
    }
    //无参数，用于管道，从重定向后的标准输入读（通常是管道的一端）
    //注意此时用户输入还是输入到屏幕中的，所以在读用户输入的时候，要根据备份的stdin文件描述符新建文件流，从中读取用户输入
    //详见下面的注释
    else

```

```

{
    //调用fdopen, 使用stdin的备份文件描述符打开文件流
    FILE *fp = fdopen(stdin_copy, "r");
    if (fp == NULL)
        return SUBPROCESS_FAILURE;
    while (fgets(line, BUF, stdin)) //从重定向后的标准输入（管道）中读取要查看的文件内容
    {
        if (num_of_lines == PAGE_LEN) //屏幕已满
        {
            printf("\033[32m\nmore?\033[0m"); //输出提示信息等待用户输入
            int reply;
            while ((reply = getc(fp)) != EOF) //从[屏幕]获得用户输入, 注意这里的来源是
                fp, 即使用stdin的备份文件描述符打开文件流
            {
                if (reply == 'q' || reply == ' ' || reply == '\n') //若不是这三种字
                    符则输入不合法, 要求用户继续输入
                {
                    if (reply != '\n') //除回车外, 要再额外读一个用户输入的回车符
                        getc(fp);
                    break;
                }
            }
            if (reply == 'q') //字符q表示退出阅读
                break;
            else if (reply == ' ')
                reply = PAGE_LEN;
            else if (reply == '\n') //回车表示显示下一行
                reply = 1;
            num_of_lines -= reply;
        }
        fputs(line, stdout);
        num_of_lines++;
    }
    fclose(fp);
    return EXIT_OK;
}
}

```

## proc\_list.h

```

#pragma once
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/signalf.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "main.h"
//进程表中的项

```

```

struct proc_item
{
    pid_t pid;           //进程编号(唯一)
    char proc_name[BUF]; //进程名
    int isbg;           //是否后台
    int status;          //进程状态
};

int proc_add(pid_t pid, char *proc_name, int isbg, int status);
void proc_del(int id);
void recycle_proc();
void CtrlZHandler(int sig);

```

## proc\_list.c

```

/*
程序名: proc_list.c
作者: 黄彦玮
学号: 3180102067
说明: myshell辅助模块(包括进程表和信号控制)
完成时间: 2020-08-18
*/
#include <string.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "proc_list.h"
#include "main.h"

struct proc_item proc_list[MAX_PROC];
int proc_num = 0;
//向进程表里添加项
int proc_add(pid_t pid, char *proc_name, int isbg, int status)
{
    proc_list[proc_num].pid = pid;
    memset(proc_list[proc_num].proc_name, 0, sizeof(proc_list[proc_num].proc_name));
    strcpy(proc_list[proc_num].proc_name, proc_name);
    proc_list[proc_num].isbg = isbg;
    proc_list[proc_num++].status = status;
    return proc_num - 1;
}

//删除进程表中下标为id的项
void proc_del(int id)
{
    int i;
    for (i = id; i < proc_num - 1; i++)
    {

```

```

        proc_list[i].pid = proc_list[i + 1].pid;
        proc_list[i].isbg = proc_list[i + 1].isbg;
        proc_list[i].status = proc_list[i + 1].status;
        strcpy(proc_list[i].proc_name, proc_list[i + 1].proc_name);
    }
    proc_num--;
}

//回收僵尸进程
void recycle_proc()
{
    int i;
    for (i = 0; i < proc_num; i++)
    {
        int stat_val;
        //回收僵尸进程，若检测到进程已终止则waitpid函数会自动回收
        //注意这里使用WNOHANG，目的是为了在不阻塞的情况下检查进程状态
        if (waitpid(proc_list[i].pid, &stat_val, WNOHANG) != 0)
        {
            printf("[Finished] %d\n", proc_list[i].pid);
            //已经终止的进程，从进程表中删除
            proc_del(i);
            i--;
        }
    }
}

//处理Ctrl-Z信号
void CtrlZHandler(int sig)
{
    //进程表为空则返回
    if (proc_num == 0)
        return;
    //获取子进程的pid，并发送SIGSTOP信号，令其停止
    pid_t pid = proc_list[proc_num - 1].pid;
    kill(pid, SIGSTOP);
    //更新进程表
    proc_list[proc_num - 1].status = STOPPED;
    proc_list[proc_num - 1].isbg = 1;
}

```

## 实验心得

首先，我觉得通过Linux程序设计这门课的学习，我对Linux操作系统有了基本的了解，并且能够熟练掌握shell命令及其编程和系统程序设计，让我感觉收获很大。实验2的五个实验中后两个虽然花费了我大量的精力，但是通过这两个实验，我对于shell这一强大的工具无论是理解深度还是编程熟练度都有了极大的提升，特别是实验5的系统程序设计更是让我对进程这一Linux操作系统动态执行的基本单元有了深入的了解，学到了很多上课没有涵盖的东西，这让我觉得我所付出的精力都是值得的。

实验1-3的主要目的是熟悉Shell编程，难度不大，但由于初学shell编程，刚上手的时候还是遇到了很多问题。例如：字符串比较时用双中括号比较安全，用单中括号可能会出现结果错误、运算符一侧变量为空时程序出错等问题；“=~”号的右侧不能加引号；变量中含下划线时，引用该变量需要加大括号，等等。不过这些问题都相对较小，通过自己编几个类似的指令在shell中实验，或是查阅网上的资料都可以轻松解决。

实验3是实现两个目录的同步，在写的时候并不是特别熟练，因此写得复杂了，现在看来其实有更简单的写法，即直接遍历目录下的文件，遇到需要递归搜索的目录则直接递归调用自身，当时写的时候也有过这个想法，但因为觉得Linux shell没有临时变量，在递归时会产生一些麻烦而放弃了，事后过了半个月仔细思考发现之前的想法其实是可行的，如果以后还有机会可以进行修改。但反过来，在写这个实验的过程中我觉得我自己的应变能力有了不错的提高，例如自己想出了用Linux shell实现堆栈来存储临时变量，我觉得也是一次不错的尝试。

实验4是写一个简单的教务管理系统，由于Linux对于文件处理和文本处理的强大特性，我大胆尝试了将所有数据储存在文本中，事实证明这一尝试是非常成功的，在写代码时也非常轻松，只是许多本质重复的功能让代码显得冗长了许多。我觉得这一实验中我的收获是训练了清晰的思路，从设计之初就想好了层层递进、即时交互的设计思路，用户菜单是层层递进、逐渐具体的，编写代码也跟随菜单层层递进，这使得我的代码的结构非常清晰，虽然代码很长但依然有着不错的可读性。另外，我还大胆尝试自己加入了一些需求中没有提及的细节，丰富了程序的功能，在此期间也有了一些额外的收获，比如学会了日期的较好的处理方法——转化成时间戳，等等。

实验5对我而言是不小的挑战。在实验前，我先仔细阅读了上课的pdf、Neil Matthew和Richard Stones的《Linux程序设计》一书和《Unix环境高级编程》一书，但依然遇到了不少的麻烦。具体如下：

- 首先是程序的架构。一开始我打算根据命令解析结果直接实现具体的指令，最后实现管道。但写了一会儿就发现管道操作涉及到了进程控制、前后台管理和串联具体指令等多个部分，必须先加以考虑，因此我改变了思路。这给我的教训是在动手写之前必须要仔细分析需求，形成清晰的思路。
- 其次是对于一些命令和函数的理解不够完整造成的问题。例如，我一开始将所有指令都新建一个进程执行，但很快我发现这样做有着巨大的问题：对于一些内嵌指令，如cd，新建子进程后执行是没有意义的，主进程的目录没有被改变。为此，我认识到要将部分命令单独提出来直接执行。再例如，waitpid这个函数的第三个参数为0时，遇到子进程停止（stopped，不是终止terminated）时仍然会阻塞，不会返回值。我利用vs code的单步执行调试发现了这个问题后，一直不知道如何解决，在网上查阅了很多资源，包括stackoverflow上的问题，也没有得到答案。最后我查阅了Linux系统调用的[官方文档](#)，才得以解决。但是，对于一些细节的问题，例如waitpid错误信息的可能成因，官方文档也没有给出特别详细的解释，或是其解释经尝试后证实并非错误的真正成因。在这些问题的解决上，我都是通过自己静下心来一段时间思考可能的成因，最后意识到错误原因来解决的。
- 一个令我印象很深刻的问题：在实现从文件中读入指令并执行的过程中，我写了一个while循环逐行读取文件，在循环体中执行指令的时候设计到了fork()即创建进程的操作。但是，这一操作却导致我的文件被反复读入（例如，文件共有三行，while循环会以此读入第一行、第二行、第三行、第一行、第二行、第三行、...）形成死循环。对此我百思不得其解，于是在stackoverflow上进行了提问，最终我的提问被关联到了一个点击量非常大的[问题](#)。通过阅读这一问题下的回复，我了解到Linux中新建的子进程在退出后会对在读的文件执行一个等价于fseek的操作，这是一个出现在Linux内核的问题，没有办法直接解决。最终我只得换了一种写法，将文件一次性读完储存起来，然后逐行执行。我相信对于这一问题应该有更加优美的解决方法，以后如有机会还可以进行探索。
- 在修改SHELL环境变量的过程中，我发现如果在程序内部调用系统调用函数修改SHELL这一环境变量，使之加入myshell的绝对路径，会造成程序异常退出。经过推理，我认为是SHELL中加入了自身的路径造成了程序自己对自己的调用，从而使得程序异常退出。经过实验，我又发现在运行程序前在bash中修改SHELL环境变量，之后再运行程序，则不会出现这一问题。

虽然最终成功实现了题目需求中的功能，但必须承认我的程序在鲁棒性和完善性上还有很大的提升空间。

最后，我觉得我在这门课上掌握了Linux程序设计这一计算机专业的重要技能，为后续课程的学习中打下了不错的基础，希望在未来课程的学习中学到更多的知识，得到更大的提升。